Neural, Parallel & Scientific Computations 8 (2000) 55-83

BYY Learning System and Theory for Parameter Estimation, Data Smoothing Based Regularization and Model Selection *

Lei Xu
Department of Computer Science and Engineering
Chinese University of Hong Kong, Shatin, NT, Hong Kong, P.R. China
Phone 852 2609 8423, Fax 852 2603 5024, Email lxu@cse.cuhk.edu.hk

ABSTRACT

Bayesian Ying-Yang (BYY) learning system and theory has been developed in recent years by the present author as a unified statistical learning framework for unsupervised learning, supervised learning, and temporal modeling. A general learning problem is formulated as the problem of building a pair of two complementary models, based on two Bayesian decomposition representations of the joint distribution on the space where samples data are observed and the space where samples are represented. Given a set of observed data samples, the pair of models are built up via learning according to a fundamental harmony principle that not only minimizes the mismatch between the two models but also let the matched pair as firm as possible. Under this general framework, we not only revisit several existing major learning approaches with new insights, but also obtain a number of new results for learning tasks of different levels. On the level of parameter estimation, not only both the batch way and adaptive EM-like algorithms are revisited with new applications to the implementation of maximum likelihood learning on radial basis function nets and three layer forward net, but also a new general learning algorithm called Coordinated Competitive Learning *is obtained for training several major unsupervised and supervised learning models. On the level of regularization, an easy-implementing regularization technique called data smoothing learning is obtained for various statistical learning tasks on a size sample set. On the level of model selection, a general principle as well as its specific criteria is proposed for a number of major unsupervised and supervised learning tasks. This paper further systematically elaborate the BYY learning system and theory, not only from the perspectives of system structures, typical architectures, fundamental theory and generalized formulation as well as implementing techniques, but also from its applications to four major types of unsupervised learning tasks and three major types of supervised learning networks, with not only those existing major results revisited as special cases but also a series of new models and new results presented.*

1 Introduction

The basic sprit of simultaneously building up models in both the directions, between the space where samples data are observed and the space where samples are represented, has been adopted in literature of modeling a perception or recognizing learning system for several decades. A typical example is the ART theory developed by Grossberg and colleagues, started at 70's. In the past decade, this sprit has been widely adopted in various modeling methods for perception and learning, examples include Kawato's bi-directional theory, Mumford's pattern theory of vision, Ullman's bi-directional information flow, and Hinton and colleagues' Helmholtz machine, from different viewpoints and for different purposes. In addition, the LMSER self-organizing rule (Xu, 1991&93) can also be regarded as an effort that suggests to use bi-directional modeling for statistical unsupervised learning. Readers are referred to Xu(1999c) for a further reference.

Firstly proposed in 1995(Xu, 1995&96) and continuously developed in the subsequent years (Xu, 1997a&b&c, 1998a&b&c, 1999a&b&c, 1999d&98, 1999e&98), the BYY learning system and theory further formulates the bi-directional modeling sprit into a general statistical framework in two complementary Bayesian representations of the joint distribution on the observation space and representation space. The BYY theory relates to the information geometry theory (Csiszar & Tusnady, 1984; Amari, 1985&95) by sharing a

*This work was supported by the HK RGC Earmarked Grant CUHK 4297/98.

Received July 19, 1999

common feature that both theoreies involve the ingredients of matching two distributions by Kullback divergence and implementing the matching by an EM-like algorithm for parameter learning. Also, a specific adaptive type of the BYY learning on a bidirectional architecture becomes equivalent to the Helmholtz machine learning (Hinton, et al, 1995; Dayan, et al, 1995; Dayan & Hinton, 1996). However, the three studies are made from different perspectives with different purposes. The BYY theory aims at not only empirical parameter learning but also data smoothing based regularization and model selection, while the information geometry theory and Helmholtz machine learning consider empirical parameter learning only. Even focusing our comparison on empirical parameter learning, the information geometry theory considers geometry structures and general properties of matching two distributions by Kullback divergence, while the Helmholtz machine learning considers effective implementation of Kullback divergence matching of two specific densities by forward and backward networks. However, the BYY theory functions as a unified statistical learning framework for unsupervised learning, supervised learning, and temporal modeling, consisting of not only the matching of two distributions by Kullback divergence but also the maximization of a harmony measure. Moreover, the BYY theory focuses on considering the two complementary Bayesian representations and the inner relationship of its components in various architectures as well as their applications in various unsupervised and supervised learning tasks. The three studies are also different in several other aspects, and a detailed discussion are referred to Xu(1999c).

This paper further systematically elaborates the BYY learning system and theory. Sec. 2 introduces the fundamental issues of the system and theory for parameter estimation, data smoothing learning, and model selection, with further discussions on typical architectures, extensions to a general mathematical formulation, and practical implementation techniques. Sec. 3 systematically presents the applications of BYY learning on a number of typical unsupervised learning tasks. Furthermore, Sec. 4 introduces the BYY supervised learning system and theory, and then Sec.5 presents its applications on the studies of mixture-of-expert model, radial basis function net and three layer forward net.

In addition, the temporal BYY learning system and theory has also been developed for modeling signal in a general state space approach, which provides not only a unified point of view on Kalman filter, hidden Markov model (HMM), independent component analysis (ICA) and blind source separation (BSS) with extensions, but also several new results on these studies. Readers are referred to Xu(1999a) for details.

2 BYY Learning System and Theory

2.1 Bayesian Ying-Yang system and unsupervised learning

We describe the joint density $p(x, y)$ on $X \times Y$ by two different models in help of its two complementary Bayesian representations [1]:

$$p_{M_1}(x, y) = p_{M_{y|x}}(y|x)p_{M_x}(x), \quad p_{M_2}(x, y) = p_{M_{x|y}}(x|y)p_{M_y}(y). \tag{1}$$

The model M_1 consists of two components $M_{y|x}$ and M_x. M_x models the density on X, denoted by $p_{M_x}(x)$ or shortly p_{M_x}. $M_{y|x}$ models the density of y conditioning on x, denoted by $p_{M_{y|x}}(y|x)$ or shortly $p_{M_{y|x}}$, from which we get a specific mapping $x \to y$ by one of the following three choices:

$$\hat{y} = \begin{cases} \int y p_{M_{y|x}}(y|x)dy, & \text{(a) regression,} \\ \max_y p_{M_{y|x}}(y|x), & \text{(b) posterior,} \\ \text{randomly pick } y \text{ according to } p_{M_{y|x}}(y|x), & \text{(c) stochastic.} \end{cases} \tag{2}$$

It usually performs (a) tasks of decision, classification, recognition, ..., etc, when y is a discrete number, e.g., $y = 1, \cdots, k$, (b) tasks of encoding, perception, recognition, ..., etc, when y is a k-bit binary code, and (c) tasks of feature extraction or transformation when y is a k-dimension real vector.

[1] A discrete distribution is also described as a density, e.g., $p(y) = q\delta(y-1) + (1-q)\delta(y)$ describes Bernoulli distribution for a binary $y = 0$ or $y = 1$, where $\delta(y) = 0$ for $y \neq 0$, $\delta(y) = \lim_{h \to 0} h^{-1}$ when $y = 0$.

The model M_2 also has two components $M_{x|y}$ and M_y. M_y models the density on Y, denoted by $p_{M_y}(y)$ or shortly p_{M_y}, $M_{x|y}$ models the density of x conditioning on y, denoted by $p_{M_{x|y}}(x|y)$ or shortly $p_{M_{x|y}}$, from which we can also implement a mapping $y \to x$ with a specific $\hat{x}$ obtained by one of three choices in a similar way as given by eq.(2). It usually performs tasks of reconstruction, decoding, imagination, verification, ..., etc.

Interestingly, the formalization eq.(1) compliments to a famous Chinese ancient Ying-Yang philosophy[2]. M_x models the visible space X by p_{M_x} and thus is a Yang, and M_y models the invisible space Y by p_{M_y} and thus is a Ying. While $M_{y|x}$ models a Yang pathway from an observation into an invisible code or representation by $p_{M_{y|x}}$, and $M_{x|y}$ for a Ying pathway from y to x. Moreover, $M_x, M_{y|x}$ form the Yang model M_1 as described by $p_{M_1}(x, y)$, and $M_x, M_{y|x}$ form the Ying model M_2 as described by $p_{M_2}(x, y)$. We call such a pair of Ying-Yang models *Bayesian Ying-Yang (BYY) system.*

The task of specifying all the aspects of the components $p_{M_{y|x}}, p_{M_x}, p_{M_{x|y}}, p_{M_y}$ is called *learning* in a broad sense, which is unsupervised since it is based on a given training set $D_x = \{x_i\}_{i=1}^N$ only.

The input of observation to the system is functioned by p_{M_x}, which is estimated from the training set D_x. Typically, it is given by the Parzen window nonparametric estimate:

$$p_h(x) = \frac{1}{N}\sum_{i=1}^N K_h(x - x_i), \quad e.g., \; K_h(x) = G(x, x_i, hI_d), \tag{3}$$

where $K_h(x)$ is called kernel function (Devroye, et al, 1996) and $h \geq 0$ is a smoothing parameter. A typical kernel function is the above gaussian $G(x, x_i, hI_d)$ of mean x_i and covariance matrix hI_d, where I_d is the $d \times d$ identity matrix and d is the dimension of x. For any type kernel function, we have that $K_h(x)$ becomes a δ-density $\delta(x)$ when $h = 0$. In this case, the Parzen window estimate in eq.(3) becomes the conventional empirical estimate

$$p_0(x) = \frac{1}{N}\sum_{i=1}^N \delta(x - x_i). \tag{4}$$

The estimate $p_h(x)$ in eq.(3) can be regarded as a smoothed modification of $p_0(x)$ by using a kernel with a $h > 0$ to blur out each impulse at x_i. For brevity, we use p_0, p_h to denote $p_0(x), p_h(x)$ whenever there is no confusion. The larger the parameter h is, the more smooth p_h will become. Usually, h is set heuristically. In this paper, a general theory as well as its implementing algorithms will be developed.

The density of y on Y is described by p_{M_y}. A certain structure can be designed for p_{M_y} according to the nature of problem and a priori knowledge. First, we choose the representation form for y. It can be discrete, e.g., a number $y = 1, \cdots, k$ for tasks of decision, classification, recognition, ..., etc, or a k-bit binary code for tasks of *encoding, perception, recognition, ..., etc.* It can also be a k-dimension real vector. Second, we specify a structure in a parametric density form [3] for p_{M_y} with a set θ_y of finite number unknown parameters, where a specific value of θ_y represents a specific density in the family of all the densities that share this given structure.

The structures of $p_{M_{y|x}}, p_{M_{x|y}}$ are also designed according to the nature of problem and a priori knowledge. First, we exclude those degenerated structures with the relationship between x and y broken, i.e., either $p_{M_{x|y}}(x|y) = p_{M_{x|y}}(x)$ or $p_{M_{y|x}}(y|x) = p_{M_{y|x}}(y)$. Then, we choose among two types of structures. One is described by a parametric density with a set of finite number unknown parameters, similar to the above p_{M_y}, e.g., we have parameter sets $\theta_{y|x}$ and $\theta_{x|y}$ for $p_{M_{y|x}}, p_{M_{x|y}}$. The other is called *structure-free*, which means no any structural constraints such that p_{M_a} for each $a \in \{x|y, \; y|x\}$ is free to take any element

[2] It should be "Yin" in the Mainland Chinese spelling. However, "Ying" is used here for keeping the sprit of symmetry. The key point of the philosophy is that every normal entity in the universe involves a harmony interaction between a "Ying" part and a "Yang" part in a twofold sense. Namely, *a best matching* between the Ying and the Yang and *the firmness* of the matched the Ying-Yang pair. Usually, a visible or physical component is called "Yang", and an invisible or spiritual component is called "Ying", which represents a static pair of Ying-Yang spaces. Moreover, there is another dynamic pair of "Ying-Yang" that describes the interactions between this pair of static spaces. E.g., a male animal is such a 'Yang' that functions as a transform from a physical body into an invisible code, and a female animal is a 'Ying' that transfers an invisible code into a next generation physical body.

[3] A discrete distribution is automatically understood as a parametric density since it is always specified by a finite number of parameters.

of $\mathcal{P}_a$, where $\mathcal{P}_{x|y}$ and $\mathcal{P}_{y|x}$ denote the family of all the densities in the form $p(x|y)$ and $p(y|x)$, respectively.

A combination of structures for $p_{M_{y|x}}, p_{M_y}, p_{M_{x|y}}$ specifies a system architecture. There are three typical architectures, featured by the structures of $p_{M_{y|x}}, p_{M_{x|y}}$:

- *Backward architecture* which consists of a parametric density $p_{M_{x|y}}$ for directly implementing the backward pathway, and a structure-free $p_{M_{y|x}}$ with no structure for directly implementing the forward pathway;
- *Forward architecture* which consists of a parametric density $p_{M_{y|x}}$ for directly implementing the forward pathway and a structure-free $p_{M_{x|y}}$;
- *Bi-directional architecture* where both $p_{M_{y|x}}, p_{M_{x|y}}$ are parametric densities for directly implementing the bi-directional pathways.

The case that both $p_{M_{y|x}}, p_{M_{x|y}}$ are structure-free is useless because both pathways can not be implemented.

A specific architecture with the structures of all the component specified is called a model M. For a given $M = M(\theta)$, what remains to be further specified is a set $\theta = \{\theta_{y|x}, \theta_y, \theta_{x|y}, h\}$ of all the unknown parameters. We use *parameter learning* to refer to the task of determining a specific value θ^*, and correspondingly use $M^* = M(\theta^*)$ to denote a specification of M by θ^*. Usually, we consider a set of models $\{M_k\}$ that share a same architecture and a same structure on each component but in different complexity scales. Typically, such a set is obtained by enumerating k as the representation complexity. Sometimes, we even can get a set $\{M_k\}$ that shares the same architecture with same structure on each component under a same complexity scale, obtained by enumerating only a number of specific values taken by some parameters in θ. We use *model selection* to refer to the task of selecting a particular k^* among a set of given models $\{M_k(\theta^*)\}$.

2.2 Bayesian Ying-Yang learning theory

The analogy between the BYY system and the Chinese Ying-Yang philosophy motivates us to use "Ying-Yang harmony" as a fundamental learning principle for implementing *parameter learning* and *model selection* in the BYY system. That is, we let the Ying model p_{M_2} and the Yang model p_{M_1} to be best harmony in a twofold sense. First, the difference between the two models should be minimized since they model the same joint density on $X \times Y$. Second, the obtained Ying-Yang pair should be as firm or confident as possible in its representation of the observation D_x.

To make a mathematical formulation, we consider the information theoretic measures:

$$\begin{aligned} &\mathcal{D}(p\|q) = \int p(x) \ln [p(x)/q(x)]dx, \\ &H(p) = -\int p(x) \ln p(x)dx, \ H(p(x|u)) = -\int p(x|u) \ln p(x|u)dx, \end{aligned} \tag{5}$$

where $\mathcal{D}(p\|q)$ is the Kullback-Leibler divergence that measures the difference between two densities p, q, $H(p)$ is the entropy that measures the uncertainty of the density p, and $H(p(x|u))$ denotes the entropy of the conditional density $p(x|u)$.

We adopt the Kullback-Leibler divergence to measure the difference between p_{M_1} and p_{M_2} in eq.(1) with p_{M_x} given by eq.(3), that is,

$$\mathcal{D}(M) = \mathcal{D}(p_{M_1}\|p_{M_2}) = \mathcal{D}^-(M) - H(p_h) \tag{6}$$

where $H(p_h)$ depends on h only and is irrelevant to θ, k. Specifically we have

$$\mathcal{D}^-(M) = \int p_h(x)\mathcal{D}^-(M|x)dx, \ \ \mathcal{D}^-(M|x) = \int p_{M_{y|x}}(y|x) \ln \frac{p_{M_{y|x}}(y|x)}{p_{M_{x|y}}(x|y)p_{M_y}(y)}dy. \tag{7}$$

Moreover, we consider the entropy $H(p_{M_{y|x}}(y|x))$. For each x, the er the $H(p_{M_{y|x}}(y|x))$ is, the more sure the mapping $x \to y$ is, and thus the more confident we are on the representation of D_x by the Ying-Yang pair. Therefore, we use $H_e = \int p_h(x)H(p_{M_{y|x}}(y|x))dx$ to measure the uncertainty of the representation on a given data D_x by the Ying-Yang pair.

