International Journal of Neural Systems, Vol. 11, No. 1 (2001) 43–69

Best Harmony, Unified RPCL and Automated Model Selection for Unsupervised and Supervised Learning on Gaussian Mixtures, Three-Layer Nets and ME-RBF-SVM Models *

Lei Xu
Department of Computer Science and Engineering, Chinese University of Hong Kong
Shatin, NT, Hong Kong, P.R. China, E-mail: lxu@cse.cuhk.edu.hk

After introducing the fundamentals of BYY system and harmony learning, which has been developed in past several years as a unified statistical framework for parameter learning, regularization and model selection, we systematically discuss this BYY harmony learning on systems with discrete inner-representations. **First**, we shown that one special case leads to unsupervised learning on Gaussian mixture. We show how harmony learning not only leads us to the EM algorithm for maximum likelihood (ML) learning and the corresponding extended KMEAN algorithms for Mahalanobis clustering with criteria for selecting the number of Gaussians or clusters, but also provides us two new regularization techniques and a unified scheme that includes the previous rival penalized competitive learning (RPCL) as well as its various variants and extensions that performs model selection automatically during parameter learning. Moreover, as a by-product, we also get a new approach for determining a set of 'supporting vectors' for Parzen window density estimation. **Second**, we shown that other special cases lead to three typical supervised learning models with several new results. On three layer net, we get (i) a new regularized ML learning, (ii) a new criterion for selecting the number of hidden units, and (iii) a family of EM-like algorithms that combines harmony learning with new techniques of regularization. On the original and alternative models of mixture-of-expert (ME) as well as radial basis function (RBF) nets, we get not only a new type of criteria for selecting the number of experts or basis functions but also a new type of the EM-like algorithms that combines regularization techniques and RPCL learning for parameter learning with either least complexity nature on the original ME model or automated model selection on the alternative ME model and RBF nets. Moreover, all the results for the alternative ME model are also applied to other two popular nonparametric statistical approaches, namely kernel regression and supporting vector machine. Particularly, not only we get an easily implemented approach for determining the smoothing parameter in kernel regression, but also we get an alternative approach for deciding the set of supporting vectors in supporting vector machine.

1. Introduction

1.1. *Bayesian Ying-Yang learning*

Bayesian Ying-Yang (BYY) learning was proposed as a unified statistical learning framework firstly in 1995 [85] and systematically developed in past years. The obtained results can be summarized from both the perspective of general learning framework and the perspective of specific learning paradigms. From the first perspective, BYY learning consists of a general BYY system and a fundamental harmony learning principle as a unified guide for developing two new regularization techniques, a new type of criteria for model selection, and a new family of algorithms that perform parameter learning with least complexity nature or even automated model selection. These

*The work was fully supported by a grant from the Research Grant Council of the Hong Kong SAR (Project No: CUHK 4383/99E).

issues will be briefly introduced in Sec.2. The details are referred to some recent reviews [68,69,70,71].

From the second perspective, BYY learning with specific structure designs forms three major paradigms. First, the BYY unsupervised learning provides new results on several existing major unsupervised learning tasks, including clustering, gaussian mixture density estimation, principal component analysis (PCA), independent component analysis (ICA), independent factor analyses, LMSER self-organization, and Helmholtz machine learning, etc. The details of these results are distributed in several papers [79,80,76,75,77] and preliminary reviews can be found in two conference papers [72,71]. Readers are further referred to an up-coming systematic review [68]. Second, the BYY supervised learning provides not only new understanding on three major supervised learning models, namely three layer forward net with back-propagation learning [57], mixture expert (ME) model [30,34] and its alternative model [86] as well as normalized radial basis function (RBF) nets and its extensions [74], but also new adaptive learning algorithms and new criteria for deciding the number of hidden units, of experts and of basis functions. The details are distributed in the papers [72,73,74]. Third, the temporal BYY learning [69] acts as a general state space approach for modeling data that has temporal relationship among samples, which provides not only a unified point of view on Kalman filter, Hidden Markov model (HMM), ICA and blind source separation (BSS) with extensions, but also several new results such as higher order HMM, independent HMM for binary BSS, temporal ICA and temporal factor analysis for noisy real BSS, with adaptive algorithms for implementation and criteria for selecting the number of states or sources [69].

This paper provides systematic review and a number of new advances on a major class of BYY systems. This class is featured by using discrete inner-representations. Specifically, we consider four typical examples, including Gaussian mixture for unsupervised learning and three major supervised learning models, namely, three layer net, the original and alternative ME model, and RBF nets.

1.2. *Clustering and gaussian mixture*

A major unsupervised learning task is clustering [3,64,48,21,15,31] or vector quantization [43,50] that uses a number of vectors to represent a data set such that each vector locates at each cluster center. Such a task is equivalently tackled under the name of competitive learning in the literature of neural networks [61,26,19,1]. The existing algorithms for clustering can be grouped in two types. One is usually called incremental/hierarchical/dynamic clustering or learning [21,15,31,26]. The key point is incrementally adding one cluster center once a newly coming sample is regarded to be far beyond a threshold of a pre-specified distance measure. This type is easy to implement and the number of clusters is decided dynamically. However, the results highly depend on the initialization and the specific way that the clusters grows up. Thus, this type usually produces unsatisfactory results on complicated data that is non-homogeneous in center locations and cluster configurations. The other type of clustering algorithms considers all the possible cluster centers in parallel via minimizing a cost or maximizing a criterion. A typical example is the KMEAN algorithm [21,15] that minimizes the mean square error. However, the complexity of finding the global minimum grows exponentially with the number of clusters, and thus the problem is usually tackled by a heuristic algorithm such as KMEAN, which usually produces a solution at local minimum that may also be a bad solution especially when the algorithm starts at a bad initialization.

Alternatively, the clustering problem is solved by competitive learning (CL) which has also two types that share the above discussed features correspondingly. A typical example that performs incremental clustering is the ART learning [26]. A typical example that performs clustering in parallel is the classic CL [61]. With a bad initialization, similar to KMEAN, one serious problem of the classic CL is the so called "dead unit" problem. That is, only one or few center vectors (or so called units) are finally located among samples of data, while other units locate far away from data and thus are 'dead'. This problem is solved by introducing the so called 'conscience' mechanism [27,19,1], e.g., by the so called frequency sensitive CL (FSCL) [1]. Also, another critical problem is encountered by clustering in parallel via using KMEAN or CL. That is, it works well only when a correct number k of clusters or units are pre-given. If we do not know this number and set k inappropriately, we can get a very poor clustering performance [89]. One possible solution to the problem is to choose a best k^* by a selection criterion. Many heuristic criteria have

been proposed in the statistic literature [22,47,48,64,60]. However, this way suffers a large computational cost since we need to make clustering at a number of different value of k, even though such a process can be organized in a more efficient way, e.g., embedding the checking of the value of a selection measure as was done during clustering by the ISODATA algorithm [3].

First proposed in 1992 [89], the so called rival penalized CL (RPCL) solves this problem with the correct number k^* determined automatically during learning for the initial k large enough, in the sense that extra units are driven far away from data due to appropriately penalized learning on the rival. Similar to KMEAN and FSCL, the original RPCL works well on spherical clusters with each cluster sharing a same or similar portion of samples. Later, it has been extended to cover the cases of any elliptic shapes and in any portion of samples [84,83,76]. Moreover, RPCL has also been adopted to various applications, including information retrieval in image databases[39,35,37], plant diagnosis[23], nonlinear prediction model and hidden Markov model[13,12], clustering, vector quantization, object classification in 3-D images, scene segmentation in 2D&3D image as well as multidimensional data [7,40,14,44]. Also, following the initial suggestion[89] for training RBF net, a number of authors have used or recommended RPCL algorithm for the training of various RBF nets[4,6,91,63,11,38,7]. However, there still remains a key problem to be solved for RPCL. That is, RPCL is proposed heuristically without a theory to justify it yet. Especially, a theory is needed to appropriately control the learning rate on the winner and the de-learning rate on the rival, instead of being given by some empirical range.

Proposed heuristically as early as in the 60's [21] and later re-interpreted under the incomplete data theory [20], Gaussian mixture with the EM algorithm provides a good general solution for clustering in parallel [20,58,9,46]. Not only has the EM algorithm several advantages that considerably improve the problems of initialization and local minimum [82], but also each gaussian in general can describe each cluster by a gaussian density in any elliptic shape and in any portion of samples [80]. However, as in KMEAN and FSCL, it still needs that the correct number k of gaussians is pre-given. After ML learning by the EM algorithm, as suggested in a previous paper [81], we may get some gaussians that own a very little portion of samples for a given k large enough, and we may discard those gaussians. However, such a way works occasionally only. In a previous paper [80], not only the relation of BYY learning to ML learning is set up, with the hard-cut EM algorithm obtained as extensions of KMEAN for clustering on clusters in any elliptic shape and in any portion of samples, but also a new class of criteria is obtained from the harmony learning principle for selecting a correct k. In Sec.3 of this paper, it is further shown that BYY harmony learning provides a general framework that combines the hard-cut EM algorithm, two new regularization techniques and an automatic mechanism that decides the number of gaussians during learning. Moreover, not only the RPCL learning[89] is shown to be regarded as a special case of the framework with a theoretical guide for appropriately controlling the learning rate on the winner and the de-learning rate on the rival, but also the framework is shown to provide various RPCL variants and extensions. Furthermore, as a by-product, we also get a new approach for determining a set of 'supporting vectors' for Parzen window density estimation.

1.3. *Three typical supervised learning models*

Supervised learning tasks can be summarized by the key job that implements a desired input-output mapping, based on a given set of input-output sample pairs. The most popular example is probably three layer net with learning by back-propagation [57]. One criticism of this model is that it is a black-box with its hidden units difficult to be observed and understood. Alternatively, the radial basis function (RBF) net is suggested for solving the problems, with the feature that the entire input-output mapping is cooperatively implemented by many local input-output mapping. The existing RBF nets can be classified into two types. One is non-normalized RBF nets, which have a clear interpretation by Tikhonov-type regularization theory [66,53,10,24,52]. The other is the normalized RBF nets [49,51,32], which have a close relation to the kernel regression method [16,17,18,25,54] in the literature of nonparametric statistics [88]. The parameter learning on RBF nets is made usually in two sequential steps. The first step decides the centers of basis functions usually via certain clustering algorithm, and the second step determines the parameters of the output layers by ML or least square

learning. Such a two-step algorithm actually provides a suboptimal solution.

The third typical model is the mixture-of-expert (ME) models [30,33,34] which implements the entire input-output mapping by a number of local experts that combined via a probabilistic controlling of a so called gating net, with each individual expert being a three layer net alone. The EM algorithms can be used to separate the learning of experts and the gating nets. Then, each of them can be further learned by back-propagation. Later, an alternative gating model is proposed such that the entire learning can be made in the sense of ML by the EM algorithm in the case that each expert is described by a gaussian with a linear regression [86,55,74]. Moreover, the hard-cut EM algorithm and adaptive EM algorithm have been proposed for fast learning of both the original and alternative ME models in help of the so called coordinated competition [74].

For supervised learning models trained on a training set of finite number of samples, regularization and model selection are two major issues for improving their generalization ability. The key point is the number of hidden units, or equivalently the number of individual experts or basis functions. Among the above three typical models, the generalization performance of three layer net is prone to the number of hidden units. Therefore, many existing studies on regularization and model selection focus on three layer net. On one hand, several theories and many heuristic techniques for regularization have been proposed to force the free parameters in networks with constraints such that it is effectively equivalent to a reduced number of hidden units. Typical systematic theories for this purpose include Tikhonov-type regularization theory [66,24,5], Bayesian theory[41,42] and MDL theory[59,60], which are actually closely related to each other. On the other hand, model selection explicitly selects a structure with the best number of hidden units, in help of enumerating a number of structures with different number of hidden units. Typical model selection theories include Bayesian theory [41,42], MDL[59,60,28], VC dimension based generalization theory[67], cross validation[65,62] and AIC [2] as well as its extensions AICB, CAIC, SIC [9,8].

Though the generalization performance of the original and alternative ME models as well as RBF nets is less sensitive to the number of experts or basis functions, the above regularization and model selection theories are also applicable and helpful. One main weak point of these existing systematic theories that each of them is usually very expensive in computational costs. Thus, several techniques and heuristics are usually combined under the guidance of the theories. The currently popular vector support machine (SVM) is a typical example that can be regarded as a nice combination of kernel approach, sample editing technique[18], and VC dimension based generalization theory[67].

In Sec.4 of this paper, the harmony learning is shown to act as a general framework for learning on all the above three typical models with new results on regularization and model selection. Specifically, supervised BYY harmony learning is introduced via the so called Yang-dominated system and Ying-dominated system. In Sec.4.2, a new regularized ML learning is obtained on three layer nets, a new criterion is given for selecting the number of hidden units, and a family of EM-like algorithms that combines harmony learning with two new techniques of regularization obtained. Moreover, in Sec.4.3 and Sec.4.4, the previous results [74] on the original and alternative ME models as well as RBF nets are further extended in help of the two regularization techniques and a unified RPCL framework with either least complexity nature on the original ME model or automated model selection on the alternative ME model and RBF nets. Furthermore, all the results for the alternative ME model are also applied to not only get an easily implemented approach for determining the smoothing parameter in kernel regression kernel regression and supporting vector machines, but also provide an alternative approach for deciding the set of supporting vectors in the popular supporting vector machines for better generalization. Finally, we conclude in Sec.5.

2. BYY System and Harmony Learning

2.1. *Best harmony learning principle*

In general, specifying a density $g(u)$ involves three issues. The *first issue* is a given structure. A typical example is the finite mixture [20,58,46] as follows:

$$\begin{array}{lll} p(u) & = & \sum_{i=1}^{k}\alpha_i p(u|\theta_i),\ \alpha_i \geq 0,\ \sum_{i=1}^{k}\alpha_i = 1, \\ e.g., & (i) & p(u|\theta_i) = G(u|m_i, \Sigma_i), \\ & (ii) & p(u|\theta_i) = \prod_{j=1}^{d_u} p(u^{(j)}|\theta_{j,i}) \end{array} \quad (1)$$

which is a summation structure that consists of com-

ponents. Each component is either (a) a basic component, in the sense that its density function form is given and there remains only a set of unknown parameters, e.g., the one given by example (i) in eq.(1), or (b) a compound structure that consists of a number of components via either a summation again or a product as shown by example (ii) in eq.(1). The task of *structure design* is to specify the function forms of basic components and the structure that these basic components are organized. The *second issue* is the set $\mathbf{k}$ that describes the scale of a given structure, e.g., $\mathbf{k} = \{k, d_u\}$. The task of specifying the scale is called *model selection* in the sense that a collection of different scales corresponds a collection of specific models that share a same structure but in different scales, and thus selecting a specific scale is equivalent to selecting a model. The *third issue* is a collection θ of unknown parameters. The task of specifying this θ is called *parameter estimation or parameter learning.*

In a conventional sense, learning is a process that specifies a density $g(u) = g(u|\theta, \mathbf{k})$ from a given data set $\mathcal{U} = \{u_t\}_{t=1}^N$, via specifying $\theta, \mathbf{k}$ under a given structure design. Without extra priori constraints, learning from $\mathcal{U}$ is equivalent to learn from its empirical density $p(u) = p_0(u)$ [17,18]:

$$\begin{aligned} p_0(u) &= \tfrac{1}{N}\textstyle\sum_{t=1}^N \delta(u - u_t), \\ \delta_u(u) &= \begin{cases} \lim_{du\to 0} 1/\nu(du), & u = 0, \\ 0, & u \neq 0, \end{cases} \end{aligned} \tag{2}$$

where $\nu(.)$ is a given measure on u. Particularly, for the Lebesque measure, $\nu(du)$ is the volume of du, i.e., $\nu(du) = h^{d_u}$ for a small enough $h > 0$. We have

$$\begin{aligned} \delta_u(0) &= \lim_{du\to 0} \frac{1}{\nu(du)}, \\ \delta_u(0) &= \lim_{h\to 0} \frac{1}{h^{d_u}} \text{ in the Lebesque measure.} \end{aligned} \tag{3}$$

In a broad sense, we consider learning not only in this case but also in the cases that there are two $p(u), g(u)$ in known structures but each of them having some unknown parts, e.g., in either scale or parameter or both. The task of learning is to specify all the unknowns from the known parts of both the densities. Our *fundamental learning principle* is to make $p(u), g(u)$ be best harmony in a twofold sense:

- The difference between the resulting $p(u), g(u)$ should be minimized.
- The resulting $p(u), g(u)$ should be of the least complexity.

Mathematically, we use a functional to measure the degree of harmony between $p(u)$ and $g(u)$. When both $p(u), g(u)$ are discrete densities in the form

$$q(u) = \textstyle\sum_t q_t \delta(u - u_t),\ \sum_t q_t = 1,\ u_t \in \mathcal{U}, \tag{4}$$

such a measure is simply given as follows:

$$H(p||g) = \textstyle\sum_t p_t \ln g_t. \tag{5}$$

When $p = g$ we have $H(p||p)$ which is the negative entropy of p. Interestingly, the maximization of $H(p||g)$ will not only push p, g towards to $p_t = g_t$ but also push $p(u)$ towards to the simplest form

$$p(u) = \delta(u - u_\tau),\ with\ \tau = arg\max_t\ g_t, \tag{6}$$

or equivalently $p_\tau = 1$, and $p_t = 0$ for other t, which is of the least complexity from the statistical perspective. Thus, the maximization of the functional indeed implements the above harmony purpose mathematically.

To extend eq.(5) to cover the cases that $g(u)$ is a continuous density, given a set $\mathcal{U} = \{u_t\}$ of sampling points $\{g(u_t)\}$, we let g_t given by

$$g_t = g(u_t)/z_g,\ z_g = \textstyle\sum_t g(u_t), \tag{7}$$

from which we always have $\sum_t g_t = 1$. For a discrete density, we have $z_g = \delta_u(0)$ when $g(u) = \sum_t g_t \delta_u(u - u_t)$ with $\sum_t g_t = 1$. For a continuous density $g(u)$, when $\mathcal{U}$ has a large enough size N with samples densely covering the entire space of u such that we can approximately regard that $g(u_t)$ is equal to the value of the corresponding histogram density of a hyper-cubic bin of volume $v_u = h^{d_u}$ that contains the point u_t, where $h > 0$ is a very small constant and d_u is the dimension of u. Then, we have

$$\begin{aligned} &\textstyle\sum_t g(u_t) h^{d_u} \approx 1,\ or\ z_g = \sum_t g(u_t) \approx 1/h^{d_u}, \\ &Also\quad 1/h^{d_u} \to \delta_u(0)\ as\ N \to \infty,\ h \to 0. \end{aligned} \tag{8}$$

Putting eq.(7) into eq.(5), we have

$$H(p||g) = \textstyle\sum_t p_t \ln g(u_t) - \ln z_g. \tag{9}$$

In other words, we can use eq.(9) in place of eq.(5) for implementing the best harmony learning.

We can also approximate a continuous $p(u)$ in a similar way and get $\sum_t (p(u_t)/z_p) \ln g(u_t)$. Then,

similar to eq.(8) we have $\sum_t p(u_t)h^{d_u}\ln g(u_t) \approx \int p(u)\ln g(u)\nu(du)$, which is actually also true when $p(u)$ is a discrete density. Thus, a general form of the harmony measure is given as follows:

$$H(p||g) = \int p(u)\ln g(u)\nu(du) - \ln z_g, \qquad (10)$$

where z_g is given in eq.(7).

Therefore, we implement the harmony learning by

$$\max_{\theta,\mathbf{k}} H(\theta,\mathbf{k}),\ H(\theta,\mathbf{k}) = H(p||g), \qquad (11)$$

where θ consist of all the unknown parameters, and $\mathbf{k}$ consist of unknown parts of scales in $p(u), g(u)$.

2.2. *Maximum likelihood learning and two new regularizations*

The harmony learning eq.(11) brings new results both in the above conventional sense via eq.(2) and in a broad sense via the comprehensive BYY learning framework. The latter will be the major focus of this paper. Now, we start at the former to understand the relation between the Maximum Likelihood (ML) learning and the harmony learning eq.(11).

Given $p(u)$ by eq.(2) and given $g(u|\theta)$ with its structure and scale fixed, what we need to do is *parameter learning* for determining θ. Putting these $p(u)$ and $g(u|\theta)$ into eq.(10), in the case of eq.(8) by approximately regarding $z_g \approx 1/h^{d_u} \to \delta_u(0)$, we have that $\max_\theta H(p||g)$ becomes equivalent to

$$\max_\theta L(\theta),\ L(\theta) = \frac{1}{N}\sum_t \ln g(u_t|\theta), \qquad (12)$$

which is exactly the conventional *Maximum Likelihood (ML)* learning on $g(u|\theta)$.

While using z_g in eq.(7) without the approximation in eq.(8), we have that $\max_\theta H(p||g)$ becomes

$$\max_\theta L_R(\theta),\ L_R(\theta) = L(\theta) - \ln[\textstyle\sum_t g(u_t|\theta)], \qquad (13)$$

which consists of the ML learning plus a regularization that prevents $g(u|\theta)$ to over-fit a finite size data set $\mathcal{U}$. This point can be better observed by comparing the gradients:

$$\begin{aligned}
&\nabla_\theta L(\theta) = Gd(\gamma_t)|_{\gamma_t=\frac{1}{N}},\\
&\nabla_\theta L_R(\theta) = Gd(\gamma_t)|_{\gamma_t=\frac{1}{N}-\tilde{g}(u_t|\theta)},\\
&Gd(\gamma_t) = \textstyle\sum_t \gamma_t \nabla_\theta \ln g(u_t|\theta),\\
&\tilde{g}(u_t|\theta) = g(u_t|\theta)/\textstyle\sum_\tau g(\tau|\theta).
\end{aligned} \qquad (14)$$

It follows from $\nabla_\theta L_R(\theta)$ that a de-learning is introduced to ML learning for each sample in proportional to the current fitting of the model to the sample.

An alternative regularization method can also be obtained via the approximation $z_g \approx 1/h^{d_u}$ in eq.(8), with $p(u)$ not given by eq.(2) but by the Parzen window estimate:

$$p(u) = p_{\sigma^2}(u),\ p_{\sigma^2}(u) = \frac{1}{N}\sum_{t=1}^N G(u|u_t,\sigma^2 I). \qquad (15)$$

As $\sigma^2 \to 0$, $G(u|u_t,\sigma^2 I_d) \to \delta(u-u_t)$, and $p_{\sigma^2}(u)$ returns to $p_0(u)$. Generally, $p_{\sigma^2}(u)$ is a smoothed modification of $p_0(u)$ by using a gaussian kernel $G(u|0,\sigma^2 I_d)$ with mean 0 and covariance $\sigma^2 I_d$ to blur out each impulse at u_t.

We have[70] $h = \sqrt{2\pi}\sigma$ and thus it follows that $\max_\theta H(p||g)$ becomes equivalent to

$$\begin{aligned}
&\max_{\theta,\sigma^2} L_S(\theta,\sigma^2),\ L_S(\theta,\sigma^2) = 0.5 d_u \ln\sigma^2 +\\
&\frac{1}{N}\textstyle\sum_t \int G(u|u_t,\sigma^2 I)\ln g(u|\theta)du,
\end{aligned} \qquad (16)$$

which regularizes the ML learning by smoothing each likelihood $\ln g(u_t|\theta)$ in the near-neighbor of u_t. This new regularization method is referred as *data smoothing*. Also, we have[70]

$$\begin{aligned}
&\sigma^2 = N d_u/\textstyle\sum_\tau Tr[H_g(u_\tau|\theta)],\ \ H_g(u_t|\theta) \text{ is}\\
&\text{the Hessian of } \ln g(u|\theta) \text{ with respect to } u,
\end{aligned} \qquad (17)$$

which is closely related to the Tikhonov-type regularization theory [66,24,5]. We can directly put eq.(17) into eq.(16) such that we need only to update θ to maximize $L_S(\theta,\sigma^2(\theta))$.

Moreover, we can also consider to estimate both θ,σ^2 in implementation of eq.(16) by iterating the following two steps:

$$\begin{aligned}
&Step\ \ 1: && \text{fix } \theta, \text{ get } \sigma^2 \text{ by eq.(17)},\\
&\quad or && \text{get } \sigma^{new} = \sigma^{old} + \eta_0 \frac{dL_S(\theta,\sigma^2)}{d\sigma},\\
&Step\ \ 2: && \max_\theta \textstyle\sum_t \int G(u|u_t,\sigma^2 I)\ln g(u|\theta)du,
\end{aligned} \qquad (18)$$

where $\eta_0 > 0$ is a small stepsize. We can further solve Step 2 in one of two ways. One is turning it into ML learning by Monte Carlo sampling. That is, we approximately solve it by

$$\begin{aligned}
&Step\ \ 1: && \text{get a set } \{u'_t\}_{t=1}^{N'}, u'_t = u_t + \varepsilon_t,\\
&&& \varepsilon_t \text{ is a sample from } G(u|0,\sigma^2 I_d)\ ,\\
&Step\ \ 2: && \max_\theta L'(\theta),\ L'(\theta) = \frac{1}{N'}\textstyle\sum_t \ln g(u'_t|\theta).
\end{aligned} \qquad (19)$$

The second way is to turn the problem eq.(16) into

$$\max_{\theta}\{L(\theta) + \frac{0.5\sigma^2}{N}\sum_t Tr[H_g(u_t|\theta)]\}, \qquad (20)$$

where $H_g(u_t|\theta)$ is the same as in eq.(17). Eq.(20) is obtained in help of the Taylor expansion of $\ln g(u|\theta)$ with respect to u around $E(u)$ up to the 2nd order, as later shown in eq.(27).

Given $p(u)$ by eq.(2) and given $g(u|\theta)$ with its structure as in eq.(1), the dimension d_u of u is known but there is an unknown scale k. In this case, the model selection part in eq.(11) is made in the sense of maximum likelihood via eq.(12), eq.(13), and eq.(16), that is, by $\max_{\theta,\mathbf{k}} L(\theta)$ as previously suggested[81] or by its new variants $\max_{\theta,\mathbf{k}} L_R(\theta)$ and $\max_{\theta,\sigma^2,\mathbf{k}} L_S(\theta,\sigma^2)$, corresponding to eq.(16). Moreover, when k is large enough and for $g(u|\theta)$ given in eq.(1) there is no constraint that prevents α_j to become zeros, the model selection can be implied automatically during parameter learning that pushes some of α_j to be zeros, as previously suggested[81].

However, since $p(u)$ is fixed by eq.(2) and thus there is no force of type eq.(6) that imposes the least complexity. Thus, this is not a best way of using the harmony learning eq.(11) for model selection. In contrast, as to be shown in the sequel, the harmony learning on the following *Bayesian Ying-Yang (BYY) system* can take the advantage of eq.(6) in performing model selection.

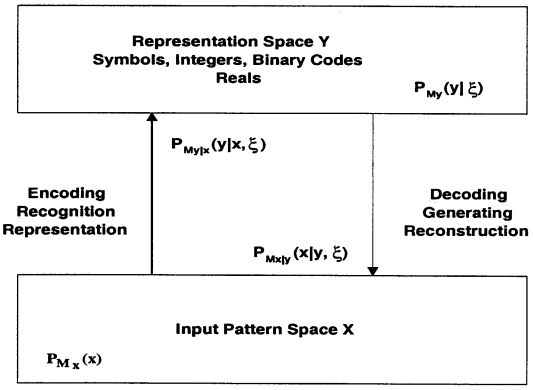
2.3. *Bayesian Ying Yang system*

We consider $u = (x, y)$ with $x \in X$ observable and $y \in Y$ invisible as shown in Fig.1. Given a set of observable samples $\mathbf{x} = \{x_t\}_{t=1}^N$ from X, we can get a density p_{M_x} either empirically via $p_0(x)$ by eq(2) or in a Parzen window estimate $p_{\sigma^2}(x)$ eq(15) under the smoothing parameter σ^2.

On one hand, we can interpret that each x_t is generated from an invisible inner representation y_t via a backward density $p_{M_{x|y}}$. At this moment, we simply ignore the notation ξ in Fig.1, which will be explained after eq.(23). The mapping from Y to X by $p_{M_{x|y}}$ can be understood from two perspectives. One is sample-to-sample mapping $y_t \to x_t$ in three choices as given in Fig.1. The other is a generative model

$$p_M(x) = \int p_{M_{x|y}}(x|y)p_{M_y}(y)\nu(dy) \qquad (21)$$

that maps an inner density $p_{M_y}(y)$ in a structure that is designed according to the nature of tasks encountered.



Encoding
Stochastic: randomly pick y by $P_{My|x}(y \mid x, \xi)$
maximum posteriori: y = argmax$_y$ $P_{My|x}(y \mid x, \xi)$
regression: E[y|$P_{My|x}$, ξ]

Decoding
Stochastic: randomly pick x by $P_{Mx|y}$ (x|y, ξ)
maximum posteriori: x = argmax$_x$ $P_{Mx|y}(x \mid y, \xi)$
regression: E[x| $P_{Mx|y}$, ξ]

Fig. 1. BYY Learning System

On the other hand, we can interpret that each x_t is represented by being mapped into an invisible inner representation y_t via a forward path $p_{M_{y|x}}$. Again, the mapping from X to Y by $p_{M_{y|x}}$ can be understood similarly from either the sample-to-sample mapping $x_t \to y_t$ in three choices as shown in Fig.1 or a representative model

$$p_M(y) = \int p_{M_{y|x}}(y|x)p_{M_x}(x)\nu(dx) \qquad (22)$$

that matches the inner density $p_{M_y}(y)$ in a specific structure.

The above two perspectives reflect the two types of Bayesian decomposition of the joint density $p(x|y)p(y) = p(x,y) = p(x)p(y|x)$ on $X \times Y$. Without any constraints, the two decompositions should be theoretically identical. However, in our above considerations, the four components $p_{M_{y|x}}, p_{M_x}, p_{M_{x|y}}, p_{M_y}$ are subject to certain structural constraints, and thus we usually have two different but complementary Bayesian representations:

$$p_{M_1}(x, y|\xi) = p_{M_{y|x}}(y|x,\xi)p_{M_x}(x|\xi),$$
$$p_{M_2}(x, y|\xi) = p_{M_{x|y}}(x|y,\xi)p_{M_y}(y|\xi), \quad (23)$$

where the notation ξ stands for a set of variables that take into consideration the effects of the environment or context. In the case of context-free or environment- irrelevant, i.e., x, y are irrelevant to ξ, we can simply discard ξ. As discussed in the original paper[85], the formalization eq.(23) compliments to the famous eastern ancient Ying-Yang philosophy, with p_{M_1} called Yang model that represents the observation space (or called Yang space) by p_{M_x} and the forward pathway (or so called Yang pathway) by $p_{M_{y|x}}$, and with p_{M_2} called Ying model that represents the invisible state space (or Ying space) by p_{M_y} and the Ying (or backward) pathway by $p_{M_{x|y}}$. Such a pair of Ying-Yang models is then called *Bayesian Ying-Yang (BYY) system.*

The task of BYY learning consists of specifying all the aspects of $p_{M_{y|x}}, p_{M_x}$, $p_{M_{x|y}}$, p_{M_y}, from a given data set $\mathbf{x} = \{x_t\}_{t=1}^N$. There is no unknown on p_{M_x} given by eq(2). More generally, p_{M_x} may also be given as in eq(15) with the unknown smoothing parameter σ^2, which can be studied in a way similar to eqs.(16)&(17)&(18).

What remains to be specified are $p_{M_{y|x}}$, $p_{M_{x|y}}$, p_{M_y}. First, we need to design a combination of structures for $p_{M_{y|x}}, p_{M_y}, p_{M_{x|y}}$ and such a combination is referred as a system architecture, which is further featured by the pairing of structures of $p_{M_{y|x}}, p_{M_{x|y}}$. Specifically, each of $p_{M_{y|x}}, p_{M_{x|y}}$ can be either *structure-free* or parametric. The former means no structural constraint. E.g., we say $p(u|v)$ is structural free in the sense that $p(u|v)$ can be any function that satisfies $\int p(u|v) = 1, p(u|v) \geq 0$. The latter is given by a parametric model. As shown later, a structure-free density is actually specified via learning in terms of other parametric structures. The architecture with both $p_{M_{y|x}}, p_{M_{x|y}}$ being structure-free is meaningless since they are no longer able to be specified via learning. Therefore, there remain the following three combinations for a meaningful architecture:

- *Backward architecture (B-architecture):* $p_{M_{y|x}}$ is structure-free and $p_{M_{x|y}}$ is parametric.
- *Forward architecture (F-architecture):* $p_{M_{x|y}}$ is structure-free and $p_{M_{y|x}}$ is parametric.
- *Bi-directional architecture(BI-architecture)* Both $p_{M_{y|x}}, p_{M_{x|y}}$ are parametric.

After introducing the fundamentals of BYY harmony learning in the following subsection, in Secs.3 & 4 we will focus on the typical examples of the B-architecture and BI-architecture. Examples of the F-architecture are referred to another paper [68].

2.4. *BYY harmony learning*

We first consider the cases that x is real with p_{M_x} given by eq.(2) or eq.(15) and each representation y is discrete, e.g., either $y = 1, \cdots, k$ (as often encountered in classification, clustering, decision and other pattern recognition tasks) or $y = [y^{(1)}, \cdots, y^{(k)}]$ with each $y^{(j)}$ being binary and taking 0 and 1 as encountered in an encoding job. In the cases, $p_{M_{y|x}}, p_{M_y}$ are discrete densities in the weighted sum of

$$\begin{aligned} p_{M_{y|x}}(y|x,\xi) &= \textstyle\sum_{\hat{y}} P_{M_{y|x}}(y|x,\xi)\delta(y-\hat{y}), \\ p_{M_y}(y|\xi) &= \textstyle\sum_{\hat{y}} P_{M_y}(y|\xi)\delta(y-\hat{y}), \end{aligned} \tag{24}$$

where $\hat{y}$ is a specific value that y takes. The sum is made over such specific values, P_{M_y} denotes a probability and $P_{M_{y|x}}$ denotes a conditional probability.

Given an environment ξ, we can make the harmony learning eq.(11) on a BYY system directly in help of eq.(10) with $p(u) = p_{M_1}, g(u) = p_{M_2}$ in one of the above three architectures, which results in

$$\begin{aligned} H(\theta, \mathbf{k}|\xi) &= H(p_{M_1}||p_{M_2}) = \tfrac{1}{N}\textstyle\sum_{t=1}^N H_t - \ln z_x, \\ H_t &= \textstyle\sum_y P_{M_{y|x}}(y|x_t,\xi)\ln p_{M_{x|y}}(x_t|y,\xi) \\ &\quad + \textstyle\sum_y P_{M_{y|x}}(y|x_t,\xi)\ln P_{M_y}(y|\xi), \\ z_x &= \begin{cases} 1, & \text{Choice (a)}, \\ \sum_{t=1}^N p_M(x_t|\xi), & \text{Choice (b)}, \\ h_x^{-d_x}, h_x = \sqrt{2\pi}|\sigma_x|, & \text{Choice (c)}; \end{cases} \\ p_M(x|\xi) &= \textstyle\sum_y p_{M_{x|y}}(x|y,\xi) P_{M_y}(y|\xi). \end{aligned} \tag{25}$$

where θ consists of all the unknown parameters, including a free density as a set of infinite number of parameters in a B-architecture or F-architecture. Moreover, z_x comes from z_g given by eq.(7). Since P_{M_y} is a discrete density, similar to what discussed after eq.(7), we get $z_g = z_x\delta_y(0)$. Specifically, we have three choice for z_x. It can be given in ae way similar to z_g in either eq.(7) or eq.(8), resulting in Choice (b) and Choice (c) in eq.(25), respectively. We can also approximately ignore the effect of z_g by simply letting $z_g = 1$, resulting in Choice (a) in eq.(25). Furthermore, we have $-\ln z_g = -\ln z_x - \ln\delta_y(0)$, and from eq.(24) we have $\int_y p_{M_{y|x}}(y|x_t,\xi)\ln p_{M_y}(y|\xi)dy = \sum_y P_{M_{y|x}}(y|x_t,\xi)\ln P_{M_y}(y|\xi) + \ln\delta_y(0)$ such that $\ln\delta_y(0)$ is cancelled in the above H_t. In addition, we also have[70] $h = \sqrt{2\pi}|\sigma|$ in a way similar to the discussion on eq.(16).