With the above two measures, the implementation of the Ying-Yang harmony learning principle becomes an optimization task that minimizes both $\mathcal{D}(M)$ and $H(p_{M_{y|x}}(y|x))$ by maximizing a *harmony* measure $\mathcal{H}(M)$ as follows:

$$-\mathcal{H}(M) = \mathcal{D}(p_{M_1}||p_{M_2}) + H_e, \quad H_e = \int p_h(x) H(p_{M_{y|x}}(y|x))dx, \tag{8}$$

which is equivalent to

$$\mathcal{H}(M) = -H(p_h) + \mathcal{H}^-(M), \quad \mathcal{H}^-(M) = \int p_{M_1} \ln p_{M_2} dxdy. \tag{9}$$

At a fixed h, $\max \mathcal{H}(M)$ is equivalent to $\max \mathcal{H}^-(M)$. When $p_{M_{y|x}}$ is fixed, $\max \mathcal{H}^-(M)$ makes p_{M_2} be as close as possible to $p_h(x)p_{M_{y|x}}(y|x)$ and finally reach it when both $p_{M_{x|y}}, p_{M_y}$ are free of structure. While when p_{M_2} is fixed, $\max \mathcal{H}^-(M)$ makes $p_{M_{y|x}}$ locate around the modes of $p_{M_2}(x,y)$ for each given x. Particularly, when $p_{M_{y|x}}$ is free of structure, it becomes the δ-density at the mode of $p_{M_2}(x,y)$, that is, for each x we have

$$\begin{aligned} &p_{M_{y|x}}(y|x) = \delta(y - \hat{y}), \ \hat{y} = arg\max_y p_{M_2}(x,y), \\ &\mathcal{H}^-(M) = \hat{\mathcal{H}}^-(M), \ \hat{\mathcal{H}}^-(M) = \int p_h(x) \ln p_{M_2}(x,\hat{y})dx. \end{aligned} \tag{10}$$

Thus, from this perspective we also see that $\max \mathcal{H}(M)$ makes the Ying-Yang pair match firmly (i.e., keep a minimal structural complexity).

For a set of specific models $\{M_k(\theta)\}$, with the notations $\mathcal{D}(M) = \mathcal{D}(\theta,k)$ and $\mathcal{H}(M) = \mathcal{H}(\theta,k)$, we consider the following two choices for implementing $\max \mathcal{H}(M)$:

$$(A) \max_{\theta,k} \mathcal{H}(\theta,k), \qquad (B) \max_{\theta,k} \mathcal{H}(\theta,k), \textit{subject to } \mathcal{D}(\theta,k) = min \textit{ at each } k. \tag{11}$$

For the choice (A), we maximize $\mathcal{H}(\theta,k)$ with respect to both θ,k. While for the choice (B), the maximization of $\mathcal{H}(\theta,k)$ is under a constraint that $\mathcal{D}(\theta,k)$ is minimized with respect to θ at each k, which consents to the convention that first determines parameters by minimizing mismatch and then selects models by a criterion for both mismatch and model complexity.

Usually, both of the choices in eq.(11) are performed sequentially in the two steps:

(1) The **first** step is *parameter learning* for θ^* at a fixed k. Specifically, we have

$$\theta^* = \begin{cases} arg\max_\theta \ \mathcal{H}(\theta,k), & \text{for the choice (A)}, \\ arg\min_\theta \ \mathcal{D}(\theta,k), & \text{for the choice (B)}. \end{cases} \tag{12}$$

We refer the choice (A) shortly by *$\mathcal{H}$-learning* and the choice (B) by *$\mathcal{D}$-learning*. The $\mathcal{H}$-learning tends to enforce the forward pathway $p_{M_{y|x}}$ to be as simple or compact as possible. Thus, in comparison with the $\mathcal{D}$-learning, the *$\mathcal{H}$-learning* is imposed by a regularization that reduces structural complexity.

(2) The **second** step is *model selection*. For both the choices (A) & (B), we get

$$k^* = arg\min_k \ J(k), \ J(k) = -\mathcal{H}(\theta^*,k). \tag{13}$$

In addition, it should be noticed that usually $\mathcal{D}(p||q) \neq \mathcal{D}(q||p)$ for eq.(5), which will lead us to other variants of mathematical formulations. E.g., we can also use

$$\mathcal{D}(M) = \mathcal{D}(p_{M_2}||p_{M_1}) \tag{14}$$

to measure the difference between p_{M_1} and p_{M_2}. Correspondingly, we can use the entropy $H(p_{M_2})$ to measure the uncertainty of the representation on D_x by the Ying-Yang pair. That is, we have

$$-\mathcal{H}(M) = \mathcal{D}(p_{M_2}||p_{M_1}) + H(p_{M_2}). \tag{15}$$

which is equivalent to

$$\mathcal{H}(M) = \int p_{M_2} \ln p_{M_1} dxdy, \tag{16}$$

With these $\mathcal{D}(M) = \mathcal{D}(\theta,k)$ and $\mathcal{H}(M) = \mathcal{H}(\theta,k)$, we can again implement eq.(12) and eq.(13).

2.3 Empirical learning versus data smoothing learning

In eq.(12), the estimation θ^* consists of the searching of an optimal smoothing parameter h^*, which includes the special case that we simply prefix $h = 0$. We call eq.(12) at $h = 0$ *empirical learning* because it is made directly on the i.i.d. empirical samples $D_x = \{x_i\}_{i=1}^N$. In contrast, we call eq.(12) with h^* searched *data smoothing learning* because $p_{h^*}(x)$ can smooth out those disturbances in $p_0(x)$. The role of p_{h^*} in place of p_0 can be further understood by using a gaussian kernel as in eq.(3). In this case, we have

$$p_{h^*}(x') = \int p_0(x) G(x' - x, 0, h^* I_d) dx, \quad x' = x + \varepsilon. \tag{17}$$

That is, $p_{h^*}(x')$ is the density of the sum x' of two independent variables x from p_0 and ε from $G(\varepsilon, 0, h^* I_d)$. In other words, the data smoothing learning is equivalent to the empirical learning on a noise blurred data set of a large enough size N':

$$D_x^h = \{x_i'\}_{i=1}^{N'}, \; x_i' = x_i + \varepsilon, \; \varepsilon \text{ is from } G(\varepsilon, 0, hI_d), \tag{18}$$

where D_x^h is a data set that depends on the given value of h.

It is well understood in literature that adding noise into data is a type of regularization that improves the performances in the cases of size of samples.

The regularization role of the *data smoothing learning* can also be understood by a further analysis on eq.(6) and eq.(9). For a fixed h, we can ignore $H(p_h)$. Thus, eq.(12) becomes equivalent to

$$\theta^{-*} = \begin{cases} arg\max_{\theta^-} \; \mathcal{H}^-(\theta, k), & \text{for the choice (A)}, \\ arg\min_{\theta^-} \; \mathcal{D}^-(\theta, k), & \text{for the choice (B)}, \end{cases} \quad \theta^- = \theta - \{h\}, \tag{19}$$

with the notations $\mathcal{D}^-(M) = \mathcal{D}^-(\theta, k)$ and $\mathcal{H}^-(M) = \mathcal{H}^-(\theta, k)$. It further becomes empirical learning at $h = 0$ with $\mathcal{H}^-(\theta, k), \mathcal{D}^-(\theta, k)$ replaced by $\mathcal{H}^-(\theta, k)|_{h=0}, \mathcal{D}^-(\theta, k)|_{h=0}$, in the sense of

$$T(M) = \int p_h(z) T(z) dz, \quad T(M)|_{h=0} = \tfrac{1}{N} \textstyle\sum_{i=1}^N T(z_i). \tag{20}$$

We consider a kernel $K_h(x) > 0$ with

$$\int K_h(x) dx = 1, \; \int x K_h(x) dx = 0, \; \int x^2 K_h(x) dx = h. \tag{21}$$

A typical example of such kernel is gaussian of zero mean and covariance matrix hI.

We further consider the 2nd order Taylor expansion at each sample $x_i = [x_{i,1}, \cdots, x_{i,d}]^T$:

$$\begin{aligned} & T(x) \approx T(x_i) + (x - x_i)^T \nabla_x T(x)|_{x=x_i} + \frac{1}{2}(x - x_i)^T H_i (x - x_i), \\ & H_i \text{ is the Hessian of } T(x) \text{ at } x = x_i, \\ & \int K_h(x - x_i) T(x) dx = T(x_i) + 0.5h \textstyle\sum_{j=1}^d \dfrac{\partial^2 T(x_i)}{\partial x_{i,j}^2}, \\ & T(M) = \int p_h(x) T(x) dx = T(M)|_{h=0} + 0.5h \tfrac{1}{N} \textstyle\sum_{i=1}^N \sum_{j=1}^d \dfrac{\partial^2 T(x_i)}{\partial x_{i,j}^2}. \end{aligned} \tag{22}$$

E.g., for $\mathcal{D}^-(M) = \mathcal{D}^-(\theta, k)$ in eq.(7) we have

$$\mathcal{D}^-(M) = \mathcal{D}^-(M)|_{h=0} + 0.5hR_D, \quad R_D = \tfrac{1}{N} \textstyle\sum_{i=1}^N \sum_{j=1}^d \partial^2 \mathcal{D}^-(M|x_i) / \partial x_{i,j}^2. \tag{23}$$

Therefore, the *data smoothing learning* is equivalent to the empirical learning plus a Tikhonov-type regularization term R_D controlled by the smooth parameter h. In the literature of regularization methods, the parameter h in the position of eq.(23) is usually called hyper-parameter. For those existing regularization methods (Girosi, et al, 1995; Mackay, 1992), the task of estimating an appropriate hyper-parameter is usually a hard job that is tedious to handle. In contrast, the data smoothing learning eq.(12) solves this problem in a different framework that is easy to implement, which is introduced as follows.

We start at a simple but generally applicable technique, based on eq.(17) and eq.(18). Here, we only describe the case for the $\mathcal{D}$-learning in eq.(12) but the details are same for the case of the $\mathcal{H}$-learning. Let D_d^h to denote D_x^h given in eq.(18), the best h^* is searched by the following iterative procedure:

$$\begin{aligned} &Step\ 1: \text{ For a fixed } h, \text{ get } D_d^h \text{ and get } J(h) = \mathcal{D}^-(h) - H(p_h), \\ &Step\ 2: \text{ Get } h^{new} = h^{old} + \delta h, \text{ with } \delta h = \begin{cases} \eta, & \text{if } J(h_x^{old} + \eta) < J(h_x^{old}), \\ -\eta, & \text{if } J(h_x^{old} - \eta) < J(h_x^{old}). \end{cases} \end{aligned} \quad (24)$$

where $\eta > 0$ is a stepsize, it follows from eq.(7) that $\mathcal{D}^-(h)$ and $H(p_h)$ in Step 1 are given by

$$\begin{aligned} &\mathcal{D}^-(h) = \frac{1}{N'}\sum_{i=1}^{N'} \mathcal{D}^-(M^*|x_i'), \quad \mathcal{D}^-(M^*|x) = \mathcal{D}^-(M|x)|_{\theta^*}, \\ &H(p_h) \approx -\frac{1}{N'}\sum_{i=1}^{N'} \ln p_h(x_i'), \end{aligned} \quad (25)$$

where $\theta^* = \{\theta^{-*}, h = 0\}$ is obtained from performing $\min_{\theta^-} \mathcal{D}^-(\theta^*, k)|_{h=0}$ on the noisy data D_x^h, which is equivalent to performing $\min_{\theta^-} \mathcal{D}^-(\theta, k)$ at h on the original data D_x. Here and in the sequel, when we say that we perform maximization or minimization of a cost $T(M)$ as shown in eq.(20) on a data set $D = \{z_i\}_{i=1}^N$, we mean that based on this data set we get $p_h(z)$ by eq.(3) or $p_0(z)$ by eq.(4).

Either or both $\mathcal{D}^-(h)$ and $H(p_h)$ can be estimated alternatively:

(1) We can also approximately get θ^* from performing $\min_\theta \mathcal{D}^-(M)|_{h=0}$ on the original data D_x. Moreover, after ignoring $\mathcal{D}^-(M)|_{h=0}$ in eq.(23) which is irrelevant to h, in eq.(24) we can simply use $\mathcal{D}^-(h) = 0.5hR_D$, where R_D is obtainable for some specific architectures, as will be further introduced latter.

(2) We can also estimate $H(p_h)$ based on D_x directly. For a gaussian kernel, we get $H(p_h) = \frac{1}{N}\sum_{i=1}^N \int G(x, x_i, hI_d) \ln p_h(x)dx = c_0 - 0.5d\ln h + \frac{1}{N}\sum_{i=1}^N \int G(x, x_i, hI_d) \ln \sum_{i=1}^N e^{-0.5\|x-x_i\|^2/h} dx$, where c_0 is irrelevant to h and thus can be ignored. In help of eq.(22) and ignoring terms that are irrelevant to h, we have

$$\begin{aligned} &H(p_h) = -0.5d\ln h + D(h) + 0.5\frac{h_0 - e_0}{h}, \quad D(h) = \frac{1}{N}\sum_{i=1}^N \ln \sum_{j=1}^N S(x_i), \\ &h_0 = \frac{1}{N}\sum_{i=1}^N\sum_{j=1}^N p_j(x_i)\|x_i - x_j\|^2, \quad e_0 = \frac{1}{N}\sum_{i=1}^N \|x_i - \sum_{j=1}^N p_j(x_i)x_j\|^2, \\ &S(x) = e^{\frac{-0.5\|x-x_j\|^2}{h}}, \quad p_i(x) = \frac{e^{\frac{-0.5\|x_i-x_j\|^2}{h}}}{\sum_{j=1}^N S(x_i)}. \end{aligned}$$

Moreover, lumping up the above $H(p_h)$ in eq.(26) and $\mathcal{D}^-(h) = 0.5hR_D$, instead of using Step 2 in eq.(24), we can also update h in gradient descent way as follows:

$$h^{new} = h^{old} - \eta\frac{\partial J(h)}{\partial h}, \quad J(h) = 0.5hR_D + H(p_h), \quad \frac{\partial J(h)}{\partial h} = 0.5R_D + 0.5(\frac{e_0}{h^2} - \frac{d}{h}), \quad (26)$$

where the relevance of $p_j(x_i)$ to h is ignored. Furthermore, we can even directly consider the two roots of $\frac{\partial J(h)}{\partial h} = 0$, i.e., $R_D h^2 - dh + e_0 = 0$ and take the one that is a minimum point of $J(h)$ as an initialization of eq.(26) or eq.(24).

However, the above discussions from eq.(19) up to now is not applicable to the formulation eqs.(14)&(15)&(16), for which we can not avoid the integral over x. Instead, the Monte-Carlo technique should be used to handle the integrals, as will be further discussed in Sec.2.5.

2.4 $\mathcal{H}$-learning versus $\mathcal{D}$-learning on forward and backward architectures

The situations we encounter on a bi-directional architecture are same as the general settings discussed in the previous subsections. However, specific features and further insights can be found on the forward and backward architectures.

1. Backward architecture

It consists of a parametric density $p_{M_{x|y}}$ for implementing the backward pathway and a structure-free $p_{M_{y|x}}$ for the forward pathway. This architecture describes how to generate or reconstruct the data D_x by the marginal density

$$p_{M_x^c}(x) = \int p_{M_{x|y}}(x|y)p_{M_y}(y)dy, \tag{27}$$

and thus is also called *generative or reconstruction architecture.*

• **$\mathcal{D}$-learning** The minimization of $\mathcal{D}(M)$ with respect to the structural free $p_{M_{y|x}}$ results in

$$\begin{aligned}
&p_{M_{y|x}}(y|x) = p_{M^c_{y|x}}(y|x), \quad p_{M^c_{y|x}}(y|x) = \frac{p_{M_{x|y}}(x|y)p_{M_y}(y)}{p_{M_x^c}(x)}, \\
&\mathcal{D}(M) = \begin{cases} \mathcal{D}(p_h\|p_{M_x^c}), & \text{for eq.(6)}, \\ \mathcal{D}(p_{M_x^c}\|p_h), & \text{for eq.(14)}; \end{cases} \\
&-\mathcal{H}(M) = \begin{cases} \mathcal{D}(p_h\|p_{M_x^c}) + \int p_h(x)H(p_{M^c_{y|x}}(y|x))dx, & \text{for eq.(8)}, \\ \mathcal{D}(p_{M_x^c}\|p_h) + H(p_{M_2}), & \text{for eq.(15)}. \end{cases}
\end{aligned} \tag{28}$$

Moreover, $\min_{\theta_{x|y},\theta_y,h}\mathcal{D}(M)$ can be implemented either directly by gradient descent or by iterating two steps:

$$\begin{aligned}
&\textit{Step 1}: \quad \text{with } \theta_{x|y},\ \theta_y \text{ fixed, update } h \text{ to reduce } \mathcal{D}_1(M), \text{ get } p_{M_{y|x}}(y|x), \\
&\textit{Step 2}: \quad \text{with } h \text{ fixed, update } \theta_{x|y},\ \theta_y \text{ to reduce } \mathcal{D}_2(M), \\
&\textit{Where} \quad \mathcal{D}_1(M) = \begin{cases} \mathcal{D}(p_h\|p_{M_x^c}), & \text{for eq.(6)}, \\ -\int p_{M_x^c}\ln p_h(x)dx, & \text{for eq.(14)}; \end{cases} \\
&\qquad\qquad \mathcal{D}_2(M) = \begin{cases} -\int p_h(x)\ln p_{M_x^c}dx, & \text{for eq.(6)}, \\ \mathcal{D}(p_{M_x^c}\|p_h), & \text{for eq.(14)}. \end{cases}
\end{aligned} \tag{29}$$

We make a further consideration on the case of eq.(7). It follows that

$$\mathcal{D}(p_h\|p_{M_x^c}) = \mathcal{D}^-(M) - H(p_h), \quad \mathcal{D}^-(M) = -\int p_h(x)\ln p_{M_x^c}(x)dx. \tag{30}$$

Thus, from eq.(20) we see that the *empirical learning* $\min_{\{\theta_{x|y},\theta_y,\}}\mathcal{D}^-(M)|_{h=0}$ becomes exactly the conventional maximum likelihood (ML) learning on the marginal density $p_{M_x^c}(x)$ of the Ying model. Particularly, for a discrete $y = 1,\cdots,k$, this $p_{M_x^c}(x)$ is a finite mixture. In this special case, eq.(28) becomes exactly the Expectation Maximization(EM) algorithm (Dempster, et al, 1977), which is currently popular in the literature of neural networks.

The data smoothing learning $\min_{\{\theta_{x|y},\theta_y,h\}}\mathcal{D}(M)$ can be regarded as a regularized extension of the ML learning. The detailed implementation for updating h can be made, as discussed in the previous subsection. Moreover, in this special case we can get

$$R_D \approx -\frac{1}{N}\sum_{i=1}^N\|e_i\|^2, \ e_i = \int p_{M^c_{y|x}}(y|x)\nabla_x \ln p_{M_{x|y}}(x_i)dy, \tag{31}$$

by approximately using a general property $E[\nabla(\nabla\ln p(x))] = -E[\nabla\ln p(x)(\nabla\ln p(x))^T]$.

• **$\mathcal{H}$-learning** For the $\mathcal{H}$-learning in eq.(12) with eq.(16), $\max\mathcal{H}(M)$ with respect to the structure-free $p_{M_{y|x}}$ will result in the same $\mathcal{H}(M)$ in eq.(28).

Furthermore, , for each fixed h the $\mathcal{H}$-learning in eq.(12) with eq.(9) leads us to eq.(10). Corresponding to eq.(29), we can also implement this type of $\mathcal{H}$-learning by alternatively repeating the following two steps:

$$\begin{aligned}
&\textit{Step 1}: \quad \text{for a sample } x, \ \textit{fixed } \theta^-, \ \textit{get } \hat{y} \textit{ by eq.}(10), \\
&\textit{Step 2}: \quad \text{update } \theta_{x|y} \text{ to increase } \ln p_{M_{x|y}}(x|\hat{y}), \text{ update } \theta_y \text{ to increase } \ln p_{M_y}(\hat{y}),
\end{aligned} \tag{32}$$

where each sample x comes either directly from the original data set D_x for empirical learning or from the noisy data set D_x^h by eq.(18) for data smoothing learning. Due to

$\max_y p_{M_2}(x,y) = \max_y p_{M^c_{y|x}}(y|x)$ for each x, we see that $\delta(y-\hat{y})$ is actually a winner-take-all (WTA) version of $p_{M^c_{y|x}}(y|x)$. Thus, eq.(10) can be regarded as a hard-cut version of eq.(29). Particularly, when $h=0$, eq.(32) can be regarded as the so called hard-cut EM algorithm, which was previously obtained in Xu(1997a) heuristically.

Actually, $\hat{y} = arg\max_y p_{M_2}(x,y)$ in eq.(10) is a generalized formulation of the conventional WTA. To get a further insight, we equivalently consider

$$\hat{y} = arg\max_y \ln p_{M_2}(x,y), \quad \ln p_{M_2}(x,y) = \ln p_{M_{x|y}}(x|y) + \ln p_{M_y}(y). \tag{33}$$

In the special case that $p_{M_{x|y}} = G(x, m_y, \sigma^2 I_d)$ and $p_{M_y}(y) = 1/k$, we have that $\max_y \ln p_{M_{x|y}}(x|y)$ is equivalent to $\hat{y} = min_y \|x_i - m_y\|^2$, which is exactly the conventional WTA used in many competitive learning algorithms. The WTA in eq.(33) not only considers the general form $\max_y \ln p_{M_{x|y}}(x|y)$, but also coordinately considers the effect of $\ln p_{M_y}(y)$ that imposes a priori preference. Thus, we call this type of WTA by *Coordinated Competition* and the learning eq.(32) by *Coordinated Competitive Learning (CCL)*, which acts as a general formulation of many existing WTA-based competitive learning algorithms.

• **Model selection** After the parameter learning eq.(12), with the obtained θ^*, h^* we can make model selection by eq.(13) with $\mathcal{H}(M)$ given by eq.(28) or directly by eq.(9) or eq.(16), except for the $\mathcal{H}$-learning based on eq.(9) for which we insert $\hat{\mathcal{H}}^-(M)$ given by eq.(10) into eq.(9) to get $\mathcal{H}(M)$ for eq.(13). Particularly, if the parameter learning is empirical, i.e., $h=0$, we can directly use $\hat{\mathcal{H}}^-(M)$ given in either eq.(9) or eq.(10) as $\mathcal{H}(M)$ for model selection by eq.(13).

2. Forward architecture

It consists of a parametric $p_{M_{y|x}}$ for implementing the forward pathway and a structure-free $p_{M_{x|y}}$. This architecture describes how to represent an input x by an inner representation y via eq.(2), and thus is called *representative or recognition architecture.*

• **$\mathcal{D}$-learning** Since $p_{M_{x|y}}$ is structure-free, $\min_{p_{M_{x|y}}} \mathcal{D}(M)$ results in

$$p_{M_{x|y}} = p_{M^c_{x|y}}, \quad p_{M^c_{x|y}}(x|y) = \frac{p_{M_{y|x}}(y|x)p_h(x)}{p_{M^c_y}(y)}, \quad p_{M^c_y}(y) = \int p_{M_{y|x}}(y|x)p_h(x)dx,$$

$$\mathcal{D}(M) = \begin{cases} \mathcal{D}(p_{M^c_y}\|p_{M_y}), & \text{for eq.(7)}, \\ \mathcal{D}(p_{M_y}\|p_{M^c_y}), & \text{for eq.(14)}; \end{cases}$$

$$-H(M) = \begin{cases} \mathcal{D}(p_{M^c_y}\|p_{M_y}) + \int p_h(x) H(p_{M_{y|x}}(y|x))dx, & \text{for eq.(8)}, \\ \mathcal{D}(p_{M_y}\|p_{M^c_y}) + H(p_{M_2}), & \text{for eq.(15)}. \end{cases} \tag{34}$$

Again, $\min_{\{\theta_{y|x}, \theta_y, h\}} \mathcal{D}(M)$ can be implemented either directly by gradient descent or by iterating two steps:

$$\begin{aligned} &Step\ 1 && : \text{with } \theta_{y|x},\ h \text{ fixed, get } p_{M^c_y} \text{ and update } \theta_y \text{ to reduce } \mathcal{D}_1(M), \\ &Step\ 2 && : \text{with } \theta_y \text{ fixed, update } \theta_{y|x}, \text{and } h \text{ to reduce } \mathcal{D}_2(M), \end{aligned} \tag{35}$$

$$Where \quad \mathcal{D}_1(M) = \begin{cases} -\int p_{M^c_y} \ln p_{M_y} dy, & \text{for eq.(6)}, \\ \mathcal{D}(p_{M_y}\|p_{M^c_y}), & \text{for eq.(14)}. \end{cases}$$

$$\mathcal{D}_2(M) = \begin{cases} \mathcal{D}(p_{M^c_y}\|p_{M_y}), & \text{for eq.(6)}, \\ -\int p_{M_y} \ln p_{M^c_y} dy, & \text{for eq.(14)}. \end{cases}$$

The empirical learning is featured by $p_{M^c_y}(y)|_{h=0}$ in the sense of eq.(20). The data smoothing learning is its regularized extension and the detailed implementation for updating h is made as discussed in the previous subsection. In this special case, we also have

$$R_D = \frac{1}{N}\int (1 + \ln \frac{\sum_{i=1}^N p_{M_{y|x}}(y|x_i)}{N p_{M_y}(y)}) \sum_{i=1}^N \sum_{j=1}^d \frac{\partial^2 p_{M_{y|x}}(y|x_i)}{\partial x_{i,j}^2} dy. \tag{36}$$

• **$\mathcal{H}$-learning** For the $\mathcal{H}$-learning in eq.(12) with eq.(9), $\max \mathcal{H}(M)$ with respect to the structure-free $p_{M_{x|y}}$ will result in the same $\mathcal{H}(M)$ as in eq.(34).

However, for the $\mathcal{H}$-learning in eq.(12) with eq.(16), when $p_{M_x}(x) = p_0(x)$ as in eq.(4) and p_{M_y} is also structure-free, $\max \mathcal{H}(M)$ with respect to p_{M_2} will result in

$$\hat{y} = arg \max_y p_{M_{y|x}}(y|x_i), \quad p_{M_2}(x,y) = \begin{cases} \delta(y-\hat{y})\delta(x-x_i), & \text{at } x = x_i, \\ 0, & \text{otherwise;} \end{cases}$$
$$\mathcal{H}(M) = \hat{\mathcal{H}}(M), \quad \hat{\mathcal{H}}(M) = \frac{1}{N}\sum_{i=1}^{N} \ln p_{M_{y|x}}(\hat{y}|x_i), \tag{37}$$

which is a general formulation of the representative model based WTA-competitive learning.

• **Model selection** After the parameter learning eq.(12), we can make model selection by eq.(13) with $\mathcal{H}(M)$ given by eq.(34) or directly by eq.(9) or eq.(16), except for the $\mathcal{H}$-learning based on eq.(16) for which we use $\mathcal{H}(M) = \hat{\mathcal{H}}(M)$ given by eq.(37).

2.5 Generalized formulations

We consider a general measure called f-Divergence, which was first introduced by Csiszar in 1967 and a nice introduction can be found in (Devroye, et al, 1996):

$$\mathcal{D}(p\|q) = \int p(x) f(\frac{q(x)}{p(x)})dx, \quad \frac{d^2 f(u)}{d^2 u} > 0 \text{ on } [0,\infty) \text{ with } f(1) = 0. \tag{38}$$

By Jensen's inequality, $\mathcal{D}(p\|q) \geq 0$ and $\mathcal{D}(p\|q) = 0$ if $p = q$. This f-Divergence includes several interesting special cases:

$$\mathcal{D}(p\|q) = \begin{cases} \int |p(x) - q(x)|dx, & \text{if } f(x) = |x-1|, \text{ variational distance,} \\ 2(1 - \int \sqrt{p(x)q(x)}dx), & \text{if } f(x) = (1-\sqrt{x})^2, \text{ Hellinger distance,} \\ \int \frac{p^2(x)}{q(x)}dx - 1, & \text{if } f(x) = (x-1)^2, \chi^2\text{-divergence,} \\ \int p(x) \ln \frac{p(x)}{q(x)}dx, & \text{if } f(x) = -\ln x, \text{ Kullback-Leibler divergence.} \end{cases} \tag{39}$$

That is, the Kullback-Leibler divergence in eq.(5) is a special case of f-Divergence.

Correspondingly, we can also extend the entropy definition in eq.(5) by replacing $-\ln x$ with $f(x)$ defined in eq.(38), that is,

$$H(p) = \int p(x) f(p(x))dx, \quad H(p(x|u)) = \int p(x|u) f(p(x|u))dx. \tag{40}$$

We can directly generalize the definition of $\mathcal{D}(p_{M_1}\|p_{M_2})$ with $\mathcal{D}(p\|q)$ given by eq.(38). While, for extending the definition of $\mathcal{H}(M)$, we have two choices. One is made by using eq.(40) as $H(p_{M_{y|x}}(y|x))$ in eq.(8) and as $H(p_{M_2})$ in eq.(15). The other is obtained directly from eq.(9) or eq.(16) by replacing $-\ln x$ with $f(x)$, that is,

$$\mathcal{H}^-(M) = -\int p_{M_1} f(p_{M_2})dxdy, \quad or \quad \mathcal{H}(M) = -\int p_{M_2} f(p_{M_1})dxdy. \tag{41}$$

Most of the previous discussions directly apply to the generalized formulations by replacing $-\ln x$ with $f(x)$, except those cases based on the property $\ln xy = \ln x + \ln y$. In these cases, we should return from $\ln x + \ln y$ back to $\ln xy$. E.g., in eq.(29) and eq.(35), we should let $\mathcal{D}_1(M) = \mathcal{D}_2(M) = \mathcal{D}(M)$ with

$$\mathcal{D}(M) = \begin{cases} \mathcal{D}(p_h\|p_{M_x^c}), & \text{for eq.(6),} \\ \mathcal{D}(p_{M_x^c}\|p_h), & \text{for eq.(14),} \end{cases} \quad for\ eq.(29);$$
$$\mathcal{D}(M) = \begin{cases} \mathcal{D}(p_{M_y^c}\|p_{M_y}), & \text{for eq.(6),} \\ \mathcal{D}(p_{M_y}\|p_{M_y^c}), & \text{for eq.(14),} \end{cases} \quad for\ eq.(35). \tag{42}$$

2.6 Empirical implementation vs. Monte-Carlo implementation

In both parameter learning and model selection, $\mathcal{D}(M), \mathcal{H}(M)$ in the forms of $T(M)$ in eq.(20) can get rid of the computational difficulty of the integral over x with $p_0(x)$ given by eq.(4) obtained either directly on the original data set D_x when $h = 0$ or indirectly on a noisy data set D_x^h given by eq.(18). Moreover, if y is discrete or the integral over y is analytically solvable, we can totally get rid of the computational difficulty of integrals.

However, in many cases we encounter the integral of a general type $\int T(x, y, \theta)dxdy$. In the sequel, we propose to deal with such integrals by the Monte-Carlo technique, either in a batch way or in an adaptive way.

(a) *Batch Monte-Carlo sampling.* We make random sampling according to a known reference density $p^r(x)$ to get a set of samples $\{x_i^r\}_{i=1}^{N^r}$ with a large enough N^r. Then, we get an empirical density $p_0^r(x)$ based on this data set and obtain

$$\int T(x, y, \theta)dxdy \approx \int \frac{p_0^r(x)}{p^r(x)} T(x, y, \theta)dxdy = \frac{1}{N^r}\sum_{i=1}^{N^r} \int \frac{1}{p^r(x_i^r)} T(x_i^r, y, \theta)dy. \quad (43)$$

Typically, we can use p_h by eq.(3) as $p_r(x)$, which includes as a special case the previously discussed technique of handling $\mathcal{D}(M), \mathcal{H}(M)$ in the forms of $T(M)$ as in eq.(20) in help of $p_0(x)$ by eq.(4) obtained indirectly on a noisy data set D_x^h by eq.(18).

The integrals over y is simply the summations when y is discrete. Otherwise, we can turn an integral over y into a summation in a way similar to the above discussed case for the integrals over x.

(b) *Adaptive Monte-Carlo sampling.* We get a sample x^r from $p_r(x)$ which could be either the current estimate $p_{M_x^c}$ or $p_{M_{x|y}}$. Similarly, we can use p_{M_y} or $p_{M_{y|x}}$ to get an y_r. Then, we minimize the cost functional $\int C(x, y, \theta)dxdy$ by stochastic approximation via updating

$$\theta^{new} = \theta^{old} - \eta\nabla_\theta[C(x^r, y^r, \theta)/(p_r(x^r)p_r(y^r))], \ \eta > 0 \text{ is a learning stepsize.} \quad (44)$$

Next, we use θ^{new} to get new estimates of $p_r(x)$ and $p_r(y)$ for sampling the next x^r, y^r.

3 Applications to Unsupervised Learning Tasks

3.1 BYY inversion and posteriori density estimation

1. Revisit of old results

Item 4.1 (*Bayesian inverse mapping and Bayesian classifier*) On backward architecture with $p_{M_{x|y}}, p_{M_y}$ known, we have from eq.(28) that $p_{M_{y|x}} = p_{M_{y|x}^c}$, which is exactly the Bayesian inverse of the mapping $y \to x$. Particularly, this $p_{M_{y|x}}$ gives the conventional Bayesian classifier (Duda & Hart, 1972) when $y = 1, \cdots, k$.

2. New results

Item 4.2 (*BYY inversion*) For the tasks that need to compute $p_{M_{y|x}^c}(y|x_i)$, by eq.(28) for each sample x_i, e.g., for implementing an inverse mapping $x_i \to y$, we have to calculate $p_{M_x^c}(x)$, which is usually costly either when $y = [y_1, \cdots, y_k]$ is a binary code with a large k or when y is real and either of $p_{M_{x|y}}$ and p_{M_y} is nongaussian. This problem not only increases computing cost but also is hard to make an on-line mapping $x_i \to y$.