When the environment ξ is either constant or irrelevant to our consideration x, y, we can directly implement the harmony learning eq.(11) with $H(\theta, \mathbf{k}|\xi)$ given in eq.(25). When ξ is a stochastic according to a distribution $p(\xi)$, $H(\theta, \mathbf{k}|\xi)$ in eq.(25) also changes randomly with ξ. In this case, we consider

$$H(\theta, \mathbf{k}) = \int p(\xi) H(\theta, \mathbf{k}|\xi) \nu(d\xi), \tag{26}$$

where we encounter ther problem of the integral over ξ, which can be tackled in two ways:

(a) In some cases, $p(\xi)$ can be estimated by eq.(2) from a set of samples of ξ. In this case, the integral becomes a sum naturally.

(b) Instead of knowing the entire density $p(\xi)$, it is usually easier to get its statistics $\bar{\xi} = E(\xi)$, $\Sigma_\xi = E[(\xi - \bar{\xi})(\xi - \bar{\xi})^T]$. In the cases, we consider the Taylor expansion of $H(\theta, \mathbf{k}|\xi)$ with respect to ξ around $\bar{\xi} = E(\xi)$ up to the 2nd order and approximately get

$$\begin{aligned} H(\theta, \mathbf{k}) &= H(\theta, \mathbf{k}|\bar{\xi}) + c_T Tr[\Sigma_\xi H_\xi], \\ H_\xi &= \frac{\partial H(\theta, \mathbf{k}|\xi)}{\partial \xi \xi^T}, \end{aligned} \tag{27}$$

$$c_T = \begin{cases} 0, & \text{the 1st order approximation,} \\ 0.5, & \text{the 2nd order approximation.} \end{cases}$$

Though this paper focuses on the cases that y is discrete, for completeness we remark that all the discussions are extendable to the general cases that y is real and x is either real or binary. The key is to get a set of samples $\{y_{\tau,t}\}$ for each x_t via $p_{M_{y|x}}$. Then, similar to eq.(25), we can have

$$\begin{aligned} H(\theta, \mathbf{k}|\xi) &= \frac{1}{N} \sum_{t=1}^N H_t - \ln z_g, \\ H_t &= \sum_\tau p_{M_{y|x}}(y_{\tau,t}|x_t, \xi) \ln p_{M_{x|y}}(x_t|y_{\tau,t}) \\ &\quad + \sum_\tau p_{M_{y|x}}(y_{\tau,t}|x_t, \xi) \ln p_{M_y}(y_{\tau,t}|\xi); \\ z_g &= \begin{cases} \delta_x(0)\delta_y(0) & \text{discrete } p_{M_{x|y}} \ \& \ p_{M_y}, \\ \delta_y(0) z_x, & \text{real } p_{M_{x|y}} \ \& \text{ discrete } p_{M_y}, \\ z_{xy}, & \text{real } p_{M_{x|y}} \ \& \ p_{M_y}. \end{cases} \end{aligned} \tag{28}$$

Moreover, corresponding to z_g given by eq.(7), z_x, z_{xy} have one of the following choices:

$$\begin{aligned} &(a) \quad z_x = 1, z_{xy} = 1, \\ &(b) \quad z_x \text{ by (b) in eq.(25)}, \ z_{xy} = h_x^{-d_x} h_y^{-d_y}, \\ &(c) \quad z_x \text{ by (a) in eq.(25)}, \ z_{xy} = \sum_{t=1}^N \hat{p}_M(x_t), \\ &\quad \hat{p}_M(x_t) = \sum_\tau p_{M_{x|y}}(x_t|y_{\tau,t}, \xi) p_{M_y}(y_{\tau,t}|\xi). \end{aligned} \tag{29}$$

2.5. *Parameter learning and model selection*

We consider the details of the harmony learning eq.(11) with $H(\theta, \mathbf{k}|\xi)$ in eq.(25) under a constant environment ξ, which will be extended to a stochastic environment ξ in Sec.4.

• **Least complexity and automated model selection** In the BYY system eq.(23), $p(u) = p_{M_1}$ consists of not only p_{M_x} that is basically fixed by eq.(2) or eq.(15) but also $P_{M_{y|x}}$ that is usually free or partially free to be specified. In this case, the force of the type eq.(6) will take effect during the harmony learning eq.(11), which directly pushes $P_{M_{y|x}}$ to a least complexity form. E.g., when $p_{M_{y|x}}$ is free, maximizing $H(\theta, k)$ with respect to $p_{M_{y|x}}$ results in

$$\begin{aligned} P_{M_{y|x}} &= I(y|x_t) = I(y - y_{m,t}), \\ y_{m,t} &= arg \max_y P(y|x_t), \quad or \\ y_{m,t} &= arg \max_y [p_{M_{x|y}}(x_t|y, \xi) P_{M_y}(y|\xi)], \end{aligned} \tag{30}$$

where we use the notation

$$I(u - c) = \begin{cases} 1, & \text{for } u = c, \\ 0, & \text{otherwise,} \end{cases} \tag{31}$$

for a function that takes 1 for u at a specific value c but takes 0 for all $u \neq c$.

Actually, the process of $arg \max_y [p_{M_{x|y}}(x_t|y, \xi) P_{M_y}(y|\xi)]$ in eq.(30) implements a competition coordinately based on both $p_{M_{x|y}}$ that represents the fitting of the component to x_t and P_{M_y} that represents the preference of the component. Such a type of competition is called *coordinate competition* [74] or *posteriori competition* because $I(y|x_t)$ in eq.(30) is actually a winner-take-all (WTA) version of the posteriori probability

$$P(y|x) = \frac{p_{M_{x|y}}(x|y, \xi) P_{M_y}(y|\xi)}{p_M(x|\xi)}. \tag{32}$$

In turn, the shrinking of $p_{M_{y|x}}$ in a least complexity will indirectly push $p_{M_{x|y}}$ and P_{M_y} to the least complexity via the force that lets $g(u)$ to match $p(u)$. Therefore, given ξ we can set the scales in $\mathbf{k}$ large enough and implements the harmony learning eq.(11), i.e.,

$$\max_\theta \ H(\theta), \quad H(\theta) = H(\theta, \mathbf{k}|\xi), \tag{33}$$

which determines θ with an automatic searching on a structure of the least complexity. This nature can be understood by considering $H(\theta) = H(\theta, \mathbf{k}|\xi)$ given by eq.(25) with $z_x = 1$ in its choice (a) and $P_{M_{y|x}}$

given by eq.(30). In this case, eq.(33) becomes maximizing $\sum_{t=1}^{N} H_t$ with

$$H_t = \ln p_{M_{x|y}}(x_t|y_{m,t},\xi) + \ln P_{M_y}(y_{m,t}|\xi). \quad (34)$$

That is, the harmony learning consists of the ML learning on $p_{M_{x|y}}$ and the ML learning on P_{M_y} separately under the best harmony inner representation $y_{m,t}$. The latter will push $P_{M_y}(y_{m,t}|\xi)$ towards its least complexity form $I(y-y_{m,t})$, which, according to the feature of $P_{M_y}(y|\xi)$, results in one of the following two types of behaviors:

(a) Parameter learning with automated model selection It happens when $P_{M_y}(y|\xi) = P_{M_y}(y)$ is irrelevant to ξ and there is no extra constraint on $P_{M_y}(y)$. In this case, pushing P_{M_y} towards $I(y-y_{m,t})$ for each x_t will push P_{M_y} to be as close as possible to $I(y-\hat{y})$ at a specific value $\hat{y}$ that is independent of t. When $y = 1,\cdots,k$ with a large k, it means that $P_{M_y}(y)$ may become zero for some value of y, which is equivalent to reducing k to $k-1$ over even smaller number. When $y = [y^{(1)},\cdots,y^{(k)}]$, it means that $P_{M_y}(y)$ may become $P_{M_y}(y) = P_{M_y}(y^-)I(y^{(j)}-\hat{y}^{(j)})$, which is equivalent to removing the j-th dimension $y^{(j)}$ and thus reducing the dimension k by one, where y^- is resulted from y after removing $y^{(j)}$. In other words, model selection will be made automatically during parameter learning in these cases.

(b) Parameter learning with complexity minimization In the cases that there are some constraints on $P_{M_y}(y|\xi) = P_{M_y}(y)$, e.g., $P_{M_y} = 1/k$ when $y = 1,\cdots,k$ or the covariance of $P_{M_y}(y)$ is $cI, c > 0$ with a constant when $y = [y^{(1)},\cdots,y^{(k)}]$. The constraints prevent k to be reduced. However, the harmony learning eq.(33) will still push $p_{M_{x|y}}$ and P_{M_y} to be minimized in complexity. Also, in the cases that $P_{M_y}(y|\xi)$ changes with ξ though there is on extra constraint on $P_{M_y}(y|\xi)$, pushing P_{M_y} towards its least complexity form $I(y-y_{m,t})$ does not necessarily push P_{M_y} to be as close as possible to $I(y-\hat{y})$ because this value $y_{m,t}$ may change for different ξ_t. Thus, the learning eq.(33) does not necessarily lead to the consequence that the scale k is automatically reduced.

Generally, the above two types of behaviors may still happen when $P_{M_{y|x}}$ is not free but with k large enough.

• **Coordinated competition and conscience** In the B-architecture, after coordinated competition by eq.(30), we have H_t given by eq.(34) such that the harmony learning can be simply implemented by making ML learning on $p_{M_{x|y}}$ and P_{M_y} separately under the best harmony inner representation $y_{m,t}$. However, like what we discussed on classical competitive learning (CL) and RPCL learning in Sec.1, the coordinated competition eq.(30) is a greedy WTA competition that will create many local optimal solutions in maximizing H_t. Thus, this learning may result in a bad local optimal solution, similar to the 'dead unit' problem as in the classical CL[19,1].

Such a local optimal problem can be solved from two types of strategies as follows:

(i) We can regard this harmony learning as a typical optimization problem of finding the global optimal solution among many local optimal solutions. Then, we use one of the existing classical global optimization techniques for this purpose.

(ii) We can introduce 'conscience' into either or both of the WTA competition eq.(30) and its post-competition ML learning on $p_{M_{x|y}}$ and P_{M_y}.

In this paper, we consider the use of the popular simulated annealing method [36] for implementing the first strategy. As to the implementation of the second strategy, we consider the following typical cases:

(a) When z_x is given by Choice (b) in eq.(25), we have the regularization of type eq.(13) and eq.(14), as discussed in Sec.2.2. Specifically, as to be discussed in details later, this regularization introduces 'conscience' into competition learning in a sense that not only adds a certain degree of de-learning on the winner for 'conscience' but also may make some de-learning on the rival and probably other losers as well to prevent the competition to be weaken too much.

(b) When z_x is given by Choice (c) in eq.(25), we have the *data smoothing* regularization of type eq.(16). This regularization introduces 'conscience' to moderate the WTA competition eq.(30) indirectly via adding an extra homogenous variance $\sigma_x^2 I$ to $p_{M_{x|y}}$. To prevent adding in too much 'conscience', an appropriate degree of σ_x^2 is searched in a similar way as discussed in Sec.2.2.

(c) We can also introduce 'conscience' by replacing the WTA competition eq.(30) with a soften posteriori competition by giving $P_{M_{y|x}}$ a certain structure instead of being free. A typical example is $P_{M_{y|x}}(y|x) = P(y|x)$ by eq.(32). As shown in [85,80],

the maximization of $H(\theta, \mathbf{k}|\xi)$ under this constraint becomes equivalent to minimizing the following Kullback divergence:

$$\min_\theta \; KL(p_{M_1}||p_{M_2}|\xi), \; where \; KL(p||g) = H(p||p) - H(p||g) = \int p(u) \ln \frac{p(u)}{g(u)} \nu(du). \quad (35)$$

It is further equivalent to the ML learning that only focuses on the marginal distribution $p_M(x)$ in eq.(25), which indirectly takes $p_{M_{x|y}}, P_{M_y}$ in consideration via a summation. Though the minimization of $KL(p||g)$ pushes $p(u), g(u)$ to best match, there is no force that pushes the least complexity. Thus, it is not suitable to be used for making parameter learning with automated model selection.

• **Parameter learning followed by model selection** As a complement to making learning either by eq.(33) with P_{M_y} constrained with a fixed $\mathbf{k}$ or simply by eq.(35), also as an alternative to making learning by eq.(33) with automated model selection at the scales of $\mathbf{k}$ large enough (which is also a kind of wasting resources), we can also make parameter learning and model selection in two sequential steps. That is, we can enumerate $\mathbf{k}$ from small scales incrementally. At each specific setting of $\mathbf{k}$ we perform parameter learning by either eq.(33) or eq.(35) to get the best parameter value θ^*. Then, we make a selection on a best $\mathbf{k}^*$ by

$$\min_{\mathbf{k}} J(\mathbf{k}), \quad J(\mathbf{k}) = -H(\theta^*, \mathbf{k}|\xi), \quad (36)$$

where we take the smallest one if $\mathbf{k}$ has more than one choices at which $J(\mathbf{k})$ gets the same minimum.

From an information geometry perspective[70], maximizing $H(p||g)$ corresponds to the maximum projection of p on q and minimizing $KL(p||g)$ corresponds to the minimum residual of this projection, which are complementary and closely related, but not equivalent. Therefore, there may be a discrepancy between the resulted θ^* in the two ways. However, this discrepancy will be remedied by the same subsequent step of model selection eq.(36).

3. Unsupervised Learning, Gaussian Mixture and Unified RPCL

3.1. *Gaussian mixture, harmony learning, and unified RPCL*

We consider a simple BYY system with its Ying pathway given as follows:

$$p_{M_y} = \sum_{j=1}^{k} \alpha_j \delta(y-j), \; p_{M_{x|y}} = G(x|m_y, \Sigma_y),$$
$$p_M(x) = \sum_{j=1}^{k} \alpha_j G(x|m_j, \Sigma_j), \quad (37)$$

which is a gaussian mixture. Putting eq.(37) into eq.(25) with its environment ξ being irrelevant to learning, we get $H(p_{M_1}||p_{M_2}) = \frac{1}{N}\sum_{t=1}^{N} \sum_{y=1}^{k} P_{M_{y|x}}(y|x_t) \ln [G(x_t|m_y, \Sigma_y)\alpha_y] - \ln z_x$ or

$$H(\theta, k) = L_{x|y}(\theta, k) + H_y(\theta, k) - \ln z_x, \quad (38)$$
$$H_y(\theta, k) = \textstyle\sum_{y=1}^{k} \hat{\alpha}_y \ln \alpha_y,$$
$$\hat{\alpha}_y = \frac{1}{N}\sum_{t=1}^{N} P_{M_{y|x}}(y|x_t), \; L_{x|y}(\theta, k) = \frac{1}{N}\sum_{t=1}^{N}\sum_{y=1}^{k} P_{M_{y|x}}(y|x_t) \ln G(x_t|m_y, \Sigma_y).$$

where z_x is given by one of three choices as in eq.(25), and k denotes the only scale (i.e., the number of gaussians in a mixture). In the B-architecture, $p_{M_{y|x}}$ is given by eq.(30) with

$$y_{m,t} = arg \max_y [G(x_t|m_y, \Sigma_y)\alpha_y]. \quad (39)$$

While in a BI-architecture, a typical choice of $P_{M_{y|x}}$ is given by the posteriori density

$$P_{M_{y|x}}(y|x) = \frac{G(x_t|m_y, \Sigma_y)\alpha_y}{\sum_{i=1}^{k} G(x_t|m_i, \Sigma_i)\alpha_i}. \quad (40)$$

For the above setting, the learning task is to determine the scale k and the unknown parameter set $\theta = \{\alpha_j, m_j, \Sigma_j\}_{j=1}^{k}$. We can implement parameter learning eq.(33) with automated model selection at a given large k. Alternatively, we can also enumerate a number of k values and make parameter learning at each k by either eq.(33) or eq.(35), followed with model selection by eq.(36), with $J(k)$ simplified into

$$J(k) = \ln z_x + \begin{cases} (a) \; \ln k + 0.5 d_x \ln \sigma^2, \\ (b) \; \ln k + 0.5 d_x \sum_{y=1}^{k} \alpha_y \ln \sigma_y^2, \\ (c) \; \sum_{y=1}^{k} \alpha_y \ln \frac{|\sigma_y|^{d_x}}{\alpha_y}, \\ (d) \; \sum_{y=1}^{k} \alpha_y \ln \frac{|\Sigma_y|^{0.5}}{\alpha_y}, \end{cases} \quad (41)$$

where (d) is for a general case of covariance matrices $\{\Sigma_y\}_{y=1}^{k}$, which usually have certain specific structures. For example, we have the case (c) when $\Sigma_y = \sigma_y^2 I$, the case (b) if $\alpha_y = 1/k$ too, and finally the case (a) if the variance also becomes same for each y, i.e., $\Sigma_y = \sigma^2 I$. Particularly, when z_x is

given by Choice (a) and Choice (b) in eq.(25), $\ln z_x$ becomes irrelevant to k, $J(k)$ in eq.(41) becomes the same as Eq.(24), Eq.(25) and Eq.(30) in the previous paper [80].

In implementation, for different specific structures of covariance matrices, the detailed updating rules should also be modified accordingly. Shown in Tab.1 are typical examples for estimating covariance matrices. It should be noticed is that the updating rules in Item B of Tab.1 obey the constraint that a covariance matrix should be positive definite. Moreover, the eigen-decomposition structure of covariance matrices shown in Item (C) of Tab.1 are very useful too. E.g., gaussian mixture eq.(37) with each gaussian in such a structure represents a local factor analysis model or a local subspace models [29,87,76].