The problem can be solved by a special case of the bi-directional architecture where $p_{M_{x|y}}$ and p_{M_y} are already known. In this case, via $\min_{\theta_{y|x}, h} \mathcal{D}(M)$ we directly implement $x_i \to y$ according to eq.(2) by parametric model $p_{M_{y|x}}$. We call this technique *BYY Inversion*, which can be made either with $h = 0$ or a best h searched as in Sec. 2.3.

3.2 BYY finite mixture, clustering and unsupervised classification

All these tasks share a common key feature that y is a finite integer, or equivalently $y = 1, \cdots, k$. In this case, $p_{M_x^c}$ in eq.(27) becomes a finite mixture of k individual distributions $p_{M_{x|y}}, y = 1, \cdots, k$, each of which could represent a model, a cluster or a pattern class.

1. Revisit of old results

Item 4.3 (*The EM algorithm for ML learning on finite mixture*) On a backward architecture with $p_{M_{x|y}}$, p_{M_y} unknown, as discussed after eq.(30), that the *empirical learning* $\min_{\{\theta_{x|y},\theta_y\}} \mathcal{D}^-(M)|_{h=0}$ becomes exactly the conventional maximum likelihood (ML) learning on a finite mixture $p_{M_x^c}(x)$ (including Gaussain mixture) and eq.(28) becomes exactly the EM algorithm (Dempster, et al, 1977), which is currently quite popular in the literature of neural networks (Csiszar & Tusnady, 1984; McLachlan, & Basford, 1988, Neal & Hinton, 1993; Amari, 1995; Xu & Jordan, 1996).

Item 4.4 (*Least square clustering, K-means algorithm and elliptic clustering*) In the case that $h = 0$ and $p_{M_{x|y}} = G(x, \mu_y, \Sigma_y)$ is gaussian, the $\mathcal{H}$-learning eq.(10) becomes exactly the conventional least square clustering when $p_{M_y} = 1/k$ and the covariance matrices of $p_{M_{x|y}}$ are $\Sigma_y = \sigma_y^2 I, y = 1, \cdots, k$. Also, the algorithm eq.(32) becomes equivalent to the well known K-means algorithm (Duda & Hart, 1972). Moreover, from eq.(10) we get the so called Mahalanobis distance or elliptic clustering when $\Sigma_y \neq \sigma_y^2 I$, with eq.(32) becoming an extended K-means algorithm. In addition, further variants can be obtained when $p_{M_y} \neq 1/k$. These results were also obtained previously in Xu(1997a) from a heuristic perspective called hard-cut Gaussian mixture.

2. New results

Item 4.5 (*Model number selection on finite mixture*) In company with the above ML learning on finite mixture and the EM algorithm in Item 4.3, starting from eq.(13) with $\mathcal{H})(M)|_{h=0}$ given by eq.(9), we are lead to a new criterion for selecting the number k of distributions in a finite mixture (particularly of gaussians in a gaussian mixture), as shown in (Xu, 1997a). While, in company with the above $\mathcal{H}$-learning in Item 4.4, with some simplifications we are also lead to a new criterion for selecting the number k of clusters in the least square clustering by the k-means algorithm, as well as in its extensions to elliptic clustering. The criteria were also obtained previously in Xu(1997a) from the perspective of hard-cut Gaussian mixture.

Item 4.6 (*Data smoothing learning*) For the cases of a size of samples, in help of data smoothing learning as in Sec.2.3, we get a regularized ML learning on finite mixture (particularly a gaussian mixture with $p_{M_{x|y}} = G(x, \mu_y, \Sigma_y), p_{M_y} = \alpha_y, y = 1, \cdots, k$). In implementation, for each fixed $h \neq 0$, we have two choices to make $\min_{\{\theta_{x|y},\theta_y\}} \mathcal{D}^-(M)$. One is running the EM algorithm on a noisy data set D_x^h as given by eq.(18). The other, as shown in Xu (1998a), is still made on the original data D_x but in help of the following modified EM algorithm:

$$\begin{aligned} &\textit{Step } 1: && \text{with } p_{M_y}(y) = \alpha_y \text{ and } \mu_y, \Sigma_y, \text{ fixed; get } p_{M^c_{y|x}}(y|x_i) \text{ by eq.(28)}, \\ &\textit{Step } 2: && \text{update } \alpha_y = \tfrac{1}{N}\textstyle\sum_{i=1}^N p_{M^c_{y|x}}(y|x_i), \quad m_y = \tfrac{1}{\alpha_y N}\sum_{i=1}^N p_{M^c_{y|x}}(y|x_i)x_i, \\ & && \Sigma_y = hI_d + \tfrac{1}{\alpha_y N}\textstyle\sum_{i=1}^N p_{M^c_{y|x}}(y|x_i)(x_i - m_y)(x_i - m_y)^T. \end{aligned} \tag{45}$$

The updating on h can be made as in Sec.2.3. Particularly, when eq.(26) is used, R_D in eq.(31) becomes simply $e(x_i) = \sum_{y=1}^k p_{M^c_{y|x}}(y|x_i)\Sigma_y^{-1}(x_i - m_y)$, which is the average of the normalized error of using m_y to represent x_i.

Furthermore, after data smoothing parameter learning, we can select k by directly using the criteria discussed in Item 4.5.

Item 4.7 (*Max-BMC principle*) On a forward structure, from eq.(34) we can get its special case at $h = 0$ by

$$-\mathcal{H}(M)|_{h=0} = \int p_{M_y^c}(y)|_{h=0} \ln \frac{p_{M_y^c}(y)|_{h=0}}{p_{M_y}(y)} dy - \tfrac{1}{N}\textstyle\sum_{i=1}^N \int p_{M^c_{y|x}}(y|x_i) \ln p_{M^c_{y|x}}(y|x_i) dy. \tag{46}$$

When $p_{M_y}(y)$ is a uniform distribution, we get exactly the heuristically proposed *Maximum Balanced Mapping Certainty* (Max-BMC) principle by eqn(5) and eqn(7) in Xu(1996). Moreover, when $h \neq 0$, by eq.(34) with a uniform distribution $p_{M_y}(y)$, we can get a regularized Max-BMC learning from eq.(8) as well as its variants from eq.(15).

Moreover, from eq.(8) with $H(p_{M_{y|x}})$ as in eq.(40) and $\mathcal{D}(p_{M_1}||p_{M_2})$ as in eq.(39), we get

$$-\mathcal{H}(M)|_{h=0} = \int p_{M_y^c}(y)|_{h=0} f(p_{M_y}(y)/p_{M_y^c}(y)|_{h=0})dy - \frac{1}{N}\sum_{i=1}^{N}\int p_{M_{y|x}^c}(y|x_i) f(p_{M_{y|x}^c}(y|x_i))dy, \quad (47)$$

Which is the general (Max-BMC) principle by eqns(5),(3),(4) in Xu(1996). Particularly, when $f(r)$ is given by the third choice in eq.(39), we have the Nonlinear Maximum Variance (NMV) learning rule (Xu, 1995).

Item 4.8 (*Structural regularization on finite mixture*) We can also use a parametric $p_{M_{y|x}}$ that introduces a type of structural regularization or constraint on $p_{M_x^c}$ by eq.(28). Usually, $p_{M_{y|x}}$ can be designed by a forward structure $z = [z_1, \cdots, z_k] = f(x, W)$ which is followed by a soft-max transformation $o_y = e^{z_y}/\sum_{j=1}^{k} e^{z_j}$.

3.3 Factor analysis, ICA and dependence reduction

The common key feature shared by these tasks is the factorial density

$$p_{M_y}(y) = \prod_{j=1}^{k} p_{M_y}(y_j), \quad for\ y = [y_1, \cdots, y_k]. \quad (48)$$

Thus, the Ying model M_2 provides a *independent factor model* $p_{M_x^c}$ by eq.(28) on how x is generated from k hidden independent factors; while the Yang model M_1 generally provides a *dependence reduction* mapping $x \to y$ in the sense that the resulted $p_{M_y^c}$ by eq.(34) is close to $p_{M_y}(y)$ by eq.(48), and particularly provides an *ICA* mapping $x \to y$ when $p_{M_y^c} = \prod_{j=1}^{k} p_{M_y^c}(y_j)$.

1. Revisit of old results

Item 4.9 (*ML factor models, multiple cause models, factor analysis and PCA*) On a backward architecture, the *empirical learning* $\min_{\{\theta_{x|y}, \theta_y\}} \mathcal{D}^-(M)|_{h=0}$ is equivalent to the ML learning on a general independent factor model $p_{M_x^c}$ by eq.(28). This model relates to the multiple cause model (Saund, 1995; Dayan & Zemel, 1995) when both y and x are binary codes and $p_{M_{x|y}}$ is specifically designed, and reduces to the conventional linear orthogonal normal factor model (Anderson, et al, 1956) when $p_{M_y}(y_j) = G(y_j, 0, \lambda_j)$ and $p_{M_{x|y}} = G(x, Ay, \sigma^2 I_d)$, $AA^T = I$ (Xu, 1998c). Moreover, the latter case with $p_{M_{y|x}} = p_{M_{y|x}^c}$ by eq.(28) is equivalent to the principal component analysis (PCA) (Xu, 1998c).

Item 4.10 (*LMSER self-organization and Helmholtz machine*) On a bi-directional architecture with y_j taking binary value 0 or 1, as shown in Xu (1998b), the *empirical learning* $\min_{\{\theta_{x|y}, \theta_y\}} \mathcal{D}^-(M)|_{h=0}$ will reduce into not only the Helmholtz machine learning of one hidden layer structure (Hinton, et al, 1995; Dayan, et al, 1995; Dayan & Hinton, 1996) when x is also a binary code and $p_{M_{y|x}}, p_{M_{x|y}}$ are both given by an one-layer-net of sigmoid activation units, i.e., given by eqn.(12) and eqn.(17) in Xu (1998b), but also a generalized LMSER self-organization rule on real input x when $p_{M_{x|y}}$ and $p_{M_{y|x}}$ are specifically designed. The original LMSER learning was firstly proposed in (Xu, 1991&1993) and has been later applied to ICA under the name of nonlinear PCA with success (Karhunen and Joutsensalo, 1994). This original LMSER rule is actually just an additive part (and thus a rough approximation) of the generalized LMSER rule.

Item 4.11 (*MMI-ICA, Informax-ICA, LPM-ICA and MCA*) On a forward architecture with $\mathcal{D}(p_{M_y^c}||p_{M_y})$ given by eq.(34), as shown in Xu (1998b), $\min_{\theta_{y|x}} \mathcal{D}(p_{M_y^c}||p_{M_y})|_{h=0}$ with θ_y prefixed will reduce into the following special cases: **(a)** the Informax-ICA (Bell & Sejnowski, 1995) when $p_{M_y}(y_j)$ is uniform on $[0, 1]$ and $p_{M_{y|x}} = \delta(y - S(Wx))$ is deterministic with $S(z) = [s(z_1), \cdots s(z_k)]^T$ for $z = Wx$ and an invertible W, where $0 < s(r) < 1$ is a prefixed sigmoid function, **(b)** the minimization mutual information (MMI) ICA (Amari et al, 1996) when $p_{M_y}(y_j)$ is given by a truncated Gram-Charlier series and $p_{M_{y|x}} = \delta(y - Wx)$ is deterministic with an invertible W, and **(c)** the minor

component analysis (MCA) when $p_{M_y}(y_j) = G(y_j, 0, \lambda_j)$ and $p_{M_{y|x}} = \delta(y - Wx), W^tW = I$ (Xu, 1999c). In these cases, $\min_{\theta_{y|x}} \mathcal{D}(p_{M_y^c}\|p_{M_y})|_{h=0}$ with θ_y prefixed, becomes the maximization of

$$J(W, \theta_y) = \frac{1}{N}\sum_{i=1}^{N}[\ln|W| + \ln p_{M_y}(y)|_{y=Wx_i}], \tag{49}$$

with respect to W under prefixed density form of p_{M_y}.

If we also adapt θ_y instead of prefixing it, i.e., $\min_{\{\theta_{y|x},\ \theta_y\}} \mathcal{D}(p_{M_y^c}\|p_{M_y})|_{h=0}$, we get exactly the learned parametric model(LPM) based ICA (Xu, et al, 1998f).

2. New results

Item 4.12 (*New algorithms for the linear orthogonal normal factor model and new variants of linear and logistic binary factor models*) In Xu(1998b&c), we have also get both a new batch algorithm and an adaptive algorithm for the conventional linear orthogonal normal factor model (Anderson, et al, 1956). Moreover, from the general factor model $p_{M_x^c}$ eq.(28), we can get various specific factor models. E.g., for binary y_j with $p_{M_y}(y_j) = q^{y_j}(1-q)^{1-y_j}$, the design of $p_{M_{x|y}} = G(x, Ay, \sigma^2 I_d)$ for a real noisy observation x leads us to a linear binary factor model that is different from the multiple cause model (Saund, 1995; Dayan & Zemel, 1995). In Xu(1998b), a simple adaptive algorithm is proposed to implement this model and works well on binary source separation. Furthermore, if we design $p_{M_{x|y}} = s(z_i)^{x_i}(1 - s(z_i))^{1-x_i}, z = Ay$ for a binary observation x, where $0 < s(r) < 1$ is a sigmoid function, we can get a logistic binary factor model and an adaptive algorithm for its implementation(Xu, 1998b). Generally, from $p_{M_x^c}$ by eq.(28), we can also get various other specific factor models, based on different designs of $p_{M_y}(y_j)$ and $p_{M_{x|y}}$.

Item 4.13 (*Linear non-orthogonal factor models and generalizations of LMSER*) On a bi-directional architecture, when $p_{M_{y|x}} = G(y, f(Wx), \Lambda_{y|x})$ and $p_{M_y} = G(y, m_y, \Lambda_y)$, $p_{M_{x|y}} = G(x, Ay, \Sigma_{x|y})$ with $\Lambda_{y|x}, \Lambda_y$ being diagonal and $f(z) = [f(z_1), \cdots f(z_k)]^T$ for $z = Wx$, we have that $\mathcal{D}^-(M)|_{h=0}$ in eq.(7) becomes

$$\mathcal{D}^-(M)|_{h=0} = -\ln\frac{|\Lambda_{y|x}|}{|\Lambda_y||\Sigma_{x|y}|} - k + \frac{1}{N}\sum_{i=1}^{N}(f(Wx) - m_y)^T\Lambda_y^{-1}(f(Wx) - m_y) \tag{50}$$
$$+Tr[\Lambda_y^{-1}\Lambda_{y|x}] + Tr[\Sigma_{x|y}^{-1}A\Lambda_{y|x}A^T] + \frac{1}{N}\sum_{i=1}^{N}(x_i - Af(Wx_i))^T\Sigma_{x|y}^{-1}(x_i - Af(Wx_i)).$$

which is a correction of a typo in eqn.(47) of Xu(1998c) where $Tr[\Sigma_{x|y}^{-1}A\Lambda_{y|x}A^T]$ was absent.

For the special cases with $f(u) = u$, the minimization of the above $\mathcal{D}^-(M)|_{h=0}$ provides a general scheme for linear non-orthogonal normal factor model, which reduces to the conventional orthogonal model (Anderson, et al, 1956) if $\Lambda_y = I, A^TA = I_k$.

More generally, when $A = W^T$, $\Sigma_{x|y} = \sigma_{x|y}^2 I_d$ and f is a sigmoid function, the minimization of the last term in eq.(50) is equivalent to the LMSER rule originally proposed in Xu(1993). Thus, from eq.(50) we can also get various generalizations of the original LMSER rule. In addition, on a specific design, the $\mathcal{D}$-learning eq.(19) with eq.(50) will become eqns.(44)&(47) in Xu (1998b) which give us another type of generalization of the LMSER rule.

Item 4.14 (*Non-invertible ICA, nonlinear ICA and noisy ICA*) On a forward architecture, as discussed in Item 4.11, $\min_{\theta_{y|x}} \mathcal{D}(p_{M_y^c}\|p_{M_y})_{h=0}$ given by eq.(34) will reduce into the existing MMI based ICA, Informax ICA , LPM ICA approaches when $p_{M_{y|x}}$ becomes a deterministic δ-density by an invertible deterministic mapping $y = Wx$.