Tab.1 Updating Rules for Covariance Matrices

(A) $\max_\Sigma \sum_{t=1}^N \eta_t \ln G(e_t\|0,\Sigma), \eta_t \geq 0, \forall t$
In batch: $\Sigma = \frac{S}{n}$, $S = \sum_{t=1}^N S_t$, $n = \sum_{t=1}^N \eta_t$, Adaptive: $\Sigma^{new} = (1-\eta_0\eta_t)\Sigma^{old} + \eta_0 S_t$. $S_t = \eta_t \begin{cases} e_t e_t^T, & \text{(a) } \Sigma \text{ in general,} \\ diag[e_t e_t^T], & \text{(b) } \Sigma \text{ is diagonal,} \\ d_u^{-1}\|e_t\|^2, & \text{(c) } \Sigma = \sigma^2 I. \end{cases}$ $\eta_0 > 0$ is a small constant, and the diagonal $diag[F]$ consists of diagonal elements of F.
(B) $\max_\Sigma \sum_{t=1}^N \eta_t \ln G(e_t\|0,\Sigma)$, η_t maybe negative or positive.
The rules in Item A can no longer guarantee that Σ is positive definite. Instead, we use $\Sigma = BB^T$ and update $B^{new} = B^{old} + \eta_0 \delta B$ $\delta B = (\Sigma^{-1}S\Sigma^{-1} - \zeta\Sigma^{-1})B$, $S = \begin{cases} \sum_{t=1}^N S_t, & \text{in batch,} \\ S_t, & \text{adaptive,} \end{cases}$, $\zeta = \begin{cases} \sum_{t=1}^N \eta_t, & \text{in batch,} \\ \eta_t, & \text{adaptive.} \end{cases}$
(C) $\max_\Phi \sum_{t=1}^N \eta_t \ln G(e_t\|0,\Sigma)$ in help of the following eigen-decomposition
$\Sigma = \sigma_0^2 I + \sum_{j=1}^m \sigma_j^2 \phi_j \phi_j^T$, $m \leq d_u$, $\|\phi_j\|^2 = 1$, $\phi_i^T \phi_j = 0, i \neq j$, $\Phi = \{\sigma_0, \{\sigma_j, \phi_j\}_{j=1}^m\}$, $\phi_j, \sigma_j^2 > 0$ are the j-th eigen-vector and eigen-value of $\Sigma - \sigma_0^2 I$, *respectively*;

A general algorithm is given in Tab.2 for parameter learning on θ at a given k. In the sequel, we discuss a number of its typical cases.

• **KMEAN clustering, Mahalanobis extensions, and the hard-cut EM algorithm** We consider the algorithm in Tab.1 in the following specific settings:

Tab.2 The EM Algorithm and Adaptive Algorithms for Gaussian Mixtures

(A) A General Algorithm
Step 1 : For each x_t, get $\eta_t(y)$ as shown in the Item B below, based on a subset Y_t that consists of $\hat{k}_t \leq k$ different indices of $\{1,\cdots,k\}$ that correspond the first $\hat{k}_t$ largest values of $p_{i,t} = G(x_t\|m_i,\Sigma_i)\alpha_i, i = 1,\cdots,k$. Particularly, when $\hat{k}_t = 1$, Y_t has only $y_{m,t} = arg\max_i[G(x_t\|m_i,\Sigma_i)\alpha_i]$. Step 2 : update $\theta_y = \{m_y, \Sigma_y, \alpha_y\}$ by solving the root of equation $\nabla_{\theta_y} H = 0$ or by gradient ascending $\theta_y^{new} = \theta_y^{old} + \eta_0 \nabla_{\theta_y} H$, $\nabla_{\theta_y} H = \begin{cases} \sum_{t=1}^N \sum_y \eta_t(y) \nabla_{\theta_y} \ln[G(x_t\|m_y,\Sigma_y)\alpha_y], & \text{(a)}, \\ \sum_y \eta_t(y) \nabla_{\theta_y} \ln[G(x_t\|m_y,\Sigma_y)\alpha_y]. & \text{(b)}, \end{cases}$ either in batch as (a) or adaptively as (b).
(B) $\eta_t(y) = \frac{P_{M_{y\|x}}(y\|x_t)}{N} - \gamma_t P(y\|x_t)$,
$P_{M_{y\|x}}(y\|x_t) = \begin{cases} I(y - y_{m,t}), & \text{eq.(30)}, \\ \frac{G(x_t\|m_y,\Sigma_y)\alpha_y}{\sum_{j=1}^k G(x_t\|m_j,\Sigma_j)\alpha_j}, & \text{eq.(40)}, \end{cases}$ $\gamma_t = \begin{cases} \frac{p_M(x_t)}{\sum_{t=1}^N p_M(x_t)}, & \text{Choice (b) in eq.(25)}, \\ 0, & \text{(a) \& (c) in eq.(25)}, \end{cases}$ $p_M(x_t) = \sum_{y \in Y_t} G(x_t\|m_y,\Sigma_y)\alpha_y$, $P(y\|x_t) = \frac{\sum_{\hat{y} \in Y_t} G(x_t\|m_{\hat{y}},\Sigma_{\hat{y}})\alpha_{\hat{y}} I(y-\hat{y})}{p_M(x_t)}$.
(C) EM Algorithm for Step 2 (if $\gamma_t = 0, \forall t$)
$N_{y,ef} = \sum_{t=1}^N \eta_t(y)$, $\alpha_y = \frac{N_{y,ef}}{\sum_{y=1}^k N_{y,ef}}$ $m_y = \frac{1}{N_{y,ef}} \sum_{t=1}^N \eta_t(y) x_t$, $e_{y,t} = (x_t - m_y^{old})$, $\Sigma_y = \sigma_x^2 I + \frac{1}{N_{y,ef}} \sum_{t=1}^N \eta_t(y) e_{y,t} e_{y,t}^T$.
(D) Iterative Rules for Step 2 (any cases)
$\alpha_y = \frac{e^{c_y}}{\sum_j e^{c_j}}$, $c_y^{new} = c_y^{old} + \eta_0 \delta c_y$, $\delta c_y = \begin{cases} \sum_{t=1}^N \delta c_{y,t}, & \text{(a) in batch,} \\ \delta c_{y,t}, & \text{(b) adaptively;} \end{cases}$ $\delta c_{y,t} = \eta_t(y) - \alpha_y \sum_r \eta_t(r)$, $m_y^{new} = m_y^{old} + \eta_0 \delta m_y$, $\delta m_y = \begin{cases} \sum_{t=1}^N \eta_t(y) e_{y,t}, & \text{(a) in batch,} \\ \eta_t(y) e_{y,t}, & \text{(b) adaptively;} \end{cases}$ Update Σ_y as in Tab.1 where Σ, η_t, S_t are substituted by $\Sigma_y, \eta_t(y), S_{t,y}$ with $S_{t,y} = \eta_t(y) \begin{cases} \sigma_x^2 I + e_{y,t} e_{y,t}^T, & \text{(a) } \Sigma_y \text{ in general,} \\ \sigma_x^2 I + diag[e_{y,t} e_{y,t}^T], & \text{(b) } \Sigma_y \text{ is diagonal,} \\ \sigma_x^2 + d_x^{-1}\|e_{y,t}\|^2, & \text{(c) } \Sigma_y = \sigma_y^2 I. \end{cases}$

(a) When $z_x = 1$ is given by Choice (a) in eq.(25) with $\sigma_x^2 = 0$ in Items (C) & (D), we have that Step 1 in Item (A) is simply given by eq.(30), i.e., Y_t consists of only one index $y_{m,t}$. Thus, $\eta_t(y) = P_{M_{y\|x}}(y\|x_t)/N$

because $\gamma_t = 0$.

(b) Step 2 in Item (A) is either given by Item C or Item D with $\sigma_x^2 = 0$ in updating Σ_y.

The algorithm in this setting becomes exactly (a) the KMEAN algorithm for the least square distance clustering when $\Sigma_y = \sigma^2 I, \alpha_j = 1/k$ simply; (b) an extended KMEAN algorithm that performs Mahalanobis (or elliptic) distance clustering when the above constraint $\Sigma_y = \sigma^2 I$ is released; and (c) the so called hard-cut EM algorithm previously given in [80] when the constraint $\alpha_j = 1/k$ is also released. The details are referred to in the paper [80].

This understanding provides a new perspective on the KMEAN algorithm and its extensions. According to the previous discussion, model selection can be made automatically during parameter learning eq.(33). However, such a behavior has not been observed in the past studies of the KMEAN algorithm. The reason is that the constraint $\alpha_y = 1/k$ for each y is implied in the KMEAN algorithm such that we can only get complexity minimization under a fixed scale k instead of automated model selection.

Conceptually, the learning by the hard-cut EM algorithm should be accompanied with automated model selection since the constraint $\alpha_j = 1/k$ is released. In implementation, however, maximizing $H(\theta, k)$ in eq.(38) by this algorithm may stick at a local maximum.

• The EM algorithm, ML estimation and harmony learning As discussed in Sec.2.4, one solution to improve the above local maximum problem is to introduce 'conscience' by replacing the WTA competition eq.(30) with $P_{M_{y|x}}$ in a certain structure. Specifically, in the implementation of the algorithm in Tab.2, we may use one of the following three strategies:

(a) In the case $z_x = 1$, we let $\eta_t(y) = P_{M_{y|x}}(y|x_t)/N$ with $P_{M_{y|x}}$ by eq.(40) as Step 1 in Item (A) with Step 2 in Item (A) given by either Item C or Item D with $\sigma_x^2 = 0$. The algorithm in this special case becomes exactly the well known EM algorithm for ML estimation on gaussian mixture [80,58]. However, as previously discussed in Sec.2.4, the learning is no longer maximizing $H(p_{M_1}||p_{M_2})$ in eq.(38) but minimizing the Kullback divergence $KL(p_{M_1}||p_{M_2})$ which does not force the least complexity nature and no automatic model selection is made. Such a weak point can be remedied by a simulated annealing [36] process that maximizes the weighted average

$$(1-\lambda)H(p_{M_1}||p_{M_2}) - \lambda KL(p_{M_1}||p_{M_2}), \qquad (42)$$

with $0 \leq \lambda \leq$ starting at 1 and then gradually reducing to 0 during learning. In this way, the learning gradually shifts from minimizing $KL(p_{M_1}||p_{M_2})$ to maximizing $H(p_{M_1}||p_{M_2})$. In implementation, under a fixed $P_{M_{y|x}}$, either maximizing $H(p_{M_1}||p_{M_2})$ or minimizing $KL(p_{M_1}||p_{M_2})$ leads to the same Step 2 in Tab.2. Thus, only Step 1 is affected, where the above weighted average is maximized through updating $P_{M_{y|x}}$.

(b) We can perform another simulated annealing [36] process that approximately uses a weighted average of $P_{M_{y|x}}$ given by eq.(40) and that by eq.(30), that is,

$$\eta_t(y) = \frac{1}{N}[(1-\lambda)I(y|x_t) + \lambda P_{M_{y|x}}(y|x_t)], \qquad (43)$$

as Step 1 in Tab.2, with λ gradually reducing from 1 to 0.

(c) We can also directly insert $P_{M_{y|x}}$ by eq.(40) into $H(\theta, k)$ in eq.(38), and then maximize the resulted $H(\theta, k)$ with respect to θ by gradient ascending technique.

• Data smoothing regularization As discussed in Sec.2.4, the second solution for the local maximum problem is the algorithm in Tab.2 at the special case that a smoothing parameter $\sigma_x^2 > 0$ or $h_x = \sqrt{2\pi}|\sigma_x|$ is used in Item C or Item D during updating Σ_y. This special case provides either a smoothed hard-cut EM algorithm by Item C or a smoothed adaptive EM algorithm by Item D. For a fixed σ_x^2, both cases implement a regularized learning that maximizes

$$H(\theta, k) = H(p_{M_1}||p_{M_2}) = d_x \ln(\sqrt{2\pi}\sigma_x) + \int p_{\sigma_x}(x) \sum_{y=1}^{k} P_{M_{y|x}}(y|x) \ln[G(x|m_y, \Sigma_y)\alpha_y] dx,$$

where $p_{\sigma_x}(x)$ is given by eq.(15).

We can search a best smoothing parameter $\sigma_x^2 > 0$ in a similar way to what discussed in Sec.2.2.

Alternatively, we can start a value σ_x^2 larger enough. During parameter learning by the above discussed algorithm, we gradually reduce σ_x^2 from a larger enough initial value towards zero, which is also a kind of simulated annealing process [36] that makes the learning tends to a global maximum of $H(\theta, k)$ in eq.(38), even in the previously discussed case $z_x = 1$.

• **Parzen window density on support vectors** Considering one simplified case of eq.(37), in help of one of the algorithms in Tab.2, we can also get a new result on Parzen window density eq.(15). At the setting

$$k = N, t = y, \Sigma_y = \sigma_t^2 I, m_t = x_t, \tag{44}$$

it is not difficult to observe that the gaussian mixture eq.(37) degenerates to

$$p_{\sigma^2}(x) = \sum_{t=1}^{N} \alpha_t G(x|x_t, \sigma^2 I). \tag{45}$$

Tab.3 Parzen Window Density on Support Vectors

(A) A General Algorithm
Step 1 : For each input x_t, get $\eta_{r,t}$ as shown in the Item B below, based on a subset Y_t that consists of the $\hat{k}_t \leq k$ indices that corresponds to the $\hat{k}_t$ smallest value of $d_r^2 = \frac{\|x_t - x_r\|^2}{\sigma_{x\|r}^2} + d_x \ln \sigma_{x\|r}^2$, or simply $d_r^2 = \|x_t - x_r\|^2$ for the special case $\sigma_{x\|r}^2 = \sigma_x^2$. If $\hat{k}_t = 1$, Y_t contains only $r_{m,t} = arg\min_r d_r^2$.
Step 2 : $\alpha_t = e^{c_t}/\sum_r e^{c_r}$, $c_r^{new} = c_r^{old} + \eta_0 \delta c_r$, $\delta c_r = \begin{cases} \sum_{t=1}^{N} \delta c_{r,t}, & \text{(a) in batch,} \\ \delta c_{r,t}, & \text{(b) per sample;} \end{cases}$ $\delta c_{r,t} = \eta_{r,t} - \alpha_r \sum_t \eta_{r,t}$, update $\sigma_{x\|r}^{new} = \sigma_{x\|r}^{new} + \eta_0 \delta \sigma_{x\|r}$ $\delta \sigma_{x\|r} = \begin{cases} \sum_{t=1}^{N} \delta \sigma_{x\|r,t}, & \text{in batch,} \\ \delta \sigma_{x\|r,t}, & \text{per sample;} \end{cases}$ $\delta \sigma_{x\|r,t} = \eta_{r,t} \frac{d_x \sigma_{x\|r}^{old\ 2} - \|x_t - x_r\|^2}{\sigma_{x\|r}^{old\ 3}}$, For the special case $\sigma_{x\|r}^2 = \sigma_x^2$, instead we do $\sigma_x^{new} = \sigma_x^{new} + \eta_0 \begin{cases} \sum_{r=1}^{N} \delta \sigma_{x\|r}, \\ \delta \sigma_{x\|r,t}. \end{cases}$
(B) $\eta_{r,t} = \frac{P_{M_{y\|x}}(r\|x_t)}{N} - \gamma_t P(r\|x_t)$, $P_{M_{y\|x}}(r\|x_t) = \begin{cases} I(r - r_{m,t}), & \text{(a)} \\ \frac{G(x_t\|x_r, \sigma_{x\|r}^2 I)\alpha_r}{\sum_{j=1}^{N} G(x_t\|x_j, \sigma_{x\|j}^2 I)\alpha_j}, & \text{(b).} \end{cases}$ $\gamma_t = \begin{cases} \frac{p_M(x_t)}{\sum_{t=1}^{N} p_M(x_t)}, & \text{Choice (b) in eq.(25),} \\ 0, & \text{(a) \& (c) in eq.(25),} \end{cases}$ $p_M(x_t) = \sum_{r \in Y_t} G(x_t\|x_r, \sigma_{x\|r}^2 I)\alpha_r$, $P(r\|x_t, \xi_t) = \sum_{j \in Y_t} \frac{G(x_t\|x_j, \sigma_{x\|j}^2 I)\alpha_j I(r-j)}{p_M(x_t)}$.
(C) The EM Algorithm when $\gamma_t = 0, \forall t$
$\alpha_t = \frac{1}{N} \sum_{t=1}^{N} \eta_{r,t}$, $\sigma_{x\|r}^2 = \frac{1}{d_x \alpha_t N} \sum_{t=1}^{N} \eta_{r,t} \|x_t - x_r\|^2$, or $\sigma_x^2 = \frac{1}{N} \sum_{t=1}^{N} \sigma_{x\|r}^2$ if $\sigma_{x\|r}^2 = \sigma_x^2$.

We use one of the algorithms in Tab.2 to estimate the parameters $\{\sigma_t^2, \alpha_t\}$. After learning, a sample x_t with its corresponding α_t smaller than $1/N$ has been discounted and even been discarded when α_t becomes much smaller than $1/N$. Moreover, we can force $\alpha_t = 0$ when α_t is smaller than a threshold $\epsilon < 1/N$ such that x_t is completely discarded. In other words, *we estimate the density based on only a set of supporting sample vectors for a better performance*, instead of equally using all the samples.

Specifically, the detailed algorithms is given in Tab.3. Particularly, for Choice (a) and Choice (c) of z_x in eq.(25), we have $\gamma_t = 0, \forall t$ we get the EM algorithm in Item (C) of Tab.3 from simplifying Item (C) of Tab.2. In Item (A), the set Y_t is obtained via d_r^2 after comparing $d_t^2, t = 1, \cdots, N$, which is made on all the samples. Thus, the learning by algorithms in Tab.3 is of a batch manner in nature. However, we can also update $\alpha_t, \sigma_{x\|r}^2$ per sample as provided in Tab.3.

3.2. *A general RPCL learning framework*

Another solution discussed in Sec.2.4 for the local maximum problem is the algorithm in Tab.2 at the special case that $\sigma_x^2 = 0$ is used in Items (C) & (D) and z_x is given by Choice (b) in eq.(25). In this case, not only 'conscience' is introduced by a certain degree of de-learning on the winner, but also some de-learning are made on the rival and probably other losers, which leads to a RPCL learning framework that extends the original RPCL learning [89].

The key to understanding this general RPCL learning framework is the sign of $\eta_t(y)$ given in Item B of Tab.2.

Given by eq.(30) or eq.(39), $P_{M_{y|x}}$ is zero for each index of $\{1, \cdots, k\}$ except at and only at the winner index $y_{m,t}$ where the maximum posteriori probability occurs. Correspondingly, $\eta_t(y)$ will be either zero or negative on each index that is not $y_{m,t}$. Actually, $\eta_t(y)$ will be zero when $P(y|x_t)$ is zero and will be negative when $P(y|x_t)$ takes a nonzero value.

Moreover, $P(y|x_t)$ is given by Item B, by which whether it takes zero depends on the choices of the set Y_t in Step 1 of Item A. A typical case is that Y_t consists of only the 1st winner $y_{m,t}$ and 2nd winner $y_{r,t}$ (or so called rival), that is,

$$\begin{aligned} Y_t &= \{y_{m,t}, y_{r,t}\}, \\ y_{m,t} &= arg \max_i [G(x_t|m_i, \Sigma_i)\alpha_i], \\ y_{r,t} &= arg \max_{i \neq y_{m,t}} [G(x_t|m_i, \Sigma_i)\alpha_i]. \end{aligned} \tag{46}$$

In this case, it follows from Item B in Tab.2 that $P(y|x_t)$ and thus $\eta_t(y)$ will be nonzero only at $y_{m,t}, y_{r,t}$. Moreover, $\eta_t(y)$ is definitely negative at the rival $y_{r,t}$, while we have

$$\eta(y_{m,t}|x_t)\begin{cases} > 0, & 1/N > \gamma_t P(y_{m,t}|x_t), \\ < 0, & 1/N < \gamma_t P(y_{m,t}|x_t), \end{cases} \quad (47)$$
$$P(y_{m,t}|x_t) = \frac{G(x_t|m_{y_{m,t}}, \Sigma_{y_{m,t}})\alpha_{y_{m,t}}}{\sum_{l=m,r} G(x_t|m_{y_{l,t}}, \Sigma_{y_{l,t}})\alpha_{y_{l,t}}},$$
$$\gamma_t = p_M(x_t)/\sum_{t=1}^{N} p_M(x_t),$$

at the winner $y_{m,t}$, where $p_M(x)$ is given either accurately by eq.(37) or approximately by Item B of Tab.2. Moreover, the adaptive updating on m_y in Item D of Tab.2 is simplified into

$$\gamma_m = \eta(y_{m,t}|x_t),\ \gamma_r = -\theta\eta(y_{r,t}|x_t),$$
$$m_y^{new} = m_y^{old} + \eta_0 \delta m_y,$$
$$\delta m_y = \begin{cases} \gamma_m(x_t - m_y^{old}), & y = y_{m,t}, \\ -\gamma_r(x_t - m_y^{old}), & y = y_{r,t}, \\ 0, & \text{other } y, \end{cases} \quad (48)$$

which becomes the same form as the original RPCL [89]. Actually, it becomes exactly the same in the cases that $1/N > \gamma_t P(y_{m,t}|x_t)$ and thus $\gamma_m > 0$, e.g., it occurs when $\gamma_t \approx 1/N$, which is usually the cases when the mixture fits data reasonably well. As shown by many experiments as well as many allocations discussed in Sec.1, with an appropriate ratio of the winner learning rate over the rival de-learning rate (e.g., a heuristic range $5-20$), the correct number of clusters or gaussians are automatically decided [89] during RPCL learning in the sense that the means of extra classes or gaussians are driven far away from data.

This connection provides further supports to the previously discussed nature of automatic model selection in Sec.2.4. Moreover, we observe that the algorithm eq.(48) will demonstrate not only a similar RPCL behavior but also improved performances in several perspectives:

(a) Instead of setting the learning and de-learning rates γ_m, γ_r in a heuristic way[89], we here get them given by eq.(48), for which $P(y_{m,t}|x_t), P(y_{r,t}|x_t)$ can be calculated online upon the current sample x_t, but γ_t depends on the sum $\sum_{t=1}^{N} p_M(x_t)$. We can approximate the sum by either the moving sum $S_p(t+1) = (1-\lambda)S_p(t) + \lambda p_M(x_t)$ or simple the rough approximation $1/N$.

(b) The original RPCL[89] only updates the means m_y, implicitly with an assumption that $\Sigma_y = \sigma^2 I, \alpha_y = 1/k$. Later in a paper[74], two extensions of RPCL (called Type A and Type B, respectively) have been proposed for covering the general cases of Σ_y, α_y, but still with heuristic rates of learning and de-learning. Here, the updating rules in Item (D) of Tab.2 not only apply to the general cases of α_y, Σ_y too, but also get the learning and de-learning rates by eq.(48). Moreover, in addition to driving extra units far away from data, we can also discard a gaussian when its corresponding α_y is very small.

(c) When $\sum_{t=1}^{N} p_M(x_t)$ is more accurately estimated, as discussed above there are chances that $1/N < \gamma_t P(y_{m,t}|x_t)$ for some x_t and thus $\gamma_m < 0$. That is, the net de-learning occurs for preventing a particular winner to over-dominate data. This effect becomes more clear if we look at the degenerated case $Y_t = \{y_{m,t}\}$ of eq.(46). In this case, eq.(48) is further simplified into

$$m_y^{new} = m_y^{old} + \eta \begin{cases} x_t - m_y^{old} & y = y_{m,t}, \\ 0, & \text{other } y, \end{cases}$$
$$\eta = \eta_0(\frac{1}{N} - \gamma_t), \quad (49)$$

which acts as a regularized version of the previous hard-cut EM algorithm [80], since the cases with $\frac{1}{N} - \gamma_t < 0$ gives the winner more 'conscience' to avoid over-fitting data.

(d) Moreover, we consider the general algorithm in Tab.2. We start at $Y_t = \{1, \cdots, k\}$, where $p_M(x_t)$ in Item B of Tab.2 becomes the same as in eq.(37). In such a case, the algorithm in Item D implements de-learning on all the gaussians or clusters except the winner. The winner usually learns with 'conscience' in the cases with $\eta_t(y) < 0$. In the cases that Y_t is a subset of $\{1, \cdots, k\}$ as described in Item A of Tab.2, we get a similar situation except the learning and de-learning occur only within gaussians in Y_t.

(e) Even more generally, when $P_{M_{y|x}}$ is given by the Bayesian posteriori probability in Item B of Tab.2, $P_{M_{y|x}}(y|x_t)$ will be nonzero not only at $y = y_{m,t}$ but at all the other y. As a result, $\eta_t(y)$ can be negative or positive for each gaussian in various different combinations and thus we get a number of combinations of learning and de-learning.

In summary, the implementation of eq.(38) with z_x in Choice (b) as shown in Tab.2 provides a general RPCL framework that extends the original RPCL learning [89] with improved performances in various perspectives. With k initially given large enough,

model selection for a best k^* can be made automatically during the parameter learning eq.(33) implemented by an algorithm in this RPCL family.

4. Supervised Learning on Three Layer Nets and ME-RBF-SVM Models

4.1. *Supervised BYY harmony learning: Yang-dominated vs Ying-dominated system*

• **Supervised BYY harmony learning via Yang-dominated system** The BYY system and harmony learning can also be applied to supervised learning tasks of mapping $\xi \to x$ based on a given data set $\{\xi_t, x_t\}_{t=1}^N$. We consider $H(\theta, \mathbf{k})$ given in eq.(26) in a stochastic environment ξ that changes according to a distribution $p(\xi)$, with $p(\xi)$ given by eq.(2) and $p_{M_x}(x|\xi)$ simply given by

$$p_{M_x}(x|\xi) = \begin{cases} \delta(x - x_t), & \xi = \xi_t, \\ \text{not-care}, & \text{otherwise.} \end{cases} \quad (50)$$

Again, we get eq.(25), eq.(30), eq.(32), and eq.(34).

The key difference from Sec.3 is here we consider to implement a desired mapping $\xi \to x$ via the generative conditional mixture $p_M(x|\xi)$ in eq.(25) in the following two particular interesting models:

(a) *Three layer net* When $y = [y^{(1)}, \cdots, y^{(k)}]^T$ and $p_{M_{x|y}}(x|y,\xi) = p_{M_{x|y}}(x|y,\theta_{xy})$, i.e., the output x becomes independent from the environment input ξ once knowing the corresponding inner representations. In this case, $p_M(x|\xi)$ in eq.(25) becomes

$$p_M(x|\xi) = \sum_y p_{M_{x|y}}(x|y,\theta_{xy}) P_{M_y}(y|\xi,\theta_{y\xi}). \quad (51)$$

This is actually a general three layer net with the mapping $\xi \to y$ via the hidden layer $P_{M_y}(y|\xi,\theta_{y\xi})$ and the mapping $y \to x$ by the output layer $p_{M_{x|y}}(x|y,\theta_{xy})$.

(b) *The Mixture-of-Expert model* For the case of $y = 1, \cdots, k$, $p_M(x|\xi)$ in eq.(25) becomes

$$p_M(x|\xi) = \sum_{y=1}^k p_{M_{x|y}}(x|y,\xi,\theta_{xy}) P_{M_y}(y|\xi,\theta_{y\xi}), \quad (52)$$

which provides a general form of the mixture-of-expert (ME) model, with $p_{M_y}(y|\xi)$ acting as a general gating net and $p_{M_{x|y}}(x|y,\xi,\theta_{x|y})$ at each y value acting as each local expert.

We can further get various specific forms of the ME model $p_M(x|\xi)$ for different structures of $p_{M_{x|y}}, P_{M_y}$. Typical examples include not only the original ME model [30,33,34], the alternative ME model, the normalized Gaussian RBF Nets [86,74], and other basis functions, but also kernel regression and vector support machines. The details are given in Sec.4.3 and Sec.4.4.

The above two models and the general model $p_M(x|\xi)$ in eq.(25) share a common feature that the pathway from external environment ξ to the inner representation y is directly implemented by a representative (*or so called Yang*) parametric model $p_{M_y}(y|\xi)$ that is learned in help of the harmony learning eq.(33) with $H(\theta,k)$ by by eq.(25) & eq.(34). Thus, we say that such a $p_M(x|\xi)$ is Yang dominated and call the corresponding BYY system shortly by *BYY Yang-dominated system.*

• **Supervised BYY harmony learning via Ying-dominated system** Alternatively, the supervised learning tasks of mapping $\xi \to x$ can also be made by a special case of the harmony learning eq.(11) on the following BYY system

$$\begin{aligned} &p_{M_1}(x,y,\xi) = p_{M_{y|x}}(y|x,\xi) p_{M_x}(x|\xi) p(\xi), \\ &p_{M_2}(x,y,\xi) = p_{M_{x|y}}(x|y,\xi) p_{M_y}(y,\xi), \\ &p_{M_y}(y,\xi) = p(\xi|\phi_y) \textstyle\sum_{\hat{y}} \alpha_y \delta(y - \hat{y}), \\ &\textstyle\sum_y \alpha_y = 1, 0 \le \alpha_y \le 1. \end{aligned} \quad (53)$$

The Yang machine p_{M_1} and its components $p_{M_{y|x}}$, p_{M_x}, $p(\xi)$ remain the same as before. In the Ying machine p_{M_2}, $p_{M_{x|y}}$ remains as before, while $p_{M_y}(y,\xi)$ itself acts as a small Ying machine that consists of parametric density $p(\xi|\phi_y)$ and a discrete density $\sum_{\hat{y}} \alpha_y \delta(y - \hat{y})$.

When $y = 1, \cdots, k$, similar to eq.(25), it follows from eq.(53) that

$$\begin{aligned} &H(\theta,k) = \tfrac{1}{N} \textstyle\sum_{t=1}^N H_t(\theta,k) - \ln z_x, \\ &H_t(\theta,k) = \textstyle\sum_{y=1}^k P_{M_{y|x}}(y|x_t,\xi_t) \times \\ &\ln [p_{M_{x|y}}(x_t|y,\xi_t) p(\xi_t|\phi_y) \alpha_y], \\ &z_x = \begin{cases} 1, & \text{Choice (a)}, \\ \sum_{t=1}^N p_M(x_t,\xi_t), & \text{Choice (b)}, \\ h_x^{-d_x} h_\xi^{-d_\xi}, & \text{Choice (c)}; \end{cases} \\ &p_M(x,\xi) = \textstyle\sum_y p_{M_{x|y}}(x|y,\xi) p(\xi|\phi_y) \alpha_y, \\ &h_x = \sqrt{2\pi}|\sigma_x|, \; h_\xi = \sqrt{2\pi}|\sigma_\xi|. \end{aligned} \quad (54)$$

Similar to eq.(30), a free $P_{M_{y|x}}(y|x_t,\xi_t)$ becomes

$$\begin{aligned} &P_{M_{y|x}}(y|x_t,\xi_t) = I(y|x_t,\xi_t) = I(y - y_{m,t}), \quad (55) \\ &y_{m,t} = arg \max_y [p_{M_{x|y}}(x_t|y,\xi_t) p(\xi_t|\phi_y) \alpha_y], \end{aligned}$$

which is the WTA version of the Bayesian posteriori probability

$$P_{M_{y|x}}(y|x_t,\xi_t)=\frac{p_{M_{x|y}}(x_t|y,\xi_t)p(\xi_t|\phi_y)\alpha_y}{\sum_y p_{M_{x|y}}(x_t|y,\xi_t)p(\xi_t|\phi_y)\alpha_y}. \quad (56)$$

Further from eq.(54), $H_t(\theta,k)$ is simplified into

$$H_t(\theta,k)=\ln\left[p_{M_{x|y}}(x_t|y,\xi_t)p(\xi_t|\phi_y)\alpha_y\right]|_{y=y_{m,t}}. \quad (57)$$

Now the role of mapping $\xi\to x$ is made via the generative finite mixture $p_M(x,\xi)$ in eq.(54) through

$$p_M(x|\xi)=\frac{p_M(x,\xi)}{\sum_r p(\xi|\phi_r)\alpha_r}= \sum_y p_{M_{x|y}}(x|y,\xi)P_{M_y}(y|\xi,\theta_g), \quad (58)$$

The above $p_M(x|\xi)$ is exactly the alternative ME model [86,74], with the Bayesian gate

$$P_{M_y}(y|\xi,\theta_g)=p(\xi|\phi_y)\alpha_y/\textstyle\sum_r p(\xi|\phi_r)\alpha_r,\\ \sum_r\alpha_r=1, 0\le\alpha_r\le 1, \quad (59)$$

As to be shown later in Sec.4.3 and Sec.4.4, the above $p_M(x|\xi)$ again leads us to not only the normalized Gaussian and other basis functions, but also kernel regression and vector support machines.

Though, $p_M(x|\xi)$ in eq.(58) can be written in the same form as that in eq.(54), the pathway from ξ to y is indirectly implemented via $P_{M_y}(y|\xi,\theta_g)$ in eq.(58) in help of the small Ying machine $p(\xi|\phi_y)\alpha_y$ that is learned in help of the harmony learning eq.(33) with $H(\theta,k)$ by eq.(54) & eq.(57). Thus, we call the BYY system of this type shortly by *BYY Ying-dominated system* and the corresponding learning *BYY harmony learning via Ying-dominated system.*

Comparing eq.(34) and eq.(57), both the learning via Yang-dominated system and the learning via Ying-dominated system share the ML learning on $p_{M_{x|y}}(x|y,\xi)$ for the mapping $\xi\to x$. However, the two types of learning are different in the rest part of the tasks. The learning via Yang-dominated system makes ML learning on $P_{M_y}(y|\xi_t,\theta_g)$ that is pushed towards the least complexity form $I(y-y_{m,t})$, i.e., $P_{M_y}(y|\xi_t,\theta_g)=0$ for all y except a particular value $y_{m,t}$. But this value $y_{m,t}$ may change for different x_t, and thus we are not necessarily lead to $P_{M_y}(y)=0$ constantly for some value of y such that the scale k can be automatically reduced.

In contrast, the learning via Ying-dominated system directly models the density of ξ by a number of densities $p(\xi|\phi_y), y=1,\cdots,k$ with each in a priori probability α_y, in help of ML learning that maximizes $\ln\alpha_y$ under the constraint $\sum_{y=1}^k\alpha_y=1$. Thus, α_y goes towards 1 for some value of y and goes towards 0 for some other value of y. For a value $\hat{y}$ with $\alpha_{\hat{y}}\approx 0$, we have $P_{M_y}(\hat{y}|\xi_t,\theta_g)\approx 0$ constantly and thus the corresponding local expert $p_{M_{x|y}}(x|\hat{y},\xi)$ is equivalently discarded. In other words, the learning via Ying-dominated system is accompanied with automated model selection.

4.2. *Three layer net: adaptive algorithm, least complexity, and model selection*

In implementing a mapping $\xi_t\to x_t$ by three layer net, we need to compute $p_M(x|\xi)$ on which we can get x_t by either of the following two choices

$$(a)\ \hat{x}_t=\textstyle\int x p_M(x|\xi_t)dx,\\ (b)\ \hat{x}_t=arg\max_x p_M(x|\xi_t). \quad (60)$$

However, the sum in eq.(51) should be made over 2^k values of y for each x_t, which can be very expensive in computing cost for a large k. To avoid the difficulty, we write eq.(51) as $p_M(x|\xi)=\int p_{M_{x|y}}(x|y,\theta_{xy})\sum_{\hat{y}}P_{M_y}(\hat{y}|\xi,\theta_{y\xi})\delta(y-\hat{y})dy$. Regarding y in $p_{M_{x|y}}$ as a continuous variable, similar to the process of getting eq.(27), we have

$$p_M(x|\xi)=p_{M_{x|y}}(x|y(\xi),\theta_{xy})+\\ c_T p_{M_{x|y}}(x|y(\xi),\theta_{xy})Tr[\Sigma_{y|\xi}H_{y|\xi}]),\\ y(\xi)=\textstyle\sum_y P_{M_y}(y|\xi)y,\ H_{y|\xi}=\frac{\partial\ln p_{M_{x|y}}(x|y(\xi))}{\partial y y^T},\\ \Sigma_{y|\xi}=\textstyle\sum_y P_{M_y}(y|\xi)(y-y(\xi))(y-y(\xi))^T,\\ c_T=\begin{cases}0, & \text{1st order approximation,}\\ 0.5, & \text{2nd order approximation.}\end{cases} \quad (61)$$

There will be no approximation when $P_{M_y}(y|\xi)$ is deterministic, i.e., $P_{M_y}(y|\xi)=I(y-s(\xi,\theta_{y\xi}))$, $\Sigma_{y|\xi}=0$, and thus the 2nd term of $p_M(x|\xi)$ disappears. Usually, we can also ignore this 2nd term by the 1st order approximation via $c_T=0$.