Moreover, as proposed firstly in Xu(1997c) and then re-elaborated in Xu(1998b&99a), these ICA approaches can also be easily extended to not only the linear mapping $y = Wx$ with a non-invertible matrix W of $k \times d, k < d$ but also a deterministic nonlinear mapping $y = f(x, W)$, with eq.(49) becoming

$$J(W, \theta_y) = \frac{1}{N}\sum_{i=1}^{N}[0.5\ln|WW^T| + \ln p_{M_y}(y)|_{y=Wx_i}], \tag{51}$$
$$J(W, \theta_y) = \frac{1}{N}\sum_{i=1}^{N}[0.5\ln|D_fD_f^T| + \ln p_{M_y}(y)|_{y=f(x_i,W)}], \quad D_f = \partial f(x_i, W)/\partial x_i^T.$$

Furthermore, for a bi-directional architecture with the forward pathway still being a deterministic δ-density as above, but with the backward pathway being $p_{M_{x|y}} = G(x, Ay, \sigma^2)$, we get the so called noisy ICA model $x = Ay + e$ with e being gaussian noise from $G(e, 0, \sigma^2)$. In this case, $\min_{\{\theta_{y|x}, \theta_y\}} \mathcal{D}(p_{M_y^c} \| p_{M_y})_{h=0}$ given by eq.(34) becomes the maximization of

$$J(W, A, \sigma^2, \theta_y) = J(W, \theta_y) + \frac{1}{N} \sum_{i=1}^{N} \ln G(x_i, Ay, \sigma^2 I_d)|_{y=Wx_i \text{ or } y=f(x_i,W)}, \tag{52}$$

with respect to W, A, σ^2, θ_y, where $J(W, \theta_y)$ gives the learned parametric model(LPM) based ICA (Xu, et al, 1998f). The maximization of the 2nd term becomes equivalent to the original LMSER rule (Xu, 1991&93) when $A = W^T$, which attempts to estimate the noise variance σ^2 and filter out the effect of noise. Thus, eq.(52) describes a model that combines the features of both LPM-ICA and LMSER for handling the noise ICA problem. By gradient approach, it is easy to get an adaptive algorithm for maximizing eq.(52). as shown in (Xu, 1999a).

Item 4.15 (*Probabilistic ICA and dependence reduction*) When the mapping $x \to y$ by $p_{M_{y|x}}$ is probabilistic instead of deterministic, the minimization of $\mathcal{D}(M)$ in eq.(34) can be regarded as a generalization of the cases discussed in the above Items 4.11 and 4.14 if we use a probabilistic model $p_{M_{y|x}}$ to replace the deterministic forward mapping such that $p_{M_y^c}$ given by eq.(34) best approximates $p_{M_y}(y)$ given by eq.(48). If the resulted $p_{M_y^c}(y)$ becomes $\prod_{j=1}^{k} p_{M_y^c}(y_j)$, the mapping by $p_{M_{y|x}}$ realizes a probabilistic ICA mapping $x \to y$. Generally, even if $p_{M_y^c}(y) = \prod_{j=1}^{k} p_{M_y^c}(y_j)$ does not hold exactly, the mapping $x \to y$ that lets $p_{M_y^c}(y)$ to best approximate $p_{M_y}(y)$ also realizes dependence reduction to some extent.

Item 4.16 (*Factor number selection and model selection*) Generally speaking, we can always use eq.(13) with $\mathcal{H}(M) = \mathcal{H}^-(M)$ by eq.(9) to select the number k of independent factors or sources, that is, we have:

$$\begin{aligned} &\min_k \; J(k), \quad J(k) = -\mathcal{H}^-(M^*), \\ &\mathcal{H}^-(M) = \int p_{M_{y|x}}(y|x) p_h(x) \ln [p_{M_{x|y}}(x|y) p_{M_y}(y)] dx dy. \end{aligned} \tag{53}$$

On a bi-directional architecture, we can directly use eq.(53) for selecting k. Specifically, we can develop criteria for selecting the number of hidden units in Helmholtz machine, LMSER self-organization and its generalization as well as other bi-directional architectures, some examples are referred to (Xu, 1998b).

On a backward architecture, we can use eq.(53) with $p_{M_{y|x}}$ given by eq.(28) to decide the factor number k for the general factor model $p_{M_x^c}$ eq.(28). Moreover, detailed specific criteria can be obtained for different designs on $p_{M_y}(y_j)$ and $p_{M_{x|y}}$. For examples, a specific criterion is obtained in Xu(1998c) for selecting factor number of the conventional linear orthogonal factor model and another specific criterion is obtained in Xu(1998b) for selecting the number of independent binary and real factors.

On a forward architecture, we can also develop specific criteria from eq.(53) for detecting the number of independent components in the above mentioned ICA problems. When $p_{M_{y|x}}$ is not a δ-density, e.g., the above mentioned probabilistic ICA case, we can get $p_{M_{x|y}}$ by eq.(34), and then use eq.(53) for selecting k^*. However, when $p_{M_{y|x}}$ is a δ-density, e.g, the above mentioned MMI-ICA, Informax-ICA, LPM-ICA cases, we can not directly use eq.(53). Instead, we consider the reconstruction $\hat{x} = W^+ y$ by the pseudo inverse W^+ of W with a gaussian error $x - \hat{x}$, that is,

$$\begin{aligned} &p_{M_{x|y}} = G(x, W^+ y, \sigma^2 I_d), \quad W^+ = W^T (W W^T)^{-1}, \\ &\sigma^2 = \frac{1}{dN} \sum_{i=1}^{N} \|x_i - W^T (W W^T)^{-1} W x_i\|^2 \end{aligned} \tag{54}$$

For the cases of noise-free observation $\{x_i\}_{i=1}^N$, as k increases from 1 to d, σ^2 will decrease until becoming zero at a $k^* \leq d$. In this case, we can select this k^* as the number of independent components. Moreover, when the observation $\{x_i\}_{i=1}^N$ contains noise, we put the above $p_{M_{x|y}}$ into eq.(53) for selecting k. At $h = 0$ we get $\mathcal{H}(M) =$

$\frac{1}{N}\sum_{i=1}^{N}\ln[p_{M_{x|y}}(x|Wx_i)p_{M_y}(y)|_{y=Wx_i}]$. Further ignoring those terms which are irrelevant to k, we have

$$J(k) = 0.5d\ln\sigma^{*\,2} - \frac{1}{N}\sum_{i=1}^{N}\ln p_{M_y^*}(y)|_{y=W^*x_i}, \quad \sigma^2 \text{ is given by eq.(54).} \tag{55}$$

Item 4.17 (*The $\mathcal{H}$-learning and the CCL Based factor modeling*) On a backward architecture, the $\mathcal{H}$-learning in eq.(12) leads to eq.(10), which provides a coordinated competitive learning (CCL) technique for a general independent factor model $p_{M_x^\xi}$ by eq.(28) with $p_{M_y}(y)$ given by eq.(48). This technique is adaptively implemented by the algorithm eq.(32). Specifically, its Step 2 can be simply

$$\theta_{x|y}^{new} = \theta_{x|y}^{old} + \eta\nabla_{\theta_{x|y}}\ln p_{M_{x|y}}(x|\hat{y}), \quad \theta_y^{new} = \theta_y^{old} + \eta\sum_{j=1}^{k}\nabla_{\theta_y}\ln p_{M_y}(y_j), \tag{56}$$

and the 1st updating equation becomes an adaptive least square updating rule when $p_{M_{x|y}}$ is gaussian.

Moreover, the number of hidden factors is selected by

$$\min_k J(k), \quad J(k) = -\frac{1}{N}\sum_{i=1}^{N}\max_y[\ln p_{M_{x|y}}(x_i|y) + \sum_{j=1}^{k}\ln p_{M_y}(y_j)]. \tag{57}$$

3.4 BYY dimension reduction and data structure mining

1. Revisit of old results

Item 4.18 (*Principal subspace dimension reduction*) As indicated above, the factor model $p_{M_x^\xi}$ eq.(28) will lead us to the conventional linear orthogonal normal factor model (Anderson, et al, 1956) and PCA when $p_{M_y}(y_j) = G(y_j, 0, \lambda_j)$ and $p_{M_{x|y}} = G(x, Ay, \sigma^2 I_d)$, $AA^T = I$ (Xu, 1998c). Actually, PCA is a widely used dimension reduction tool in the cases of $k < d$, where k, d are the dimensions of y and x, respectively.

2. New results

Item 4.19 (*Determination of principal subspace dimension and selection*) For any dimension reduction mapping $x \to y$, we need to decide an appropriate dimension $k^* < d$. When the first k^* principal components by PCA is used for dimension reduction, in Xu (1998c) we got a criterion for deciding this dimension k^* which is equivalent to the number of factors in the conventional linear orthogonal normal factor model.

Similarly, we can also use ICA to get $k^* < d$ independent components for dimension reduction propose. We can perform ICA by the $k \times d$ matrix W at each k and select k^* that makes $J(k)$ by eq.(55) become the minimum. Particularly, for the cases of noise-free observation, we can select k^* simply by either increasing k from a k_0 (e.g., $k_0 = 1$) until σ^2 given by eq.(54) becomes zero or decreasing k from d until $\sigma^2 \neq 0$ with $k+1$ as k^*.

Alternatively, we can also perform ICA by an invertible $d \times d$ matrix W for only once, and then for each k, we select k column vectors of W such that $J(k)$ by eq.(55) or σ^2 by eq.(54) reaches the est. We call the corresponding k components as *principal independent components*. Then, among different values of k we select k^* that makes $J(k)$ by eq.(55) become the minimum. Particularly, for the cases of noise-free observation, we can either increase k from a k_0 until σ^2 becomes zero or decrease k from d until $\sigma^2 \neq 0$ with $k+1$ as k^*. Here, we encounter a typical combinatorial optimization problem. We can approximately handle it by sequentially adding or removing one column vector of W.

Item 4.20 (*A general BYY system for dimension reduction and structure mining*) We consider a general hidden data structure as follows:

$$p_{M_y}(y) = \sum_{r=1}^{n_y}\alpha_{y,r}p_{M_{y,r}}(y - m_{y,r}), \quad p_{M_{y,r}}(y - m_{y,r}) = \prod_{j=1}p_{M_{y,r}}(y_j - m_{y,r,j}). \tag{58}$$

which returns to eq.(48) when $n_y = 1$. Generally, it represents a mixture of n_y independent distributions that locates at different m_r. Therefore, in correspondence to Sec.3.3, we can get various localized extensions of ICA and independent factor models.

Specifically, we can consider the following factorial gaussian mixtures:

$$p_{M_{y,r}}(y_j - m_{r,j}) = G(y_j, m_{y,r,j}, \sigma^2_{y,r,j}),$$
$$p_{M_{x|y}} = \sum_{r=1}^{n_{x|y}} \alpha_{x|y,r} G(x, A_r y, \Sigma_{x|y,r}), \quad p_{M_{y|x}} = \sum_{r=1}^{n_{y|x}} \alpha_{y|x,r} p_{M_{y|x,r}}(y|x),$$
$$p_{M_{y|x,r}}(y|x) = \prod_{j=1} p_{M_{y|x,r}}(y_j|x), \quad p_{M_{y|x,r}}(y_j|x) = G(y_j, w^T_{r,j}x, \sigma^2_{y|x,r,j}), \tag{59}$$

from which we can get local linear normal factor model and local PCA.

4 Supervised BYY Learning

4.1 Supervised BYY learning system and theory

To learn a mapping $x \to z$ based on a data set $D_{zx} = \{x_i, z_i\}_{i=1}^N$, as shown in [24], we consider a pair of the enlarged Yang model and Ying model

$$p_{M_1}(x, y, z) = p_{M_{y|x,z}}(y|x, z) p_{M_{x,z}}(x, y), \quad p_{M_2}(x, y, z) = p_{M_{z|x,y}}(z|x, y) p_{M_{x,y}}(x, y) \tag{60}$$

as the *BYY supervised learning system*, which consists of the following components:

(1) $p_{M_{x,z}}$ is estimated by the Parzen window kernel estimation based on D_{zx}:

$$p_{M_{x,z}} = p_h(x, z) = p_{h_x}(x) p_{h_z}(z|x), \; h = \{h_x, h_z\}, \tag{61}$$

where $p_{h_x}(x)$ is still given by eq.(3) but with a notation change of h into the current h_x, and it is only necessary to define $p_{h_z}(z|x)$ at $x = x_i$:

$$p_{h_z}(z|x) = K_{h_z}(z - z_i), \; at \; x = x_i. \tag{62}$$

(2) $p_{M_{z|x,y}}$ describes the big Ying pathway $x, y \to z$ and is always a parametric model, while $p_{M_{y|x,z}}$ describes the big Yang pathway $x, z \to y$, and can be either a parametric model or free of structure that can take any density in the form $p(y|x, z)$.

(3) $p_{M_{x,y}}(x, y)$ has three choices as follows:

$$p_{M_{x,y}}(x, y) = \begin{cases} p_{M_{y|x}}(y|x)p(x), & \text{Type (a)}, \\ p_{M^c_{y|x}}(y|x)p(x), & \text{Type (b)}, \\ p_{M_{x|y}}(x|y)p_{M_y}(y), & \text{Type (c)}. \end{cases} \quad p_{M^c_{y|x}} \text{ is same as in eq.(28)}, \tag{63}$$

where $p(x)$ is the unknown true densities of $D_x = \{x_i\}_{i=1}^N$, and $p_{M_{y|x}}, p_{M_{x|y}}, p_{M_y}$ are same as in eq.(1).

This BYY supervised learning system implements the mapping $x \to z$ via p_{M_2} by

$$p_M(z|x) = \frac{p_M(z, x)}{p_M(x)} = \int p_{M_{z|x,y}}(z|x, y) p_H(y|x) dy, \tag{64}$$
$$p_M(z, x) = \int p_{M_2}(x, y, z) dy, \; p_M(x) = \int p_{M_{x,y}}(x, y) dy,$$
$$p_H(y|x) = \begin{cases} p_{M_{y|x}}(y|x), & \text{Type (a)}, \\ p_{M^c_{y|x}}(y|x), & \text{Types (b)\&(c)}. \end{cases}$$

Thus, the structure of $p_M(z|x)$ is featured by not only which type $p_H(y|x)$ is, but also what kind of structure that $p_{M_{z|x,y}}(z|x, y)$ takes.

We consider two typical structures for $p_{M_{z|x,y}}(z|x, y)$. One is called *modular structure*, denoting the case that $y = 1, \cdots, k$ and the mapping $x \to z$ is implemented by k individual modules $p_{M_{z|x,y}}(z|x, y), y = 1, \cdots, k$, which are combined subject to the gating of $p_H(y|x)$. The other is called *cascade structure*, denoting the case that $y = [y_1, \cdots, y_k]^T$ and $p_{M_{z|x,y}}(z|x, y) = p_{M_{z|y}}(z|y)$, which can only implement the mapping $y \to z$ that cascades the mapping $x \to y$ by $p_H(y|x)$.

Again, we use a model $M = M(\theta)$ to refer a specific combination of all the components in eq.(60), with their structures specified according to the nature of problem and a priori

knowledge. We use *parameter learning* to refer to the task of determining a specific value θ^* that includes a best $h^* = \{h_x^*, h_z^*\}$, and use *model selection* to refer to the task of selecting a particular k^* among a set of given models $\{M_k(\theta^*)\}$. We still use the "Ying-Yang harmony" learning principle for specifying θ, k, which is still implemented by eq.(12) and eq.(13), but in eq.(6), eq.(8) and eq.(9) we let p_{M_1}, p_{M_2}, h to be given by eq.(60) and eq.(61).

Specifically, we have

(a) For eq.(6), similar to eq.(7) we have

$$\begin{aligned}
&H(p_h) = H(p_h(x,z)), \quad \mathcal{D}^-(M) = L(h_x) + \mathcal{D}_*^-(M),\\
&\mathcal{D}_*^-(M) = \int p_h(x,z)\mathcal{D}_*^-(M|x,z)dxdz,\\
&L(h_x) = \begin{cases} -\int p_{h_x}(x)\ln p(x)dx, & \text{Types (a)\&(b)},\\ 0, & \text{Type (c)}; \end{cases}\\
&\mathcal{D}_*^-(M|x,z) = \begin{cases} \int p_{M_{y|x,z}}(y|x,z)\ln\frac{p_{M_{y|x,z}}(y|x,z)}{p_{M_{z|x,y}}(z|x,y)p_{M_{y|x}}(y|x)}dxdydz, & \text{Type (a)},\\ \int p_{M_{y|x,z}}(y|x,z)\ln\frac{p_{M_{y|x,z}}(y|x,z)}{p_{M_{z|x,y}}(z|x,y)p_{M^c_{y|x}}(y|x)}dxdydz, & \text{Type (b)},\\ \int p_{M_{y|x,z}}(y|x,z)\ln\frac{p_{M_{y|x,z}}(y|x,z)}{p_{M_{z|x,y}}(z|x,y)p_{M_{x|y}}(x|y)p_{M_y}(y)}dxdydz, & \text{Type (c)}. \end{cases}
\end{aligned} \tag{65}$$

Though it contains the unknown true density $p(x)$, the term $L(h_x)$ is only relevant to h_x and thus can be ignored for each fixed h_x.