To get a further insight, we look at the following example:

$$p_{M_{x|y}}(x|y(\xi),\theta_{xy})=G(x|Ay(\xi),\Sigma_{x|y}),\\ P_{M_y}(y|\xi)=\prod_{j=1}^k s(\hat{y}^{(j)})^{\hat{y}^{(j)}}[1-s(\hat{y}^{(j)})]^{1-\hat{y}^{(j)}},\\ \hat{y}=W\xi, s(r)=1/(1+e^{-r}). \quad (62)$$

In this case, for $H_{y|\xi}$ and $\Sigma_{y|\xi}$ in eq.(61), we have

$$H_{y|\xi}=A^T\Sigma_{x|y}^{-1}A,\ d_j=s(\hat{y}^{(j)})(1-s(\hat{y}^{(j)})),$$

$$\Sigma_{y|\xi} = diag[d_1, \cdots, d_k]. \quad (63)$$

Tab.4 Adaptive EM-like Algorithms for Three Layer Nets

(A) A General Algorithm

Step 1 : get $y_{m,t}$ by eq.(30),

Step 2 : update θ_{xy} either by solving the root of equation $\nabla_{\theta_{xy}} H(\theta, \mathbf{k}) = 0$ or by gradient ascending $\theta_{xy}^{new} = \theta_{xy}^{old} + \eta \nabla_{\theta_{xy}} H(\theta, \mathbf{k})$,

$$\nabla_{\theta_{xy}} H(\theta, \mathbf{k}) = \begin{cases} \eta_t \frac{\partial \ln p_{M_{x|y}}(x|y_{m,t},\xi,\theta_{xy})}{\partial \theta_{xy}}, & \text{adaptive,} \\ \sum_{t=1}^N \eta_t \frac{\partial \ln p_{M_{x|y}}(x|y_{m,t},\xi,\theta_{xy})}{\partial \theta_{xy}}, & \text{in batch.} \end{cases}$$

$$\eta_t = \frac{1}{N} - \gamma_t, \ \gamma_t = \frac{p_{M_{x|y}}(x_t|y_{m,t},\xi_t,\theta_{xy})}{\sum_{t=1}^N p_{M_{x|y}}(x_t|y_{m,t},\xi_t,\theta_{xy})};$$

Step 3 : update $\theta_{y\xi}$ by gradient ascent $\theta_{y\xi}^{new} = \theta_{y\xi}^{old} + \eta \nabla_{\theta_{y\xi}} H$,

$$\nabla_{\theta_{y\xi}} H = \begin{cases} \eta_t \nabla_{\theta_{y\xi}} \ln P_{M_y}(y_{m,t}|\xi, \theta_{y\xi}), & \text{adaptive,} \\ \sum_{t=1}^N \eta_t \nabla_{\theta_{y\xi}} \ln P_{M_y}(y_{m,t}|\xi, \theta_{y\xi}), & \text{in batch.} \end{cases}$$

(B) A Fast Solution of $y_{m,t}$ with eq.(62)

getting $y_{m,t}$ approximately by solving the root of $\nabla_y [p_{M_{x|y}}(x|y,\xi,\theta_{xy}) P_{M_y}(y|\xi,\theta_{y\xi})] = 0$:

$$\bar{y} = [\bar{y}^{(1)}, \cdots, \bar{y}^{(k)}]^T = A_y^{-1} \bar{x}_d,$$
$$\bar{x}_d = A^T \Sigma_{x|y}^{-1} x + d, \ A_y = A^T \Sigma_{x|y}^{-1} A.$$

Then, turn $\bar{y}$ into $y_{m,t} = \begin{cases} 1, & \text{if } \bar{y}^{(j)} > 0.5, \\ 0, & \text{otherwise.} \end{cases}$

(C) Updating Rules of Step 2 with eq.(62)

$$A^{new} = A^{old} + \eta_0 (R_{xy} - A^{old} R_{yy}),$$

$$R_{yy} = \begin{cases} \sum_{t=1}^N R_{yy,t}, & \text{in batch,} \\ R_{yy,t}, & \text{adaptive;} \end{cases}$$

$$R_{xy} = \begin{cases} \sum_{t=1}^N R_{xy,t}, & \text{in batch,} \\ R_{xy,t}, & \text{adaptive;} \end{cases}$$

$$R_{yy,t} = \eta_t y_{m,t} y_{m,t}^T, \ R_{xy,t} = \eta_t x_t y_{m,t}^T,$$

Update $\Sigma_{x|y}$ as in Tab.1, Σ, S_t are replaced by $\Sigma_{x|y}, S_{t,y}$ with $e_{y,t} = x_t - A y_{m,t}$, $S_{t,y} =$

$$\eta_t \begin{cases} \sigma_x^2 I + e_{y,t} e_{y,t}^T, & \text{(a) } \Sigma_{x|y} \text{ in general,} \\ \sigma_x^2 I + diag[e_{y,t} e_{y,t}^T], & \text{(b) } \Sigma_{x|y} \text{ is diagonal,} \\ \sigma_x^2 + d_x^{-1} \|e_{y,t}\|^2, & \text{(c) } \Sigma_{x|y} = \sigma_{x|y}^2 I. \end{cases}$$

If $\gamma_t = 0, \forall t$, in batch way we have

$$A = R_{xy} R_{yy}^{-1}, \ \Sigma_{x|y} = \frac{\sum_{t=1}^N S_{t,y}}{\sum_{t=1}^N \eta_t}.$$

(D) Updating Rules of Step 3 with eq.(62)

$$W^{new} = W^{old} + \eta_0 \begin{cases} \eta_t \varepsilon_t \xi_t^T, & \text{(a),} \\ \sum_{t=1}^N \eta_t \varepsilon_t \xi_t^T, & \text{(b),} \end{cases}$$

adaptively for (a) and in batch for (b);

$$\varepsilon_t = [\varepsilon_t^{(1)}, \cdots, \varepsilon_t^{(k)}]^T, \ \varepsilon_t^{(j)} = y_{m,t}^{(j)} (1 - s(\hat{y}^{(j)})) - (1 - y_{m,t}^{(j)}) s(\hat{y}^{(j)}).$$

In the case of eq.(62), we only use the 1st order approximation $p_M(x|\xi) = p_{M_{x|y}}(x|y(\xi), \theta_{xy})$, which again becomes exact in the case of a deterministic $P_{M_y}(y|\xi) = I(y - s(\xi, \theta_{y\xi}))$. Moreover, the two choices in eq.(60) coincide:

$$\hat{x}_t = Ay(\xi), \ y(\xi) = s(\xi, \theta_{y\xi})$$
$$= [s(\hat{y}^{(1)}), \cdots, s(\hat{y}^{(k)})]^T, \ \hat{y} = W\xi, \quad (64)$$

i.e., it implements a conventional three layer net with one hidden layer of sigmoid units and one output layer of linear units.

If only $P_{M_y}(y|\xi) = I(y - s(\xi, \theta_{y\xi}))$ is deterministic, it follows from eq.(30) that $y_m(x_t, \xi_t) = s(\xi, \theta_{y\xi})$ and from eq.(34) that $H_t(\theta, k|\xi_t) = \ln p_{M_{x|y}}(x_t|s(\xi, \theta_{y\xi}), \xi_t)$. In this case, the learning eq.(33), with $H(\theta, k)$ by eq.(25) and with $z_x = 1$ in choice (a), becomes the conventional ML learning. Moreover, if $\Sigma_{x|y} = \sigma_{x|y}^2 I$, it further becomes the least square learning which is usually implemented by back-propagation technique.

Generally, the harmony learning eq.(33) and eq.(36) with $H(\theta, k)$ given by eq.(25) will provide a number of new results as follows.

• **Regularized maximum likelihood learning via back-propagation** With $P_{M_y}(y|\xi) = I(y - s(\xi, \theta_{y\xi}))$ being deterministic and with z_x in choice (b). In this case, we have

$$\frac{\partial H(\theta,k)}{\partial \theta} = \frac{1}{N} \sum_{t=1}^N \eta_t \frac{\partial \ln p_{M_{x|y}}(x_t|s(\xi,\theta_{y\xi}),\xi_t)}{\partial \theta},$$
$$\eta_t = \frac{1}{N} - \gamma_t, \ \gamma_t = \frac{p_M(x_t|\xi_t)}{\sum_{t=1}^N p_M(x_t|\xi_t)}, \quad (65)$$

which consists of again the ML learning plus a regularization that introduces a de-learning to the ML learning for each sample in proportional to the current fitting of the model to the sample, similar to eq.(13) and eq.(14). This learning can be made in a way similar to back-propagation, either in batch or adaptively. E.g., in an adaptive implementation, what needs to be done is simply replacing the gradient step size η_0 with $\eta_0 \eta_t$. Moreover, we can approximate the sum $\sum_{t=1}^N p_M(x_t|\xi_t)$ by the moving sum $S_p(t+1) = (1 - \lambda) S_p(t) + \lambda p_M(x_t|\xi_t)$ for a suitable $0 < \lambda < 1$.

• **A family of EM-like algorithms: coordinated competition and regularization** In the case of eq.(62), instead of getting $y = s(\xi, \theta_{y\xi})$ at a deterministic $P_{M_y}(y|\xi) = I(y - s(\xi, \theta_{y\xi}))$, we get $y_{m,t}$ in help of the so called coordinated competition eq.(30) which provides an intermediate target for the hidden

layer. That is, it solves the classical credit assignment task[56], without using the chain rule via back propagation. With this technique, we can get a family of EM-like algorithms for learning a three layer net, given in Tab.4.

Precisely, implementing the coordinated competition in eq.(30) is a task of combinatorial optimization. If solving it by enumeration, we need 2^k comparisons which is very expensive for a large k. Alternatively, we use a fast approximation given in Item B of Tab.4 for this purpose. In the general EM-like algorithm given in Tab.4, this coordinated competition acts as Step 1 of Item A that corresponds to the E-step in the EM algorithm.

Step 2 and Step 3 of Item A together act as the M-step in the EM-algorithm, for implementing the learning on the hidden layer and the output layer respectively, either in batch or adaptively. The implementation of Step 3 (e.g., by Item D) remains the same for all the three choices of z_x in eq.(25). However, the implementation of Step 2 has three variants. One is $z_x = 1$ in Choice (a) where no regularization is enforced such that a problem similar to the 'dead unit' problem [19,1] may occur. This problem is solved by the other two variants with z_x given by Choice (b) and Choice (c) in eq.(25). In Choice (b) and with $\sigma_x^2 = 0$ in Item C, a 'conscience' effect is introduced by $\eta_t = \frac{1}{N} - \gamma_t$ that imposes a de-learning regularization via γ_t. While in Choice (c), as discussed in Sec.2.5, a 'conscience' regularization is introduced via a data smoothing parameter $\sigma_x^2 > 0$.

We can determine a best smoothing parameter $\sigma_x^2 > 0$ in a similar way as discussed in Sec.2.2, with σ_x^2 searched by maximizing

$$H(\theta, k) = d_x \ln \sigma_x + \frac{1}{N}\sum_t \int G(x|x_t, \sigma_x^2 I) \ln p_{M_{x|y}}(x_t|y_{m,t}, \xi_t). \quad (66)$$

Also, we can make learning in a simulated annealing process [36] by starting a value σ_x large enough and gradually reducing it towards zero as learning proceeds.

The algorithms in Tab.4 can also be further extended by taking the 2nd order term in eq.(61) into consideration, and thus correspondingly modifying η_t, Item C and Item D in Tab.4.

• **Least complexity and model selection** Discussed in the end of Sec.4.1, as a special case of the learning via Yang-dominated system, each of algorithms in Tab.4 implements harmony learning that pushes $P_{M_y}(y|\xi_t)$ to the least complexity in the sense that $s(\hat{y}^{(j)})$ almost taking either 1 or 0. However, no automated model selection is guaranteed. Alternatively, by enumerating a number of k values incrementally, model selection can be made by eq.(36) with

$$J(k) = \ln z_x + 0.5 \ln |\Sigma_{x|y}| - J'_k, \quad (67)$$

$$J'_k = \begin{cases} \frac{1}{N}\sum_{t=1}^N \sum_{j=1}^k d_{t,j}, & \text{type (a)}, \\ |\Sigma| + k \ln 2, & \text{type (b)}; \end{cases} \quad d_{t,j} = y_m^{(j)}(t) \ln s(\hat{y}^{(j)}) + (1 - y_m^{(j)}(t)) \ln (1 - s(\hat{y}^{(j)})).$$

Type (a) comes from $H_t(\theta, k)$ in eq.(25), and Type (b) comes from $H_t(\theta, k)$ in eq.(54) by approximately considering the inverse of $P_{M_y}(y|\xi_t)$ in help of a backward path

$$p(\xi|\phi_y) = G(\xi|By, \Sigma), \quad \alpha_y = 1/2^k. \quad (68)$$

where the parameters B, Σ are estimated during the implementation of the algorithms in Tab.4 adaptively by

$$B^{new} = B^{old} + \eta_0(R_{\xi y} - B^{old} R_{yy}),$$
$$R_{\xi y} = \begin{cases} \sum_{t=1}^N R_{\xi y,t}, & \text{in batch}, \\ R_{\xi y,t}, & \text{adaptive}; \end{cases}$$
$$R_{\xi y,t} = \eta_t \xi_t y_m(t)^T,$$

update Σ as in Tab.1, where S_t is replaced by $S_{t,\varepsilon}$ with $\varepsilon_t = \xi_t - B^{old} y_m(t)$ and $S_{t,\varepsilon} = \eta_t \begin{cases} \sigma_\varepsilon^2 I + \varepsilon_t \varepsilon_t^T, & \text{(a) } \Sigma \text{ in general}, \\ \sigma_\varepsilon^2 I + diag[\varepsilon_t \varepsilon_t^T], & \text{(b) } \Sigma \text{ is diagonal}, \\ \sigma_\varepsilon^2 + d_\xi^{-1} \|\varepsilon_t\|^2, & \text{(c) } \Sigma = \sigma^2 I. \end{cases}$

If $\gamma_t = 0, \forall t$, in batch we alternatively have

$$B = R_{\xi y} R_{yy}^{-1}, \ \Sigma = \sum_{t=1}^N S_{t,\varepsilon} / \sum_{t=1}^N \eta_t, \quad (69)$$

where R_{yy} is the same of Item C in Tab.4.

4.3. *Original and alternative ME models, RBF nets, other basis functions and kernel regression*

With $y = 1, \cdots, k$, either the ME model $p_M(x|\xi)$ in eq.(25) or the alternative ME model $p_M(x|\xi)$ in eq.(58) implements the mapping $\xi_t \to x_t$ by each individual expert $p_{M_{x|y}}(x|y, \xi)$ weighted by the gating net $p_{M_y}(y|\xi, \theta_g)$. Thus, the computing complexity of the sum over y has reduced considerably from the case of three layer net eq.(51). Moreover, the implementation of the coordinated competition eq.(30) or eq.(55) can be made in a much reduced complexity and thus usually no approximation is

needed. Furthermore, denoting the regression of each $p_{M_{x|y}}(x|y,\xi,\theta_{x|y})$ by $f_y(\xi,A_y)$, e.g., two often used special cases are

$$f_y(\xi,A_y)=\begin{cases}A_y\xi+c_y, & \text{(a) linear,}\\ c_y, & \text{(b) constant,}\end{cases} \quad (70)$$

we have that the regression function by the ME model becomes

$$(a)\quad E(x|\xi)=\sum_{y=1}^{k}f_y(\xi,A_y)P_{M_y}(y|\xi,\theta_{y\xi}),(a)$$
$$(b)\quad E(x|\xi)=\sum_{y=1}^{k}(A_y\xi+c_y)P_{M_y}(y|\xi,\theta_g),(71)$$

which is easily obtainable as a weighted sum of k individual regressions. In other words, the case (a) and case (b) provide a probabilistic piecewise function that consists of a number of nonlinear and liner pieces, respectively, with each piece engaged in with the probability $P_{M_y}(y|\xi,\theta_g)$.

We can further get various specific forms of either the ME model $p_M(x|\xi)$ in eq.(25) or the alternative ME model $p_M(x|\xi)$ in eq.(58) for different structures of $p_{M_{x|y}}$ and P_{M_y}.

One typical example for $p_{M_{x|y}}$ is

$$p_{M_{x|y}}(x|y,\xi)=G(x|f_y(\xi,A_y),\Sigma_{x|y}), \quad (72)$$

with its two typical examples via eq.(70).

Moreover, P_{M_y} is either directly a parametric structure for the Yang-dominated system as shown in the following first two typical examples, or indirectly given via eq.(58) in help of the parametric structures of $p(\xi|\phi_y)$ as shown in the following third example:

(1) *Softmax gate* P_{M_y} is implemented in the so called softmax form [45], that is,

$$P_{M_y}(j|\xi,\theta_g)=e^{u^{(j)}}/\sum_r e^{u^{(r)}},\ u=g(\xi,\theta_g), \quad (73)$$

where $g(\xi,\theta_g)$ is a forward network with the output u, e.g., the simplest case is $g(\xi,\theta_g)=\theta_g\xi+c_g$ with a parametric matrix θ_g and a parametric vector c_g. In this case, $p_M(x|\xi)$ in eq.(25) with $p_{M_{x|y}}$ in eq.(72) becomes exactly the original ME model [30,33,34].

(2) *Weighted positive gate* P_{M_y} is implemented by the normalized weighted positive functions as follows:

$$P_{M_y}(j|\xi,\theta_g)=\alpha_j K(\xi,\theta_{g,j})/\sum_r\alpha_r K(\xi,\theta_{g,r}),$$
$$\alpha_r\geq 0,\ K(\xi,\theta_{g,r})>0. \quad (74)$$

There is the scaling indeterminacy $\alpha'_r=c_r\alpha_r$ and $K(\xi,\theta_{g,r})=c_r^{-1}K(\xi,\theta_{g,r})$, which can be removed either when $\theta_{g,r}$ is fixed such that $K(\xi,\theta_{g,r})$ becomes a fixed function of ξ with no other unknown parameters or when we further impose the following constraint

$$[\sum_r\alpha_r^m]^{1/m}=\gamma,\ 0\leq\alpha_r\leq\gamma,\ m>0. \quad (75)$$

E.g., we have $\sum_r\alpha_r=\gamma$ when $m=1$. We can link eq.(74) to eq.(73) by writing $\alpha_j K(\xi,\theta_{g,j})=exp\{\ln[\alpha_j K(\xi,\theta_{g,j})]\}$ and thus $u^{(j)}=\ln\alpha_j+\ln K(\xi,\theta_{g,j})$. In eq.(73) we have $P_{M_y}(j|\xi,\theta_g)\to 0$ only when $u^{(j)}\to-\infty$. In contrast, one new feature of eq.(74) is that $\alpha_j=0$ constantly implies $P_{M_y}(j|\xi,\theta_g)=0$ irrelevant to any ξ. Also, at ξ, $K(\xi,\theta_{g,j})=0$ implies $P_{M_y}(j|\xi,\theta_g)=0$. It follows from previous discussions et the end of Sec.4.1 that both cases may result in automated model selection. The other feature of eq.(74) is that $g(\xi,\theta_g)$ is separated into k functions such that the cross-talk between learning each $\theta_{g,j}$ can be considerably reduced such that learning becomes easier to be implemented.

(3) *Bayesian gate* P_{M_y} is implemented by eq.(59), which is actually the special case of eq.(74) with $m=1,\gamma=1$ and $K(\xi,\theta_{g,j})$ being a density. With the gate eq.(59) and $p_{M_{x|y}}$ in eq.(72), $p_M(x|\xi)$ in eq.(25) provides a typical structure of the alternative ME model [86,74]. As previously shown [86,74], one major advantage of the gate eq.(59) is that the ML learning on the alternative ME model can be exactly made by the EM algorithm when

$$p_{M_{x|y}}=G(x|A_y\xi+c_y,\Sigma_{x|y}),$$
$$p(\xi|\phi_y)=G(\xi|m_y,\Sigma_y). \quad (76)$$

While the part of the ML learning on the gate in either eq.(73) or gate eq.(74) can not be handled exactly by the EM algorithm.

(c) *Normalized RBF nets* We consider the linear case (b) of eq.(71) and the gate eq.(59) with $p(\xi|\phi_y)$ in eq.(76). Under the following constraint

$$\alpha_y=\sqrt{|\Sigma_y|}/\sum_r\sqrt{|\Sigma_r|}, \quad (77)$$

(e.g., it is satisfied if $\Sigma_r=\Sigma,\forall r$ and thus $\alpha_y=1/k$), we have

$$E(x|\xi)=\frac{\sum_y(A_y\xi+c_y)e^{-0.5(\xi-m_y)^T\Sigma_y^{-1}(\xi-m_y)}}{\sum_y e^{-0.5(\xi-m_y)^T\Sigma_y^{-1}(\xi-m_y)}}. \quad (78)$$

which is exactly is the so called extended normalized Gaussian RBF net [74,32]. Particularly, when $A_y=0$,

it reduces into the normalized Gaussian RBF nets [49,51,88,74].

(d) *Other basis function nets* We further consider the linear case (b) of eq.(71) and the gate eq.(59), where $p(\xi|\phi_y) = Z(\phi'_y)^{-1}\Psi(\xi - m_y|\phi'_y)$ with $\Psi(\xi - m_y|\phi'_y) > 0$ and $\int \Psi(\xi - m_y|\phi'_y)d\xi = Z(\phi'_y) < \infty$ is invariant to its location parameter m_y. Under the constraint $\alpha_r = Z(\phi'_y)/\sum_r Z(\phi'_r)$, we can write eq.(78) into the following general form

$$E(x|\xi) = \sum_y (A_y\xi + c_y)K(\xi, m_y),$$
$$K(\xi, m_y) = \frac{\Psi(\xi - m_y|\phi'_y)}{\sum_y \Psi(\xi - m_y|\phi'_y)}, \quad (79)$$

which represents other normalized basis function nets, with integrable basis function $\Psi(\xi - m_y|\phi'_y)$ that may or may not be radial symmetrical.

More generally, for the gate eq.(74) at the special case that

$$\alpha_j = \frac{1}{k},\ K(\xi, m_y) = \frac{K(\xi, \theta_{g,y})}{\sum_r K(\xi, \theta_{g,r})}, \quad (80)$$

we get eq.(79) being applicable to other normalized basis function nets, with non-local and even non-integrable basis function $K(\xi, \theta_{g,y})$.

(d) *Kernel regression* We further look at the simplest special case of the RBF net eq.(78) at the setting

$$k = N, t = y, \Sigma_t = \sigma^2 I, \alpha_t = 1/N,$$
$$m_t = \xi_t, c_t = x_t, A_t = 0, \quad (81)$$

In this case, the regression eq.(71) of the linear case (b) becomes

$$E(x|\xi) = \sum_t x_t e^{-0.5\frac{\|\xi - \xi_t\|^2}{\sigma^2}} / \sum_t e^{-0.5\frac{\|\xi - \xi_t\|^2}{\sigma^2}}, \quad (82)$$

which, as previously pointed out in [88], is actually gaussian kernel regression that has been widely studied in the literature of statistics [25,17,16,18]. In contrast to RBF nets, the most salient feature of kernel regression is that there is no unknown parameters except that the smoothing parameter σ^2 needs to be pre-specified. Though, many studies are made in literature on getting the smoothing parameter σ^2, there are only theoretical upper bounds for σ^2 and how to estimate a best σ^2 still remains a challenge problem.

More generally, from eq.(79) and eq.(80), we can get

$$E(x|\xi) = \sum_t x_t K(\xi, \xi_t), \quad (83)$$

which is the general form of kernel regression, where kernels are non-radial-symmetrical, non-local and even non-integrable.

4.4. *EM-like algorithms, RPCL learning, automated model selection, and support vector machines*

With a given k, we can implement parameter learning eq.(33) with $H(\theta, k)$ either given by eq.(54) in help of the family of algorithms given in Tab.5 for the alternative model or given by eq.(25) in help of the family of algorithms given in Tab.6 for the original model. As discussed previously, one advantage of the alternative ME model is that we can use the EM algorithm for training its gate in the structure eq.(76).

• **Harmony learning with hard-cut EM algorithm vs. ML learning with EM algorithm** In Tab.5, when $z_x = 1$ is given by Choice (a) in eq.(54), and $\sigma_x^2 = 0, \sigma_\xi^2 = 0$ in Items (C)&(D), Step 1 in Item (A) becomes simply given by eq.(55), i.e., Y_t consists of only one index $y_{m,t}$. Thus, $\gamma_t = 0$ and $\eta_t(y) = P_{M_{y|x}}(y|x_t, \xi_t)/N$. When $P_{M_{y|x}}(y|x_t, \xi_t)$ is given by eq.(55), the algorithm in this case is the hard-cut EM algorithm previously given in a paper[74], where the details are given for implementing the coordinate competition eq.(55) to get $y_{m,t}$ in different situations. Moreover, when $P_{M_{y|x}}(y|x_t, \xi_t)$ is given by eq.(56), we get exactly the EM algorithm that implements the ML learning [86,74].

In the sequel, we provide further insights on Tab.5 and Tab.6 from three aspects.

In Tab.6, with eq.(25) in place of eq.(54), eq.(30) in place of eq.(55), and eq.(32) in place of eq.(56), we also get the hard-cut EM algorithm for the original model and its corresponding the EM algorithm that implements the ML learning [33,34,74].

Similar to Sec.3, though conceptually the hard-cut EM algorithm can implement harmony learning with automated model selection for a given k large enough, this WTA competition may also cause the 'dead unit' problem [1,19]. This problem is alleviated in the soft competition eq.(56) or eq.(32), but which also brings us back to the ML learning with a considerably reduced ability on automated model selection.

• **Harmony learning and conscience** In the family of algorithms in Tab.5, the problem can be solved again by introducing conscience.

Tab.5 EM-like Algorithms for Alternative Mixture-of-Experts and RBF Nets

(A) A General Algorithm
Step 1 : For each x_t, get $\eta_t(y)$ as shown in the Item B below, based on a subset Y_t that consists of $\hat{k}_t \leq k$ different indices of $\{1, \cdots, k\}$ that correspond the first $\hat{k}_t$ largest values of $p_{y,t} = p_{M_{x\|y}}(x\|y,\xi)p(\xi\|\phi_y)\alpha_y, y = 1, \cdots, k$. When $\hat{k}_t = 1$, Y_t contains only the winner $y_{m,t}$ given by eq.(55).
Step 2 : for each y, update $\theta_{x\|y}$ by solving the root of equation $\nabla_{\theta_{x\|y}} H = 0$ or updating $\theta_{x\|y}^{new} = \theta_{x\|y}^{old} + \eta \nabla_{\theta_{x\|y}} H$, $\nabla_{\theta_{x\|y}} H = \begin{cases} \sum_{t=1}^{N} \sum_y \eta_t(y) \nabla_{\theta_{x\|y}} \ln p_{M_{x\|y}}(x\|y,\xi,\theta_{x\|y}), & \text{(a)}, \\ \sum_y \eta_t(y) \nabla_{\theta_{x\|y}} \ln p_{M_{x\|y}}(x\|y,\xi,\theta_{x\|y}), & \text{(b)}, \end{cases}$ either in batch as (a) or adaptively as (b).
Step 3 : $\theta_g^{new} = \theta_g^{old} + \eta \nabla_{\theta_g} H$, $\nabla_{\theta_g} H = \begin{cases} \sum_{t=1}^{N} \sum_y \eta_t(y) \nabla_{\theta_g} \ln [p(\xi\|\phi_y)\alpha_y], & \text{(a)}, \\ \sum_y \eta_t(y) \nabla_{\theta_g} \ln [p(\xi\|\phi_y)\alpha_y], & \text{(b)}, \end{cases}$ either in batch as (a) or adaptively as (b).
(B) $\eta_t(y) = \frac{P_{M_{y\|x}}(y\|x_t,\xi_t)}{N} - \gamma_t P(y\|x_t,\xi_t)$,
$P_{M_{y\|x}}(y\|x_t,\xi_t) = \begin{cases} \text{given by eq.(55)}, \\ \text{given by eq.(56)}, \end{cases}$ $\gamma_t = \begin{cases} \frac{p_M(x_t,\xi_t)}{\sum_{t=1}^{N} p_M(x_t,\xi_t)}, & \text{Choice (b) in eq.(25)}, \\ 0, & \text{(a) \& (c) in eq.(25)}, \end{cases}$ $p_M(x_t,\xi_t) = \sum_{y \in Y_t} p_{M_{x\|y}}(x_t\|y,\xi_t)p(\xi_t\|\phi_y)\alpha_y$, $P(y\|x_t,\xi_t) = \sum_{\hat{y} \in Y_t} \frac{p_{M_{x\|y}}(x_t\|\hat{y},\xi_t)p(\xi_t\|\phi_{\hat{y}})I(y-\hat{y})}{p_M(x_t,\xi_t)}$.
(C) Updating Rules of Step 2 with eq.(76)
$A_y^{new} = A_y^{old} + \eta_0(R_{x\xi,y} - A_y^{old} R_{\xi\xi,y})$, $R_{\xi\xi,y} = \begin{cases} \sum_{t=1}^{N} R_{\xi\xi,y}^{(t)}, & \text{in batch}, \\ R_{\xi\xi,y}^{(t)}, & \text{adaptive}; \end{cases}$ $R_{x\xi,y} = \begin{cases} \sum_{t=1}^{N} R_{x\xi,y}^{(t)}, & \text{in batch}, \\ R_{x\xi,y}^{(t)}, & \text{adaptive}; \end{cases}$ $R_{\xi\xi,y}^{(t)} = \eta_t(y)\xi_t\xi_t^T$, $R_{x\xi,y}^{(t)} = \eta_t(y)x_t\xi_t^T$, update $\Sigma_{x\|y}$ by Tab.1 as Item (C) in Tab.4, with η_t replaced by $\eta_t(y)$, $e_{y,t} = x_t - A_y\xi_t$. If $\gamma_t = 0, \forall t$, in batch we alternatively have $A = R_{x\xi,y}R_{\xi\xi,y}^{-1}$, $\Sigma_{x\|y} = \frac{\sum_{t=1}^{N} S_{t,y}}{\sum_{t=1}^{N} \eta_t(y)}$.
(D) Updating Rules of Step 3 with eq.(76)
With $e_{y,t} = \xi_t - m_y^{old}$ and $\eta_t(y)$ given in Item B, we update α_y, m_y, Σ_y by either Item (c) in Tab.2 with the EM algorithm when $\gamma_t = 0, \forall t$ or by Item (D) in Tab.2 with either of the batch and adaptive rules in any cases of γ_t.
(E) Updating Rules for RBF Nets
Simply set $\alpha_y = \|\Sigma_y\|/\sum_r \|\Sigma_r\|$ in place of updating α_y via c_y in Item (D).

One solution is the algorithm in Tab.5 or Tab.6 at the special cases:

(1) z_x is given by Choice (b) in eq.(54),

(2) $\sigma_x^2 = 0$ is used in Items (C) and $\sigma_\xi^2 = 0$ is used in Items (D),

(3) Y_t in Step 1 of Item (A) contains only the winner $y_{m,t}$ given by eq.(55). We have $\eta_t(y) = \frac{1}{N} - \gamma_t$ that introduces a conscience via a de-learning regularization via γ_t.

Another is the algorithm in Tab.5 or Tab.6 at the special case that z_x is given by Choice (c) in eq.(54). As discussed in Sec.2.4, the smoothing parameters $\sigma_x^2 > 0, \sigma_\xi^2 > 0$ in Item C and Item D introduce a 'conscience' in the WTA competition.

Tab.6 EM-like Algorithms for Mixture-of-Experts

(A) A General Algorithm
Step 1 : The same as Step 1 of Item A in Tab.5, except that $p_{y,t} = p_{M_{x\|y}}(x\|y,\xi)P_{M_y}(y\|\xi)$ and $y_{m,t}$ given by eq.(30) instead of eq.(55).
Step 2 : The same as Step 2 of Item A in Tab.5 in help of the rules of Item (C) in Tab.5.
Step 3 : update $\theta_g^{new} = \theta_g^{old} + \eta \nabla_{\theta_g} H$, $\nabla_{\theta_g} H = \sum_{t=1}^{N} \sum_y \eta_t(y) \nabla_{\theta_g} (u_t^{(y)} - \ln \sum_r e^{u_t^{(r)}})$, $u_t = g(\xi_t, \theta_g)$.
(B) $\eta_t(y) = \frac{P_{M_{y\|x}}(y\|x_t,\xi_t)}{N} - \gamma_t P(y\|x_t,\xi_t)$,
$P_{M_{y\|x}}(y\|x_t,\xi_t) = \begin{cases} \text{given by eq.(30)}, \\ \text{given by eq.(32)}, \end{cases}$ $\gamma_t = \begin{cases} \frac{p_M(x_t\|\xi_t)}{\sum_{t=1}^{N} p_M(x_t\|\xi_t)}, & \text{Choice (b) in eq.(25)}, \\ 0, & \text{(a) \& (c) in eq.(25)}, \end{cases}$ $p_M(x_t\|\xi_t) = \sum_{y \in Y_t} p_{M_{x\|y}}(x_t\|y,\xi_t) \frac{u_t^{(y)}}{\sum_r e^{u_t^{(r)}}}$, $P(y\|x_t,\xi_t) = \frac{\sum_{\hat{y} \in Y_t} p_{M_{x\|y}}(x_t\|\hat{y},\xi_t) \frac{u_t^{(\hat{y})}}{\sum_r e^{u_t^{(r)}}} I(y-\hat{y})}{p_M(x_t\|\xi_t)}$.

Again similar to that in Sec.4.2, we can further determine the best σ_x^2, σ_ξ in a similar way as discussed in Sec.2.2. Also, we can alternatively start at values of σ_x, σ_ξ larger enough and gradually reduce it towards to zero as learning proceeds in a simulated annealing manner[36].

• **RPCL and automated model selection** In the cases that z_x is given by Choice (b) in eq.(54) for Tab.5 and in eq.(25) for Tab.6, with Y_t containing more than one indices as given in Step 1 of Item (A), we can get a unified RPCL framework for the ME model, which can be understood in a way similar to the discussions on RPCL learning in Sec.3. 2.

One typical case is that Y_t consists of the 1st winner $y_{m,t}$ and 2nd winner $y_{r,t}$ that correspond to the two largest ones of $p_{y,t}$. In this case, the algorithm in either Tab.5 or Tab.6 implements a RPCL-type learning, similar to eq.(47) and eq.(48). Generally, the algorithms in either Tab.5 or Tab.6 provide extensions of RPCL algorithms in other cases of Y_t.

When k is given large enough, the least complexity nature will be in effect during both RPCL learning and the above harmony learning with conscience. Specifically, for the alternative model, this nature will push α_y towards to 0 at some y constantly such that the corresponding expert is actually discarded. While for the original model, simlar to the case of three-layer net discussed previously, this nature is in effect in the sense that each expert becomes more specific for the task it performs with unnecessary cross-talks between experts eliminated, though it is not necessarily lead to the cases that some experts are constantly discarded.

Alternatively, by enumerating a number of k values incrementally, model selection may also be made by eq.(36) with

$$J(k) = \ln z_x + 0.5\sum_{y=1}^{k}\alpha_y \ln|\Sigma_{x|y}| - \quad (84)$$
$$\begin{cases} \frac{1}{N}\sum_{t=1}^{N}\sum_{y=1}^{k} L(y|x_t,\xi_t), & \text{(a)}, \\ \sum_{y=1}^{k}\alpha_y \ln\alpha_y - 0.5\sum_{y=1}^{k}\alpha_y \ln|\Sigma_y|, & \text{(b)}, \end{cases}$$
$$L(y|x_t,\xi_t) = P_{M_{y|x}}(y|x_t,\xi_t)\ln P_{M_y}(y|\xi_t,\theta_g),$$

with Type (a) for the original ME model and Type (b) for the alternative ME model. We can also apply Type (b) to the original model, in help of the following approximation:

$$\alpha_y = \frac{1}{N}\sum_{t=1}^{N} P_{M_{y|x}}(y|x_t,\xi_t),$$
$$\mu_y = \frac{1}{\alpha_y N}\sum_{t=1}^{N} P_{M_{y|x}}(y|x_t,\xi_t)\xi_t,$$
$$\varepsilon_{t,y} = \xi_t - \mu_y,$$
$$\Sigma_y = \frac{1}{\alpha_y N}\sum_{t=1}^{N} P_{M_{y|x}}(y|x_t,\xi_t)\varepsilon_{t,y}\varepsilon_{t,y}^T.$$

• **RBF nets** As discussed in Sec.4.3, the normalized RBF net is actually a special case of the alternative ME model under the constraint eq.(77). Thus, all the discussions on Tab.5 apply except that we simply set α_y by eq.(77) in place of the updating of α_y in Item (D).

Automated model selection is made in the sense that some Σ_y is pushed to towards 0 and thus the corresponding α_y is pushed to towards 0.

Particularly, when $\Sigma_y = \Sigma$ and thus $\alpha_y = 1/k$, as in most of existing applications, automated model selection is made in the sense that the center m_y of some basis function is driven far away from data.