(b) For eq.(8) and eq.(9), we have

$$\begin{aligned}
&H_e = \int p_h(x,z)H(p_{M_{y|x,z}}(y|x,z))dxdz, \quad \mathcal{H}^-(M) = -L(h_x) + \mathcal{H}_*^-(M),\\
&\mathcal{H}_*^-(M) = \int p_{M_1}\ln p_{M_2}dxdydz = \int p_h(x,z)\mathcal{H}_*^-(M|x,z)dxdz,\\
&\mathcal{H}_*^-(M|x,z) = \begin{cases} \int p_{M_{y|x,z}}(y|x,z)\ln[p_{M_{z|x,y}}(z|x,y)p_{M_{y|x}}(y|x)]dy, & \text{Type (a)},\\ \int p_{M_{y|x,z}}(y|x,z)\ln[p_{M_{z|x,y}}(z|x,y)p_{M^c_{y|x}}(y|x)]dy, & \text{Type (b)},\\ \int p_{M_{y|x,z}}(y|x,z)\ln[p_{M_{z|x,y}}(z|x,y)p_{M_{x|y}}(x|y)p_{M_y}(y)]dy, & \text{Type (c)}. \end{cases}
\end{aligned} \tag{66}$$

where $L(h_x)$ is given by eq.(65) and still ignored for each fixed h.

4.2 Empirical learning versus data smoothing learning

Similarly, we use *empirical learning* to refer the case of eq.(12) at $h = 0$ and *data smoothing learning* to refer to the case of eq.(12) with a best h^* searched. The *empirical learning* is still implemented by eq.(19) with $\mathcal{H}^-(\theta,k), \mathcal{D}^-(\theta,k)$ replaced by $\mathcal{H}^-(\theta,k)|_{h=0}, \mathcal{D}^-(\theta,k)|_{h=0}$, in the sense of eq.(20), where $\mathcal{H}^-(M) = \mathcal{H}_*^-(M)$ is given by eq.(66) and $\mathcal{D}^-(M) = \mathcal{D}_*^-(M)$ is given by eq.(65) since $L(h_x)$ is irrelevant to θ^- and thus ignored.

The regularization role of the *data smoothing learning* can still be understood as discussed in Sec.2.3. Specifically, eq.(23) becomes

$$\begin{aligned}
&L(h_x) = L(h_x)|_{h=0} + 0.5h_xH_p, \quad \mathcal{D}_*^-(M) = \mathcal{D}_*^-(M)|_{h=0} + 0.5(h_xR_D^x + h_zR_D^z),\\
&H_p = \begin{cases} -\frac{1}{N}\sum_{i=1}^N\sum_{j=1}^d \partial^2\ln p(x_i)/\partial x_{i,j}^2, & \text{Types (a)\&(b)},\\ 0, & \text{Type (c)}; \end{cases}\\
&R_D^x = \tfrac{1}{N}\textstyle\sum_{i=1}^N\sum_{j=1}^d \partial^2\mathcal{D}_*^-(M|x_i,z_i)/\partial x_{i,j}^2, \quad R_D^z = \tfrac{1}{N}\sum_{i=1}^N\sum_{j=1}^m \partial^2\mathcal{D}_*^-(M|x_i,z_i)/\partial z_{i,j}^2.
\end{aligned} \tag{67}$$

Similar to Sec.2.3, the simplest way for implementing data smoothing learning is still based on a noise blurred data set of a large enough size N':

$$\begin{aligned}
&D_{xz}^h = \{x_i', z_i'\}_{i=1}^{N'}, \ x_i' = x_i + \varepsilon, \ z_i' = z_i + \zeta,\\
&\varepsilon \text{ is from } G(\varepsilon, 0, h_xI_d), \quad \zeta \text{ is from } G(\zeta, 0, h_zI_m),
\end{aligned} \tag{68}$$

where D_{xz}^h is a data set that depends on the given value of h.

We can still use the algorithm eq.(24) for updating h, but with $D_d^h = D_{xz}^h$ and eq.(25) being replaced by

$$\mathcal{D}^-(h) = \mathcal{D}_*^-(M^*)|_h + L^-(h), \quad H(p_h) \approx -\frac{1}{N'}\sum_{i=1}^{N'} \ln p_h(x_i', z_i'),$$
$$L^-(h) = \begin{cases} 0.5h_x H_p, & \text{Types (a)\&(b)}, \\ 0, & \text{Type (c)}; \end{cases}$$
$$H_p = \begin{cases} 0, & \text{When } p(x) \text{ is uniform or } \ln p(x) \text{ is linear to } x, \\ \sum_{j=1}^{d} 1/s_j^2, & \text{otherwise}; \end{cases}$$
$$\mathcal{D}_*^-(M^*)|_h = \frac{1}{N'}\sum_{i=1}^{N'} \mathcal{D}_*^-(M^*|x_i', z_i'), \quad \mathcal{D}_*^-(M^*|x,z) = \mathcal{D}_*^-(M|x,z)|_{\theta^*}, \tag{69}$$

where θ^{-*} is obtained from performing $\min_{\theta^-} \mathcal{D}_*^-(M)|_{h=0}$ on the noisy data D_{xz}^h, which is equivalent to performing $\min_{\theta^-} \mathcal{D}_*^-(M)$ at the given h on the original data D_{xz}. Moreover, s_j^2 is the variance of the j-th element of the original sample x_i, and can be estimated by $s_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - m_j)^2$, $m_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$.

In eq.(69), $L^-(h)$ is a part of $L(h_x)$ in eq.(67). This $L(h_x)$ takes 0 for Type (c) in eq.(65) but usually a unknown function of h for Types (a)&(b). However, even for Types (a)&(b), it follows from eq.(20) that $L(h_x)|_{h=0} = -\frac{1}{N}\sum_{i=1}^{N} p(x_i)$ is irrelevant to learning and thus can be ignored. Moreover, when $p(x)$ is uniform or $\ln p(x)$ is linear to x, it follows from eq.(67) that $H_p = 0$; when $p(x)$ is gaussian, H_p is exactly the trace of the inverse matrix of its covariance, which is thus used as an estimate of H_p in eq.(69).

Again, similar to Sec.2.3, either or both $\mathcal{D}^-(h)$ and $H(p_h)$ can be estimated in other choices:

(1) In eq.(69), we can approximately get θ^{-*} from performing $\min_{\theta^-} \mathcal{D}_*^-(M)|_{h=0}$ on the original data D_{xz}. Moreover, in eq.(67), after ignoring $L(h_x)|_{h=0}$ and $\mathcal{D}_*^-(M)|_{h=0}$ which are irrelevant to h, in eq.(24) we can also use

$$\mathcal{D}^-(h) = 0.5h_x H_p + 0.5(h_x R_D^x + h_z R_D^z), \tag{70}$$

where R_D^x, R_D^z are obtainable for some specific architectures, as will be indicated in Sec.4.3.

(2) We can also estimate $H(p_h)$ based on D_{xz} directly. Corresponding to eq.(26), we have

$$H(p_h) = -0.5(m \ln h_z + d \ln h_x) + D(h_x) + 0.5\frac{h_0 - e_0}{h_x}, \tag{71}$$

where $D(h_x)$, h_0, e_0 are given by eq.(26) at $h = h_x$.

Moreover, corresponding to eq.(26), we also have

$$J(h) = 0.5(h_z R_D^z + h_x R_D^x + h_x H_p) + H(p_h), \quad h^{new} = h^{old} - \eta[J_x', J_z']^T, \ h = [h_x, h_z]^T,$$
$$J_x' = 0.5h_x^{-2}[(R_D^x + H_p)h_x^2 - dh_x + e_0], \ J_z' = 0.5h_z^{-1}(R_D^z h_z - m). \tag{72}$$

Furthermore, we can even directly consider the roots of $J_x' = 0, J_z' = 0$, and take the one that is a minimum point of $J(h)$ as an initialization of eq.(72) or eq.(24).

4.3 Fully coordinated system versus partially coordinated system

The nature of supervised BYY learning is also featured by the structure of $p_{M_{y|x,z}}(y|x,z)$ which coordinates the learning on $p_M(z|x)$ in eq.(64) through weighting the hidden representation y according to the corresponding relation (x_i, z_i) given by the training set D_{xz}. We call a supervised BYY learning system either *partially coordinated* when $p_{M_{y|x,z}}(y|x,z)$ is parametric with a given structural constraint or *fully coordinated* when $p_{M_{y|x,z}}(y|x,z)$ is structure-free such that it is automatically decided by eq.(12).

1. Parameter learning and model selection on fully coordinated systems

• **$\mathcal{D}$-learning** For the *$\mathcal{D}$-learning*, a structure-free $p_{M_{y|x,z}}(y|x,z)$ is automatically decided by eq.(12), resulting in

$$p_{M_{y|x,z}}(y|x,z) = p_{M_{z|x,y}}(z|x,y)p_H(y|x)/p_M(z|x), \tag{73}$$

with $p_H(y|x), p_M(z|x)$ given by eq.(64). It further follows from eq.(65) that

$$\mathcal{D}_*^-(M|x,z) = -\begin{cases} \ln p_M(z|x), & \text{for Types (a)\&(b),} \\ \ln p_M(z,x), & \text{for Type (c).} \end{cases} \tag{74}$$

For a fixed h, the *$\mathcal{D}$-learning* in eq.(12) is equivalent to $\min_\theta \mathcal{D}_*^-(M)$ given in eq.(65) with the above $\mathcal{D}_*^-(M|x,z)$. Particularly, from eq.(20) we have that the empirical learning $\min_\theta \mathcal{D}_*^-(M)|_{h=0}$ is exactly the ML learning on either $p_M(z|x)$ for the cases of Types (a)&(b) or $p_M(z,x)$ for the cases of Type (c).

Moreover, we can use the following *alternative reduction* procedure for implementing $\min_\theta \mathcal{D}_*^-(M)$ at a fixed h:

Step 1: Get $p_{M_{y|x,z}}$ *by eq.*(73). (75)

Step 2: Update $\theta_{z|y,x}$ to increase $L(\theta_{z|y,x}) = \int p_h(x,z) p_{M_{y|x,z}} \ln p_{M_{z|x,y}}(z|x,y)dxdydz$, update

$$\begin{cases} \theta_{y|x} \text{ to increase} L(\theta_{y|x}) = \int p_h(x,z) p_{M_{y|x,z}} \ln p_{M_{y|x}}(y|x)dxdydz, & \text{Type (a).} \\ \theta_{x|y}, \theta_y, \text{ to increase } L(\theta_{x|y}, \theta_y) = \int p_h(x,z) p_{M_{y|x,z}} \ln p_{M^c_{y|x}}(y|x)dxdydz, & \text{Type (b).} \\ \theta_{x|y} \text{ to increase } L(\theta_{x|y}) = \int p_h(x,z) p_{M_{y|x,z}} \ln p_{M_{x|y}}(x|y)dxdydz, & \\ \text{and } \theta_y \text{ to increase } L(\theta_y) = \int p_h(x,z) p_{M_{y|x,z}} \ln p_{M_y}(y)dxdydz, & \text{Type (c).} \end{cases}$$

This procedure can also be incorporated into Sec.4.2 for implementing the data smoothing learning for a best h^*. Further inserting eq.(74) into eq.(67), we can get rather simple forms of R_D^x, R_D^z such that it can be directly used in eq.(70) and eq.(72).

After the *$\mathcal{D}$-learning*, with the obtained θ^*, h^* or a prefixed $h = 0$, we can use eq.(13) for selecting an appropriate k^* with $\mathcal{H}(M) = \mathcal{H}_*^-(M)$ given by eq.(66). Particularly, if the *$\mathcal{D}$-learning* is empirical, i.e., we can equivalently simplify it by letting

• **$\mathcal{H}$-learning** For the $\mathcal{H}$-learning, similar to eq.(10), for each fixed h it follows from eq.(12) that

$$p_{M_{y|x,z}}(y|x,z) = \delta(y - \hat{y}), \quad \hat{y} = \max_y [p_{M_{z|x,y}}(z|x,y) p_{M_{x,y}}(x,y)], \tag{76}$$

$$\mathcal{H}^-(M) = \hat{\mathcal{H}}^-(M), \quad \hat{\mathcal{H}}^-(M) = \int p_h(x,z) \ln [p_{M_{z|x,y}}(z|x,\hat{y}) p_{M_{x,y}}(x,\hat{y})]dxdz.$$

Moreover, similar to eq.(32), we can also implement this type of $\mathcal{H}$-learning by alternatively repeating:

Step 1: for each sample pair (x,z), with θ^- fixed, get $\hat{y}$ by *eq.*(76), (77)

Step 2: update $\theta_{z|y,x}$ to increase $\ln p_{M_{z|x,y}}(z|x,\hat{y})$, and update (78)

$$\begin{cases} \theta_{y|x} \text{ to increase} \ln p_{M_{y|x}}(\hat{y}|x), & \text{for Type (a),} \\ \theta_{x|y}, \theta_y, \text{ to increase } \ln p_{M^c_{y|x}}(\hat{y}|x), & \text{for Type (b),} \\ \theta_{x|y} \text{ to increase } \ln p_{M_{x|y}}(x|\hat{y}), \quad \theta_y \text{ to increase } \ln p_{M_y}(\hat{y}), & \text{for Type (c),} \end{cases}$$

where each sample pair (x,z) comes either directly from the original data set D_{xz} for empirical learning or from the noisy data set D_{xz}^h by eq.(68) for data smoothing learning. This procedure is a hard-cut version of eq.(75) since $\delta(y - \hat{y})$ is actually a Winner-take-all (WTA) version of $p_{M_{y|x,z}}$ in eq.(73). The WTA competition is made coordinately among the two parts, namely $\ln p_{M_{z|x,y}}(z|x,y)$ and $\ln p_{M_{x,y}}(x,y)$. Thus, similar to Sec.2.4, the above learning is called *supervised Coordinated Competitive Learning (CCL).*

Similar to what has been discussed in Sec.4.2, we can also incorporate the above eq.(77) for getting a best h^*. Then, we use eq.(13) for selecting an appropriate k^* with $\mathcal{H}(M) = \hat{\mathcal{H}}^-(M)$ by eq.(76).

2. Parameter learning and model selection on partially coordinated systems

When $p_{M_{y|x,z}}$ is parametric in a structure with a set $\theta_{y|x,z}$ of parameters, the issues encountered in parameter learning and model selection are basically the same as discussed in Sec.4.1 and Sec.4.2. In the implementation of eq.(12), at a fixed h, both the $\mathcal{D}$-learning and $\mathcal{H}$-learning can be made by alternatively repeating the following two steps:

Step 1: with $\theta_{z|x,y}$ $\theta_{x,y}$ fixed, update $\theta_{y|x,z}$, to reduce $C(M)$,

$$Step\ \ 2:\ \ \text{with } \theta_{y|x,z} \text{ fixed, update } \theta_{z|x,y} \text{ and } \theta_{x,y} \text{ to reduce } C(M); \tag{79}$$

where $\theta_{x,y}$ consists of all the parameters in $p_{M_{x,y}}$, and $C(M) = \mathcal{D}(M)$ for the $\mathcal{D}$-learning and $C(M) = \mathcal{H}(M)$ for the $\mathcal{H}$-learning. Particularly, for the $\mathcal{D}$-learning, $\mathcal{D}_*^-(M)$ given in eq.(65) can be directly used as $C(M)$ in the above step 1. Moreover, the above step 2 can be further detailed into the step 2 in eq.(75).

The above procedure can be further incorporated into Sec.4.2 for implementing data smoothing learning for a best h^*. Furthermore, we use eq.(13) for selecting an appropriate k^* with $\mathcal{H}(M) = \hat{\mathcal{H}}^-(M)$ by eq.(66).

Being different from the case that $p_{M_{y|x,z}}$ is structure-free, a parametric $p_{M_{y|x,z}}$ takes two roles. First, some pre-knowledge may be incorporated into the structure of $p_{M_{y|x,z}}$ which imposes a structural constraint to regularize the learning. Second, a parametric $p_{M_{y|x,z}}$ in a simple structure can facilitate the computation by implementing Step 1 in eq.(79) instead of Step 1 in eq.(75).

As shown in Xu(1999d&98), a typical structure for $p_{M_{y|x,z}}$ consists of two factors. One is the structure for implementing a deterministic regression relation $(x, z) \to y$, it is generally denoted by

$$g(x, z, W_{y|x,z}) = [\nu_1, \cdots, \nu_k]^T = \nu. \tag{80}$$

The simplest case is linear $\nu = \eta = W_{y|x,z}[x, z]^T + b$. It can be further extended into a so called post-linear function $\nu_j = s(\eta_j)$, with $s(r)$ being a nonlinear monotonic or sigmoid function. The second factor is the density form of $p_{M_{y|x,z}}$, which depends on the representation form of y. Two typical examples are given by:

$$p_{M_{y|x,z}}(y|x, z) = \begin{cases} \prod_{j=1}^k \nu_j^{y_j}(1-\nu_j)^{1-y_j}, & \text{binary } y = [y_1, \cdots, y_k]^T, \\ G(y, g(x, z, W_{y|x,z}), \Sigma_g), & \text{real } y = [y_1, \cdots, y_k]^T. \end{cases} \tag{81}$$

5 Typical Applications to Supervised Learning Tasks

5.1 Mixture-of-experts models and RBF nets

When $y = 1, \cdots, k$ and $p_{M_{z|x,y}}(z|x, y), y = 1, \cdots, k$ has a modular structure as introduced in Sec.4.1, the results introduced in Sec.4 will not only lead us to revisit existing results but also provide us a number of new results on tasks of supervised learning based on mixture-of-experts (ME) models and radial basis function (RBF) nets.

1. Revisit of old results

Item 5.1 (*The original model and alternative model of mixture of experts*) For a structure-free $p_{M_{y|x,z}}(y|x, z)$ given by eq.(73), as previously discussed after eq.(74), we known that the empirical learning $\min_\theta \mathcal{D}_*^-(M)|_{h=0}$ is exactly the ML learning on either $p_M(z|x)$ for the cases of Types (a)&(b) or $p_M(z, x)$ for the cases of Type (c). From eq.(63) and eq.(64), we see that $p_M(z|x)$ of Type (a) is exactly the original mixture-of-expert (ME) model (Jacobs, et al, 1991; Jordan & Jacobs, 1994) and that $p_M(z, x)$ of Type (c) is exactly the alternative ME model (Xu, et al, 1994&95). Moreover, with $h = 0$ from eq.(75) we are lead to the specific EM algorithms presented in Jordan & Jacobs (1994) and Xu, Jordan & Hinton (1994&95) for implementing the ML learning on the original ME model and alternative ME model, respectively. Furthermore, $p_M(z|x)$ of Type (b) provides another variant of ME model, which can be also trained by eq.(75).

2. New results

Item 5.2 (*The EM algorithm for RBF nets*) As shown in Xu(1998a), the normalized RBF net (Moody & Darken, 1989, Xu, et al, 1994) and its further extension can be regarded as two special cases of the alternative ME model. Thus, from this connection we have derived the specific EM algorithms for the ML learning on RBF nets and shown that the EM

algorithms outperform the conventional two step algorithm for RBF net learning (Moody & Darken, 1989).

Item 5.3 (*Coordinated Competitive Learning (CCL) for ME models and RBF nets*) As discussed in Sec.4.3, the procedure eq.(77) is actually a hard-cut version of eq.(75). At the special cases of the ME models and RBF nets with empirical learning (i.e., $h = 0$), eq.(77) leads us to exactly the CCL learning technique which provides adaptive EM-like algorithms for implementing on-line learning on the original ME, alternative ME and normalized RBF (NRBF) nets as well as extended NRBF, firstly presented in Xu(1998a).

Item 5.4 (*Data smoothing learning*) We can also get the above original ME, alternative ME and RBF nets to be trained by data smoothing learning with a best h^* searched as discussed in Sec.4.2, which provides a new and easy implementing regularization method for learning on these models. Moreover, as shown in Xu(1999e&1998), for updating θ^- at each fixed h, we have two choices. One is made by directly using the above mentioned EM algorithms or adaptive EM-like algorithms on the noisy data set $D^h_{x,z}$. The other is made on the original data set by modifying these algorithms slightly, with a term $h_z I_m$ added to $\Sigma_{z|x,y}$ and a term $h_x I_d$ added to $\Sigma_{x|y}$ in a way similar to the last equation in eq.(45), where $\Sigma_{z|x,y}, \Sigma_{x|y}$ are the covariance matrices of $p_{M_{z|x,y}}$ and $p_{M_{x|y}}$, respectively.

Item 5.5 (*Criteria for selecting the number of experts and of basis functions*) From eq.(13) we can obtain specific criteria for selecting k as the number of experts in ME models or the number of basis functions in RBF nets. The details are referred to Xu(1998a). For examples, we consider eq.(13) at $h = 0$, and get the following general criterion:

$$J(k) = J_1(k) + J_2(k), \quad -J_1(k) = \frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{k} p_{M_{y|x,z}}(y|x_i,z_i)\ln p_{M_{z|x,y}}(z_i|x_i,y),$$
$$-J_2(k) = \begin{cases} \frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{k} p_{M_{y|x,z}}(y|x_i,z_i)\ln p_{M_{y|x}}(y|x_i), & \text{Type (a)}, \\ \frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{k} p_{M_{y|x,z}}(y|x_i,z_i)\ln p_{M^c_{y|x}}(y|x_i), & \text{Type (b)}, \\ \frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{k} p_{M_{y|x,z}}(y|x_i,z_i)\ln p_{M_{x|y}}(x_i|y) & \\ \quad +\sum_{y=1}^{k} p_{M_y}(y)\ln p_{M_y}(y), & \text{Type (c)}; \end{cases} \tag{82}$$

where $p_{M^c_{y|x}}$ is given by eq.(28). Type (a) apples to the original ME model and Types (b)& (c) apply to the alternative ME model, NRBF nets and extend NRBF nets with Type (b) for the cases where the regression relation $p_M(z|x)$ is focused on and with Type (c) for the cases where the joint density $p_M(z,x)$ is focused on, respectively. Specifically, we have

$$J_2^{(b)}(k) = J_2^{(c)}(k) + L(k), \quad L(k) = \frac{1}{N}\sum_{i=1}^{N}\ln p_M(x_i) \tag{83}$$

where the superscripts '(b), (c)' denote Type (b) and Type (c), respectively.

Moreover, we can further get specific forms of $J_1(k), J_2(k)$ when the densities forms are given. E.g., when $p_{M_{y|x,z}}, p_{M_{x|y}}$ are gaussians with covariance matrices $\Sigma_{y|x,z}, \Sigma_{x|y}$, after ignoring some constants we can get $\alpha_y = \frac{1}{N}\sum_{i=1}^{N} p_{M_{y|x,z}}(y|x_i,z_i)$ and

$$J_1(k) = \sum_{y=1}^{k}\alpha_y \ln|\Sigma_{y|x,z}|, \quad J_2^{(c)}(k) = \sum_{y=1}^{k}\alpha_y \ln|\Sigma_{x|y}| + \sum_{y=1}^{k}\alpha_y \ln\alpha_y, \tag{84}$$

as long as N is large enough. Readers are referred to Xu(1998a) for further details.

Item 5.6 (*Other variants*) We can also get the above original ME, alternative ME and RBF nets to be trained by either the $\mathcal{D}$-learning or $\mathcal{H}$-learning with $p_{M_{y|x,z}}(y|x,z)$ constrained by a given parametric structure, as discussed at the end of Sec.4.3.

5.2 Three layer architecture: (I) with deterministic hidden layer

When $p_{M_{z|x,y}}(z|x,y) = p_{M_{z|x,y}}(z|y)$ has a cascade structure as introduced in Sec.4.1, it follows from eq.(64) that $p_M(z|x)$ is realized by an architecture of cascaded three layers. For Type (a), both the mappings $x \to y$ and $y \to z$ are directly implemented by the forward architecture $x \to y \to z$ of $p_{M_{y|x}}, p_{M_{z|y}}$, which forms a *forward three layer architecture* with one hidden layer in a general probabilistic form. For Types (b) & (c), the mapping $x \to y$

is indirectly realized by $p_{M^c_{y|x}}$, which forms a three layer architecture of $x \leftarrow y \rightarrow z$. This architecture can be used for *bi-directional associate retrieval*. After learning, we can use a sample x as a key to call its inner code y by $p_{M^c_{y|x}}$, and then recall out $\hat{x}, \hat{z}$ as associate retrieval of x by $p_{M_{z|y}}, p_{M_{x|y}}$. Similarly, we can also use a sample z as a key to call a y.

In this subsection, we focus on the forward architecture $x \rightarrow y \rightarrow z$ with deterministic hidden layer, that is, $p_{M_{y|x}}(y|x) = \delta(y - f(x, W_{y|x}))$.

1. Revisit of old results

Item 5.7 (*Three layer forward net with deterministic hidden layer and least sqaure learning*) We again consider the case given by eq.(74), obtained from a structure-free $p_{M_{y|x,z}}(y|x,z)$ that is decided by eq.(73). As discussed after eq.(74) previously, the empirical $\mathcal{D}$-learning $\min_\theta \mathcal{D}^-_*(M)|_{h=0}$ is exactly the ML learning on $p_M(z|x)$ for Type (a), because of

$$\mathcal{D}^-_*(M)|_{h=0} = -\frac{1}{N}\sum_{i=1}^N \ln p_M(z_i|x_i), \tag{85}$$

When $p_{M_{y|x}}(y|x) = \delta(y - f(x, W_{y|x}))$, from eq.(64) we get $p_M(z|x) = p_{M_{z|y}}(z|f(x, W_{y|x}))$. So, we are lead to the ML learning on a three layer forward net with a deterministic hidden layer. Furthermore, when

$$p_{M_{z|y}} = G(z, g(y, W_{z|y}), \sigma^2 I_m),\ f(x, W_{y|x}) = [\mu_1, \cdots, \mu_k]^T,\ \mu_j = s(e_j^T(W_{y|x}x) + a_j), \tag{86}$$

where m is the dimension of z, $0 \le s(r) \le 1$ is sigmoid function, e.g, $s(r) = (1+e^{-r})^{-1}$, and $e_j^T x$ producing the j-th element of x. In this case, when N is large enough, after ignoring some constants, we find that $\min_\theta \mathcal{D}^-_*(M)|_{h=0}$ is asymptotically equivalent to the minimization of

$$\mathcal{D}^-(E_2) = 0.5m \ln E_2,\quad E_2 = \sum_{i=1}^N \|z_i - g(f(x_i, W_{y|x}), W_{z|y})\|^2, \tag{87}$$

with respective to $W_{y|x}, W_{z|y}$. That is, it is equivalent to the least square learning on the conventional three layer net, which is usually trained by the back-propagation technique.

More interestingly, we can also get a number of new results on three layer net as follows.

2. New results

Item 5.8 (*$\mathcal{H}$-learning as a new regularization*) We still consider the above special case. When $p_{M_{y|x}}(y|x) = \delta(y - f(x, W_{y|x}))$, in help of a technique developed in Sec.4.2 of Xu(1999a) for handling $\int p_{M_{y|x,z}} \ln p_{M_{y|x,z}} dy = \int \delta(y - f(x_i, W_{y|x})) \ln \delta(y - f(x_i, W_{y|x}))dy$, it follows from eq.(76) that $p_{M_{y|x,z}}(y|x,z) = \delta(y-\hat{y}) = \delta(y - f(x, W_{y|x}))$ and $\mathcal{H}^-(M)|_{h=0} = -\mathcal{D}^-_*(M)|_{h=0} + r(M)$, with $r(M) = \frac{1}{N}\sum_{i=1}^N \int \delta(y - f(x_i, W_{y|x}) \ln \delta(y - f(x_i, W_{y|x})dy = -0.5\frac{1}{N}\sum_{i=1}^N \ln |D_f(x_i)D_f(x_i)^T| + C$, $D_f(x) = \partial f(x, W_{y|x})/\partial x^T$, where C is irrelevant to $W_{y|x}$ and thus ignored. Therefore, it also follows from eq.(87) that the empirical $\mathcal{H}$-learning is equivalent to

$$\min_{W_{y|x}, W_{z|y}} -\mathcal{H}^-_0(M), \tag{88}$$
$$-\mathcal{H}^-_0(M) = m \ln E_2 + \frac{1}{N}\sum_{i=1}^N \ln |D_f(x_i)D_f(x_i)^T|,\quad D_f(x) = \partial f(x, W_{y|x})/\partial x^T,$$

where the second term acts as a regularization term.

Item 5.7 (*A new criterion for selecting hidden unit number*) We keep to consider the above special case. Not only the above empirical $\mathcal{H}$-learning leads to $p_{M_{y|x,z}}(y|x,z) = \delta(y - f(x, W_{y|x}))$ and $\mathcal{H}^-_0(M)$ in eq.(88), but also the same result is obtained from the empirical $\mathcal{D}$-learning by eq.(73) with $p_{M_{y|x,z}}(y|x,z) = p_{M_{z|y}}(z|y)p_{M_{y|x}}(y|x)/p_{M_{z|y}}(z|f(x, W_{y|x}))$, which is put into $\mathcal{H}^-(M)|_{h=0}$ to get the same $\mathcal{H}^-_0(M)$ in eq.(88). Thus, after either the empirical $\mathcal{H}$-learning or the empirical $\mathcal{D}$-learning, from eq.(13) we can select the number k of hidden units by $\min_k J(k)$ with

$$J(k) = m \ln \sum_{i=1}^N \|z_i - g(f(x_i, W_{y|x}), W_{z|y})\|^2 + \frac{1}{N}\sum_{i=1}^N \ln |D_f(x_i)D_f(x_i)^T|. \tag{89}$$

Item 5.8 (*Data smoothing learning and hidden unit number selection*) We can also implement the *data smoothing learning* via searching a best h^* as discussed in Sec.4.2, which acts as a new type of regularization. In a simplest way, we update θ^- at each fixed h by implementing the least sqaure learning or eq.(88) on the noisy data set $D^h_{x,z}$. Then, we select the number k of hidden units by eq.(89) with $J(k)$ calculated on the noist data set $D^{h^*}_{x,z}$.

5.3 Three layer architecture: (II) with probabilistic hidden layer

When $p_{M_{y|x}}(y|x)$ is not a δ-density, i.e., the hidden layer is probabilistic, the ML learning on $p_M(z|x)$ is no longer equivalent to the least square learning on E_2. Actually, we have to encounter the integral over y in eq.(64) for getting $p_M(z|x)$ and thus its regression function $E(z|x)$. To overcome this difficulty, we propose a specific approximation technique here.

We consider $\int p(u)T(u)du$ by the Taylor expansion of $T(u)$ around the mean $\hat{u} = \int up(u)du$:

$$T(u) \approx T(\hat{u}) + (u-\hat{u})^T G(\hat{u}) + c_T(u-\hat{u})^T H(\hat{u})(u-\hat{u}),$$
$$c_T = \begin{cases} 0, & \text{1st order expansion only,} \\ 0.5, & \text{2nd order expansion,} \end{cases} \quad (90)$$

where $G(u), H(u)$ are the gradient and the Hessian matrix of $T(u)$, respectively. Since $\int p(u)(u-\hat{u})du = 0$, it follows that

$$\int p(u)T(u)du \approx T(\hat{u}) + c_T Tr[\Sigma_u H(\hat{u})], \text{ where } \Sigma_u \text{ is the covariance matrix of } p(u). \quad (91)$$

Moreover, provide that $\int y p_{M_{y|x}}(y|x)dy = f(x, W_{y|x})$ and $p_{M_{z|y}}(z|y)$ has a diagonal covariance matrix $\Sigma = diag[\lambda_1, \cdots, \lambda_k]$, we consider $p_M(z|x)$ of Type (a) in eq.(64). Regarding $p_{M_{z|y}}(z|y)$ as $T(u)$ and $p_{M_{y|x}}(y|x)$ as $p(u)$, from eq.(91) we have

$$p_M(z|x) \approx p_{M_{z|y}}(z|f(x, W_{y|x})) + c_T R_T(z, f(x, W_{y|x})), \; R_T(z,y) = \sum_{j=1}^k \lambda_j \frac{\partial^2 p_{M_{z|y}}}{\partial^2 y_j}, \quad (92)$$

where H_y is the Hessian of $p_{M_{z|y}}$ with respect to y.

When $p_{M_{y|x,z}}(y|x,z) = p(y|x,z)$ is structure-free, from eq.(73) and eq.(92) we further have

$$p(y|x,z) = p_{M_{z|y}}(z|y)p_{M_{y|x}}(y|x)/[p_{M_{z|y}}(z|f(x, W_{y|x})) + c_T R_T(z, f(x, W_{y|x}))]. \quad (93)$$

Putting it into p_{M_1} in eq.(60), from eq.(66) we get $\mathcal{H}^-_*(M)|_{h=0} = \frac{1}{N}\sum_{i=1}^N \int p_{M_{y|x}}(y|x_i)\mathcal{T}(y|x_i, z_i)dy$ with

$$\mathcal{T}(y|x_i, z_i) = \frac{p_{M_{z|y}}(z_i|y)}{p_{M_{z|y}}(z_i|f(x_i, W_{y|x})) + c_T R_T(z_i, f(x_i, W_{y|x}))} \ln [p_{M_{z|x,y}}(z_i|y)p_{M_{y|x}}(y|x_i)]. \quad (94)$$

If we regard $p_{M_{y|x}}(y|x_i)$ as $p(u)$ and $\mathcal{T}(y|x_i, z_i)$ as $T(u)$ and consider the 1st order approximation in eq.(91), we get

$$\mathcal{H}^-_*(M)|_{h=0} \approx \frac{1}{N}\sum_{i=1}^N \mathcal{T}(y|x_i, z_i)|_{y=f(x_i, W_{y|x})}. \quad (95)$$

In the sequel, we get a number of results from eq.(92), eq.(93), eq.(94) and eq.(95).

1. Revisit of old results

Item 5.9 (*The least sqaure learning*) When $p_{M_{y|x,z}}(y|x,z) = p(y|x,z)$ is structure-free and thus given by eq.(73), if we only consider the 1st order expansion in eq.(92), we again

get $p_M(z|x) = p_{M_{z|y}}(z|f(x, W_{y|x}))$. So, in the case of eq.(86) the empirical $\mathcal{D}$-learning, i.e., $\min_\theta \mathcal{D}_*^-(M)|_{h=0}$, leads to eq.(87) again. In other words, the least sqaure learning on E_2 for a three layer forward net with deterministic hidden layer can be regarded as an 1st order approximation of the ML learning on $p_M(z|x)$ given by eq.(64) for a three layer forward net with probabilistic hidden layer.

2. New results

Item 5.10 (*A criterion for selecting probabilistic hidden unit number*) From eq.(13) and eq.(9), after empirical parameter learning we can select the number k of hidden units by $J(k) = -\mathcal{H}_*^-(M)|_{h=0}$ given by eq.(95). Particularly, when we consider the 1st order case in $\mathcal{T}(y|x_i, z_i)$, i.e., $c_T = 0$, we get

$$J(k) = -\frac{1}{N}\sum_{i=1}^N \ln p_{M_{z|x,y}}(z_i|f(x_i, W_{y|x})) - \frac{1}{N}\sum_{i=1}^N \ln p_{M_{y|x}}(f(x_i, W_{y|x})|x_i). \quad (96)$$

Specifically, in the cases of eq.(86) with $p_{M_{y|x}}$ being a generalized Benoulli distribution, i.e.,

$$p_{M_{y|x}}(y|x) = \prod_{j=1}^k \mu_j(x)^{y_j}(1-\mu_j(x))^{1-y_j}, \quad [\mu_1(x), \cdots, \mu_k(x)]^T = f(x, W_{y|x}), \quad (97)$$

after ignoring some irrelevant terms we get that eq.(96) equivalently becomes

$$\begin{aligned} J(k) &= 0.5m \ln \sum_{i=1}^N \|z_i - g(f(x_i, W_{y|x}), W_{z|y})\|^2 \\ &-\frac{1}{N}\sum_{i=1}^N\sum_{j=1}^k [\mu_{j,i} \ln \mu_{j,i} + (1-\mu_{j,i}) \ln (1-\mu_{j,i})], \end{aligned} \quad (98)$$

where $\mu_{j,i} = \mu_j(x_i)$. It is firstly obtained in Xu(1998a&b) via a so called mean field approximation. Here, we have justified this previous heuristics through the technique eq.(91).

Item 5.11 (*Coordinated Competitive Learning (CCL) and hidden unit selection*) We consider the empirical $\mathcal{H}$-learning with a structure-free $p_{M_{y|x,z}}(y|x, z)$. With $h = 0$, from eq.(76) and eq.(77) we get a CCL based learning algorithm for adaptively training a three layer net with probabilistic hidden layer as follows:

$$\begin{aligned} &\textit{Step}\ \ 1: \ \text{for each sample pair } (x_i, z_i), \ \text{get } \hat{y}_i = \max_y [p_{M_{z|y}}(z_i|y) p_{M_{y|x}}(y|x_i)], \\ &\textit{Step}\ \ 2: \ \text{update } \theta_{z|y} \text{ to increase } \ln p_{M_{z|y}}(z_i|\hat{y}), \text{and update } \theta_{y|x} \text{ to increase} \\ &\qquad \ln p_{M_{y|x}}(\hat{y}|x_i), \text{ e.g., by gradient ascent.} \end{aligned} \quad (99)$$

That is, we solve the so called credit-assignment task via a coordinated competition to get a winner code $\hat{y}_i$ for each x_i, and thus separately train the layer $x \to y$ by the pair $(x_i, \hat{y}_i)$ and the layer $y \to z$ by the pair $(\hat{y}_i, z_i)$.

After parameter learning, with learned parameter $\theta_{z|y}^*, \theta_{y|x}^*$, it follows from eq.(76) that $\mathcal{H}^-(M)|_{h=0} = \frac{1}{N}\sum_{i=1}^N \ln [p_{M_{z|y}}(z_i|\hat{y}_i) p_{M_{y|x}}(\hat{y}_i|x_i)]$. Thus, from eq.(13) and eq.(9) we have

$$J(k) = -\frac{1}{N}\sum_{i=1}^N \ln [p_{M_{z|y}}(z_i|\hat{y}_i) p_{M_{y|x}}(\hat{y}_i|x_i)], \quad \hat{y}_i = \max_y [p_{M_{z|y}}(z_i|y) p_{M_{y|x}}(y|x_i)], \quad (100)$$

for selecting hidden unit number. Specifically, in the cases of eqs.(86)&(97), we have

$$\begin{aligned} J(k) = \quad & 0.5m \ln \sum_{i=1}^N \|z_i - g(\hat{y}_i, W_{z|y})\|^2 \\ & -\frac{1}{N}\sum_{i=1}^N\sum_{j=1}^k [\hat{y}_i \ln \mu_j(x_i) + (1-\hat{y}_i) \ln (1-\mu_j(x_i))]. \end{aligned} \quad (101)$$

Item 5.12 (*ML learning and hidden unit selection via second order approximation*) In the sense of the 2nd order approximation in eq.(92) (i.e., $c_T = 0.5$), both the least sqaure learning on E_2 and the ML learning on $p_M(z|x) = p_{M_{z|y}}(z|f(x, W_{y|x}))$ are no longer equivalent to the ML learning on $p_M(z|x)$ since there is an additional term $R_T(z, \hat{y}(x))$ that penalizes the uncertainty due to the layer $x \to y$ and the layer $y \to x$ jointly. Thus, we can regard that this ML learning on $p_M(z|x)$ actually introduces another type of regularization.

We can implement this learning either directly by maximizing $p_M(z|x)$ or indirectly via the following adaptive EM-like algorithm in help of random sampling:

$$\begin{aligned} &Step\ 1: \text{ For each pair } (x_i, z_i), \text{ randomly sampling a set } \{y_i^{(j)}\}_{j=1}^{n_i}, \\ &\quad \text{then get } p(y_i^{(j)}|x_i, z_i) \text{ by eq.(93);} \\ &Step\ 2: \text{ for } j = 1, \cdots, n_i, \text{ with a stepsize } \eta, \text{ update} \\ &\theta_{z|y}^{new} = \theta_{z|y}^{old} + \eta p(y_i^{(j)}|x_i, z_i) \frac{\partial \ln p_{M_{z|y}}(z_i|y_i^{(j)})}{\partial \theta_{z|y}}, \\ &\theta_{y|x}^{new} = \theta_{y|x}^{old} + \eta p(y_i^{(j)}|x_i, z_i) \frac{\partial \ln p_{M_{y|x}}(y_i^{(j)}|x_i)}{\partial \theta_{y|x}}. \end{aligned} \tag{102}$$

From $-\mathcal{H}(M) = \mathcal{D}(p_{M_1}||p_{M_2}) + H_e$ with H_e given by eq.(66), eq.(13) at $h = 0$ can also be estimated in help of random sampling as follows:

$$J(k) = -\sum_{i=1}^{N} \ln p_M(z_i|x_i) + \sum_{i=1}^{N} \sum_{j=1}^{n_i} p(y_i^{(j)}|x_i, z_i) \ln p(y_i^{(j)}|x_i, z_i), \tag{103}$$

with $p_M(z|x)$ given by eq.(92) and $p(y|x, z)$ by eq.(93), where each $\{y_i^{(j)}\}_{j=1}^{n_i}$ is generated for each pair x_i, z_i.

Item 5.13 (*Partially coordinated system*) As discussed at the end of Sec. 4.3, when $p_{M_{y|x,z}}$ is parametric in a structure, the parameter learning can still be made by eq.(79) from which we get specific EM-like algorithms for various sepcific designs of $p_{M_{y|x,z}}$ such as given by eq.(80) and eq.(81). We can also get criteria for selecting hidden unit number from eq.(13).

Item 5.14 (*Data smoothing learning*) In the cases of a size of training samples, we can further improve all the above discussed cases by incorporating in the *data smoothing learning* via searching a best h^* as discussed in Sec.4.2.

6 Conclusion

BYY learning system and theory provides a unified statistical learning framework for parameter estimation, data smoothing based regularization and model selection on various tasks of unsupervised learning, supervised learning, and temporal modeling. Under this general framework, we can systematically obtain not only those existing major learning approaches with new insights, but also a series new findings. Many topics given in this paper deserve to be further explored through both experiments and mathematical analyses.

References

[1] Amari, S, (1985), *Differential geometry methods in statistics*, Lecture Notes in Statistics 28, Springer.

[2] Amari, S, (1995), " Information geometry of the EM and em algorithms for neural networks", *Neural Networks 8*, No.9, 1379-1408.

[3] Amari, S, et al (1996), "A new learning algorithm for blind separation of sources", in Touretzky, D.S., et al, eds, *Advances in Neural Information Processing 8* MIT Press, 757-763.

[4] Anderson, T.W., and Rubin, H.(1956), " Statistical inference in factor analysis", *Proc. Berkeley Symp. Math. Statist. Prob. 3rd 5*, pp111-150, UC Berkeley, 1956.

[5] Bell, A.J., & Sejnowski, T.J. (1995), "An information-maximization approach to blind separation and blind de-convolution", *Neural Computation* **7**, 1129-1159.

[6] Comon, P. (1994), "Independent component analysis - a new concept?", *Signal Processing,* **36**, 287-314.

[7] Csiszar, I.,& G. Tusnady (1984), "Information geometry and alternating minimization procedures", *Statistics and Decisions*, Supplementary Issue, No.1, 205-237.

[8] Carpenter, G.A. & Grossberg, S., (1987), "A massively parallel architecture for a self-organizing neural pattern recognition machine", *Computer Vision, Graphics, and Image Processing*, Vol.37, 54-115.

[9] Dayan, P., & Zemel, R.S. (1995), "Competition and multiple cause models", *Neural Computation 7*, pp565-579.

[10] Dayan,P., et al (1995), "The Helmholtz machine", *Neural Computation 7*, No.5, 889-904.

[11] Dayan, P. & Hinton, G., E., 1996, "Varieties of Helmholtz machine", *Neural Networks 9*, No.8, 1385-1403.

[12] Dempster, A.P., et al (1977), "Maximum- likelihood from incomplete data via the EM algorithm", *J. of Royal Statistical Society, B39*, 1-38.

[13] Devroye, L. et al (1996), *A Probability Theory of Pattern Recognition*, Springer, 1996.

[14] Girosi, F, Jones, M, & Poggio, T. (1995), "Regularization theory and neural architectures", *Neural Computation 7*, 219-269.

[15] Grossberg, S.(1976), "Adaptive pattern classification and universal recording: I. parallel development and coding of neural feature detectors; II. feedback, expectation, olfaction, illusions", Biological Cybernetics, **23**, 121-134& 187-202.

[16] Hinton, G. E., et al (1995), "The wake-sleep algorithm for unsupervised learning neural networks", *Science 268*, pp1158-1160.

[17] Jacobs, R.A., et al (1991), "Adaptive mixtures of local experts", *Neural Computation, 3*, 79-87.

[18] Karhunen, J. & Joutsensalo, J. (1994), "Representation and separation of signals using nonlinear PCA type Learning," *Neural Networks 7*, pp113-127.

[19] Mackay, D.J.C., (1992), "A practical Bayesian framework for back-propagation networks", *Neural Computation 4*, 448-472.

[20] Moody, J., & Darken, J. (1989), "Fast learning in networks of locally-tuned processing units", *Neural Computation 1*, 1989, pp281-294.

[21] Saund, E.(1995), "A multiple cause mixture model for unsupervised learning", *Neural Computation*, Vol.7, pp51-71.

[22] Xu, L.(1999a), "Temporal BYY Learning for State Space Approach, Hidden Markov Model and Blind Source Separation ", in press, *IEEE Trans. Signal processing*.

[23] Xu, L.(1999b), "Temporal Bayesian Ying-Yang Dependence Reduction, Blind Source Separation and Principal Independent Components ", and " Temporal Bayesian Ying-Yang Learning and Its Applications to Extended Kalman filtering, Hidden Markov model, and Sensor-Motor Integration", to appear on *Proc. IJCNN99*, Washington DC, July, 1999.

[24] Xu, L.(1999c), "Bayesian Ying-Yang theory for empirical learning, regularization and model selection: general formulation ", and " Data mining, unsupervised learning and Bayesian Ying-Yang theory ", to appear on *Proc. IJCNN99*, Washington DC, July, 1999.

[25] Xu, L.(1999d&98), "Bayesian Ying-Yang Supervised Learning, Modular Models, and Three Layer Nets", to appear on *Proc. IJCNN99*, Washington DC, July, 1999. A part of its preliminary version on Proc. ICONIP'98-Kitakyushu, Oct. 21-23, 1998, Japan, Vol.2, 631-634.

[26] Xu, L.(1999e&98), " BYY Data Smoothing Based Learning on A Size of Samples", to appear on *Proc. IJCNN99*, Washington DC, July, 1999. A part of its preliminary version on Proc. ICONIP'98-Kitakyushu, Oct. 21-23, 1998, Japan, Vol.1, 243-248.

[27] Xu, L.(1998a), "RBF Nets, Mixture Experts, and Bayesian Ying-Yang Learning", *Neurocomputing 19*, No.1-3, 223-257.

[28] Xu, L.(1998b), "Bayesian Kullback Ying-Yang Dependence Reduction Theory ", *Neurocomputing 22*, No.1-3, 81-112.

[29] Xu, L.(1998c), "Bayesian Ying-Yang Dimension Reduction and Determination", *Journal of Computational Intelligence in Finance*, a special issue on Complexity and Dimensionality Reduction in Finance, Vol.6, No.5, 6-18.

[30] Xu, L., Cheung, C.C., & Amari, S.-I. (1998f), "Learned Parametric Mixture Based ICA Algorithm ", *Neurocomputing*, Vol.22, No.1-3, pp69-80, 1998.

[31] Xu, L.(1997a), "Bayesian Ying-Yang Machine, Clustering and Number of Clusters", *Pattern Recognition Letters*, Vol.18, No.11-13, pp 1167-1178, 1997.

[32] Xu, L.(1997b), "New Advances on Bayesian Ying-Yang Learning System With Kullback and Non-Kullback Separation Functionals", *Proc. of 1997 IEEE-INNS IJCNN97*, Vol. III, pp1942-1947.

[33] Xu, L.(1997c), "Bayesian Ying-Yang Learning Based ICA Models", *Neural Networks for Signal Processing VII: Proc. 1997 IEEE Signal Processing Society Workshop*, 24-26 Sept., Florida, pp476-485.

[34] Xu, L. (1996), "A Maximum balanced mapping certainty principle for pattern recognition and associative mapping" *Proc. 1996 World Congress on neural Networks*, San Diego, Sept., 1996, 946-949.

[35] Xu, L. & Jordan, M.I. (1996), "On convergence properties of the EM algorithm for gaussian mixtures," *Neural Computation 8*, No.1, 129-151.

[36] Xu, L.(1995&96), "A Unified Learning Scheme: Bayesian-Kullback YING-YANG Machine", *Advances in Neural Information Processing Systems 8*, eds., D. S. Touretzky, et al, MIT Press, 1996, 444-450. A part of its preliminary version on *Proc. ICONIP95, Peking*, Oct 30 - Nov. 3, 1995, 977-988.

[37] Xu, L. (1995), "Advances on three streams of PCA studies" *Proc. 1995 IEEE Intl Conf. on Neural Networks and Signal Processing*, Nanjing, 1995, 480-483.

[38] Xu, L., Jordan, M.I., & Hinton, G.E., (1994&95) "An Alternative Model for Mixtures of Experts", *Advances in Neural Information Processing Systems 7*, eds., Cowan, J.D., et al, MIT Press, 1996, 633-640. Its preliminary version on *Proc. of WCNN'94*, San Diego, Vol. 2, 1994, 405-410.

[39] Xu, L., Krzyzak, A., & Yuille, A.L. (1994), "On Radial Basis Function Nets and Kernel Regression: Statistical Consistency, Convergence Rates and Receptive Field Size", *Neural Networks, 7* pp609-628.

[40] Xu, L. (1991&93),"Least mean square error reconstruction for self-organizing neural-nets", *Neural Networks 6*, 627-648, 1993. Its early version on *Proc. IJCNN91'Singapore*, 2363-2373, 1991.