Moreover, $J(k)$ in eq.(84) is simplified into

$$J(k) = \ln z_x + \ln k + 0.5\ln|\Sigma| + \frac{0.5}{k}\sum_{y=1}^{k}\ln|\Sigma_{x|y}|. \quad (85)$$

4.5. *Kernel regression and support vector machine (SVM)*

As shown in eq.(82), the normalized RBF net further degenerates into kernel regression. Here, an interesting new result is that the only unknown smoothing parameter σ^2 can be estimated by the corresponding special case of Tab.5. Further details are shown in Tab.7 and particularly pointed out in its Item (D).

Another interesting special case of the alternative ME model with the gate eq.(59) will lead us to a typical support vector machine (SVM). Again, we consider eq.(76) under the condition eq.(81) except that we let α_y being free to be determined via the harmony learning. That is, we consider the case

$$p_{M_{x|y}} = G(x|x_t,\sigma^2_{x|t}I),\ p(\xi|\phi_t) = G(\xi|\xi_t,\sigma^2_\xi I). \quad (86)$$

In this case, the regression eq.(71) of linear case (b) becomes

$$E(x|\xi) = \frac{\sum_t \alpha_t x_t e^{-0.5\frac{\|\xi-\xi_t\|^2}{\sigma^2}}}{\sum_t \alpha_t e^{-0.5\frac{\|\xi-\xi_t\|^2}{\sigma^2}}}, \quad (87)$$

which can be regarded as a modified kernel regression. Alternatively, we can rewrite it into

$$E(x|\xi) = \sum_t \alpha_t x_t K(\xi,\xi_t),$$
$$K(\xi,\xi_t) = \frac{e^{-0.5\frac{\|\xi-\xi_t\|^2}{\sigma^2}}}{\sum_t \alpha_t e^{-0.5\frac{\|\xi-\xi_t\|^2}{\sigma^2}}} \quad (88)$$

which is a typical case of the popular SVM[67], with the normalized gaussian kernel $K(\xi,\xi_t)$.

In the classical SVM, the parameters $\{\alpha_t\}$ are determined via maximizing the following constrained quadratic cost

$$J(\alpha) = \sum_t \alpha_t - 0.5\sum_{t,r}\alpha_t\alpha_t x_t x_r K(\xi_r,\xi_t),$$
$$0 \le \alpha_t \le \gamma,\ \sum_t \alpha_t x_t = 0, \quad (89)$$

for the cases that x_t takes either 0 or 1. This is a typical constrained quadratic programming problem.

After learning, the contribution corresponding to the kernel $K(\xi,\xi_t)$ is discounted or even been discarded when α_t becomes much smaller than $1/N$. Moreover, we can force $\alpha_t = 0$ when α_t is smaller than a threshold $\epsilon < 1/N$ such that the kernel $K(\xi,\xi_t)$ is completely removed, and the regression $E(x|\xi)$ is built only on a set of supporting vectors for a better generalization performance.

Tab.7 Harmony Learning Algorithms for Support Vector Machines

(A) A General Algorithm

Step 1 : For each pair x_t, ξ_t, get $\eta_{r,t}$ as shown in the Item B below, based on a subset Y_t that consists of the $\hat{k}_t \leq k$ indices that corresponds to the $\hat{k}_t$ smallest value of d_r^2 given in eq.(93). If $\hat{k}_t = 1$, Y_t contains only $r_{m,t} = arg\min_r d_r^2$.

Step 2 : update $\sigma_{x|r}^{new} = \sigma_{x|r}^{new} + \eta_0 \delta\sigma_{x|r}$

$$\delta\sigma_{x|r} = \begin{cases} \sum_{t=1}^N \delta\sigma_{x|r,t}, & \text{in batch,} \\ \delta\sigma_{x|r,t}, & \text{per sample;} \end{cases}$$

$$\delta\sigma_{x|r,t} = \eta_{r,t} \frac{d_x \sigma_{x|r}^{old\ 2} - \|x_t - x_r\|^2}{\sigma_{x|r}^{old\ 3}},$$

For the special case $\sigma_{x|r}^2 = \sigma_x^2$, instead we do

Step 2' : $\sigma_x^{new} = \sigma_x^{new} + \eta_0 \begin{cases} \sum_{r=1}^N \delta\sigma_{x|r}, \\ \delta\sigma_{x|r,t}. \end{cases}$

Step 3 : $\alpha_t = e^{c_t} / \sum_r e^{c_r}$, $c_r^{new} = c_r^{old} + \eta_0 \delta c_r$,

$$\delta c_r = \begin{cases} \sum_{t=1}^N \delta c_{r,t}, & \text{(a) in batch,} \\ \delta c_{r,t}, & \text{(b) per sample;} \end{cases}$$

$$\delta c_{r,t} = \eta_{r,t} - \alpha_r \sum_t \eta_{r,t},$$

update $\sigma_\xi^{new} = \sigma_\xi^{new} + \eta_0 \delta\sigma_\xi$

$$\delta\sigma_\xi = \begin{cases} \sum_{r=1}^N \sum_{t=1}^N \delta\sigma_{\xi|r,t}, & \text{in batch,} \\ \delta\sigma_{\xi|r,t}, & \text{per sample;} \end{cases}$$

$$\delta\sigma_{\xi|r,t} = \eta_{r,t} \frac{d_\xi \sigma_{\xi|r}^{old\ 2} - \|\xi_t - \xi_r\|^2}{\sigma_{\xi|r}^{old\ 3}}.$$

(B) $\eta_{r,t} = \frac{P_{M_{y|x}}(r|x_t,\xi_t)}{N} - \gamma_t P(r|x_t,\xi_t)$,

$$P_{M_{y|x}}(r|x_t,\xi_t) = \begin{cases} I(r - r_{m,t}), & \text{(a)} \\ \frac{G(x_t|x_r,\sigma_{x|r}^2 I) G(\xi_t|\xi_r,\sigma_\xi^2 I)\alpha_r}{\sum_{j=1}^N G(x_t|x_j,\sigma_{x|j}^2 I) G(\xi_t|\xi_j,\sigma_\xi^2 I)\alpha_j}, & \text{(b).} \end{cases}$$

$$\gamma_t = \begin{cases} \frac{p_M(x_t,\xi_t)}{\sum_{t=1}^N p_M(x_t,\xi_t)}, & \text{Choice (b) in eq.(25),} \\ 0, & \text{(a) \& (c) in eq.(25),} \end{cases}$$

$p_M(x_t,\xi_t) = \sum_{r\in Y_t} G(x_t|x_r,\sigma_{x|r}^2 I) G(\xi_t|\xi_r,\sigma_\xi^2 I)\alpha_r$,

$P(r|x_t,\xi_t) = \sum_{j\in Y_t} \frac{G(x_t|x_j,\sigma_{x|j}^2 I) G(\xi_t|\xi_j,\sigma_\xi^2 I)\alpha_j I(r-j)}{p_M(x_t,\xi_t)}$.

(C) Smoothing Parameter in Kernel Regression

In Step 3, fix $\alpha_t = 1/N$ without updating it.

Now, in help of the algorithms in Tab.5, we can alternatively learn the parameters $\{\alpha_t\}$ as well as the smoothing parameter σ^2, via the harmony learning eq.(54). To get a further insight, we put $P_{M_{y|x}}(y|x_t,\xi_t)$ in eq.(56) into eq.(54) and simply let $z_x = 1$, resulting in

$$\begin{aligned} H(\theta,k) &= \textstyle\sum_r \alpha_r F_0(x_r,\xi_r) \ln\alpha_r \\ &+ \textstyle\sum_r \alpha_r F_1(x_r,\xi_r), \\ F_0(x_r,\xi_r) &= \textstyle\sum_t \frac{p_{M_{x|y}}(x_t|r,\xi_t) p(\xi_t|\phi_r)}{N p_M(x_t,\xi_t)}, \\ F_1(x_r,\xi_r) &= \textstyle\sum_t \frac{p_{M_{x|y}}(x_t|r,\xi_t) p(\xi_t|\phi_r)}{N p_M(x_t,\xi_t)} \times \\ &\ln[p_{M_{x|y}}(x|y,\xi) p(\xi|\phi_y)]. \end{aligned} \tag{90}$$

Then, from $\sum_r \alpha_r = 1$ we insert $\alpha_r = 1 - \sum_{j\neq r} \alpha_j$ in eq.(90) and get

$$\begin{aligned} H(\theta,k) &= -\textstyle\sum_r \sum_{j\neq r} \alpha_j F_0(x_r,\xi_r) \ln\alpha_r \\ &+ \textstyle\sum_r F_0(x_r,\xi_r) \ln\alpha_r + \sum_r \alpha_r F_1(x_r,\xi_r), \\ 0 \leq \alpha_t &\leq 1, \sum_t \alpha_t - 1 = 0. \end{aligned} \tag{91}$$

On the other hand, we rewrite eq.(89) into

$$\begin{aligned} J(\alpha) &= -0.5 \textstyle\sum_{t,r} \alpha_t \alpha_r F(x_t,x_r,\xi_r,\xi_t) + \sum_t \alpha_t, \\ F(x_t,x_r,\xi_r,\xi_t) &= x_t x_r K(\xi_r,\xi_t), \\ 0 \leq \alpha_t &\leq \gamma,\ \textstyle\sum_t \alpha_t x_t = 0. \end{aligned} \tag{92}$$

It can be observed that eq.(91) and eq.(92) are , qualitatively quite similar, though they are not exactly the same. Specifically, it follows from eq.(91) that the harmony learning is a constrained sub-quadratic programming problem in the sense α_r of the order one is replaced by $\ln\alpha_r$ that is lower than order one. This insight provides a justification for using the algorithms in Tab.5 for learning SVM of type eq.(88) with the normalized gaussian kernel $K(\xi,\xi_t)$.

Moreover, considering other cases of eq.(59) or even eq.(74), we may also learn support vector machines with other normalized kernels by the harmony learning.

Furthermore, we can simplify Tab.5 into Tab.7. Specifically, the task of finding $\hat{k}_t \leq k$ different indices of $\{1,\cdots,k\}$ can be simplified into finding the $\hat{k}_t$ smallest value of d_r^2 as follows

$$\begin{aligned} (a)\quad d_r^2 &= \frac{\|x_t - x_r\|^2}{\sigma_{x|r}^2} + d_x \ln\sigma_{x|r}^2 + \frac{\|\xi_t - \xi_r\|^2}{\sigma_\xi^2}, \\ (b)\quad d_r^2 &= \frac{\|x_t - x_r\|^2}{\sigma_x^2} + \frac{\|\xi_t - \xi_r\|^2}{\sigma_\xi^2}, \end{aligned} \tag{93}$$

where the choice (a) is for $p_{M_{x|y}} = G(x|x_t,\sigma_{x|t}^2 I)$ in eq.(86) and the choice (b) is for its special case that $\sigma_{x|t}^2 = \sigma_x^2$. In Item (A) of Tab.7, the set Y_t is obtained via d_r^2 after comparing all $d_t^2, t = 1,\cdots,N$.

Thus, the learning by algorithms in Tab.7 is of a batch manner in nature. However, we can also alternatively update $\alpha_t, \sigma_x^2, \sigma_\xi^2$ per sample as provided in Tab.7 too.

We further consider the simplest case in Tab.7 that $\gamma_t = 0$ in Item B and $\hat{k}_t = 1$ in Step 1. In this case, we get the following competitive learning algorithm for the SVM learning on eq.(88):

$$r_m = arg\min_r d_r^2,\ d_r^2 \text{ is given in eq.(93)},$$
$$c_y^{new} = c_y^{old} + \eta_0 \begin{cases} 1-\alpha_y, & y = r_m, \\ -\alpha_y, & y \neq r_m; \end{cases}$$
$$\sigma_{x|r}^{new} = \sigma_{x|r}^{new} + \eta_0 \begin{cases} \frac{d_x \sigma_{x|r}^{old\ 2} - ||x_t - x_r||^2}{\sigma_{x|r}^{old\ 3}}, & y = r_m, \\ 0, & y \neq r_m; \end{cases}$$
$$\sigma_\xi^{new} = \sigma_\xi^{new} + \eta_0 \begin{cases} \frac{d_\xi \sigma_{\xi|r}^{old\ 2} - ||\xi_t - \xi_r||^2}{\sigma_{\xi|r}^{old\ 3}}, & y = r_m, \\ 0, & y \neq r_m. \end{cases}$$

For the special case $\sigma_{x|t}^2 = \sigma^2$, we simply replace the above $\sigma_{x|r}^{new}, \sigma_{x|r}^{old}$ by $\sigma_x^{new}, \sigma_x^{old}$.

5. Conclusion

The BYY harmony learning on systems with discrete inner-representations has been systematically introduced. For unsupervised learning on Gaussian mixture, we not only revisited the previous obtained criteria for selecting the number of Gaussians [80], but also got new regularization techniques and a unified RPCL framework that performs parameter learning with automated model selection. Moreover, a by-product is also obtained for determining a set of 'supporting vectors' for Parzen window density estimation. Similar new results are also obtained for supervised learning on the original ME model, the alternative ME model and radial basis function (RBF) nets, respectively. Moreover, three layer net is benefited with a regularized ML learning, a new criterion for selecting the hidden unit number, and a family of EM-like algorithms that combines harmony learning with new regularization techniques. Furthermore, an easily implemented approach is given for determining the smoothing parameter in the kernel regression, and an alternative approach is provided to select supporting vectors in the popular supporting vector machines for a better generalization.

References

1. S.C.Ahalt, et al, "Competitive learning algorithms for vector quantization", *Neural Networks*, **3**, 277-291 (1990).
2. H.Akaike, "A new look at the statistical model identification", *IEEE Tr. Automatic Control*, **19**, 714-723 (1974).
3. G.H. Ball & D.J. Hall, "ISODATA: A novel method of data analysis and pattern classification', *Tech. Rep. No. AD 699616*, Stanford Research International (1965).
4. S. A. Billings & G. L. Zheng, "Radial basis function network configuration using genetic algorithms", *Neural Networks*, **8**, 877-890 (1995).
5. C.M Bishop, "Training with noise is equivalent to Tikhonov regularization", *Neural Computation 7*, 108-116 (1995).
6. A. G. Bors and I. Pitas, "Median radial basis function neural network", *IEEE Trans. on Neural Networks*, **7**, 1351-1364 (1996).
7. A. G. Bors & I. Pitas, "Object classification in 3-D images using alpha-trimmed mean radial basis function network," *IEEE Trans. on Image Process*, **8**, 1744-1756 (1999).
8. H. Bozdogan, " Model Selection and Akaike's Information Criterion: The general theory and its analytical extension", *PSYCHOMETRIKA*, **52**, 345-370 (1987).
9. H. Bozdogan, " Mixture-Model Cluster analysis using model selection criteria and a new information measure of complexity", *Proc. 1st US/Japan Conf. on the Frontiers of Statistical Modeling*, **2**, 69-113 (1994).
10. D.S.Broomhead & D.Lowe, "Multivariable functional interpolation and adaptive networks", *Complex Systems* **2**, 321-323 (1988).
11. P. R. Chang & W. H. Yang, "Environment-adaptation mobile radio propagation prediction using radial basis function neural networks", *IEEE Trans. on Vehicular Technology*, **46**, 155-160 (1997).
12. Y. M. Cheung & L. Xu, "A RPCL-based approach for Markov model identification with unknown state number," *IEEE Signal Processing Letters*, **7**, 284-287 (2000).
13. Y. M. Cheung, et al, "A RPCL-CLP architecture for financial time series forecasting," in *Proc. of 1995 IEEE ICNN*, **2**, 829-832 (1995).
14. E. Chiarantoni, et al, "Scene segmentation in video sequences by a RPCL neural network," *Proc. of 1998 IEEE WCCI*, **3**, 1877-1882 (1998).
15. P. A. Devijver & J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall (1982).
16. L.Devroye, "On the almost everywhere convergence of nonparametric regression function estimates", *The Annals of Statistics*, **9**, 1310-1319 (1981).
17. L. Devroye, *A Course in Density Estimation*, Birkhauser, Boston (1987).
18. L. Devroye, et al, *A Probability Theory of Pattern Recognition*, Springer (1996).
19. D.Desieno, "Adding a conscience to competitive learn-

ing", *Proc. IEEE Intl. Conf. on Neural Networks*, **I**, 117-124 (1988).
20. A.P.Dempster, et al, "Maximum- likelihood from incomplete data via the EM algorithm", *J. of Royal Statistical Society*, **B39**, 1-38 (1977).
21. R.O.Duda and P.E.Hart, *Pattern classification and Scene analysis*, Wiley (1973).
22. H.P.Friedman & J.Rubin, " On Some invariant criteria for grouping data", *J. Amer. Statis., Assoc*, **62**, 1159-1178 (1967).
23. H. Furukawa, et al, "A systematic method for rational definition of plant diagnostic symptoms by self-organizing neural networks", *Neurocomputing*, **13**, 171-183 (1996).
24. F. Girosi, et al, "Regularization theory and neural architectures", *Neural Computation*, **7**, 219-269 (1995).
25. W.Greblicki, et al, "Distribution-free consistency of Kernel regression estimate", *The Annals of Statistics*, **12**, 1570-1575 (1984).
26. S. Grossberg, "Competitive learning: from interactive activation to adaptive resonance", *Cognitive Science*, **11**, 23-63(1987).
27. R. Hecht-Nielsen, "Counterpropagation networks," *Appl. Opt.*, **26**, 4979-4984 (1987).
28. G. E. Hinton & R.S. Zemel, "Autoencoders, minimum description length and Helmholtz free energy", *Advances in NIPS*, **6**, 3-10 (1994).
29. G.E.Hinton, et al, "Modeling the manifolds of images of handwritten digits", *IEEE Trans. Neural Networks*, **8**, 65-74 (1997).
30. R.A.Jacobs, et al, "Adaptive mixtures of local experts", *Neural Computation*, **3**, 79-87 (1991).
31. , A.K.Jain & R.C.Dubes, *Algorithm for Clustering Data*, Prentice-Hall (1988).
32. R.D.Jones et al, "Information theoretic derivation of network architecture and learning algorithms", *Proc. of IIJCNN91-Seattle*, **II**, 473-478 (1991).
33. M. I. Jordan & R.A.Jacobs, " Hierarchical mixtures of experts and the EM algorithm", *Neural Computation*, **6**, 181-214 (1994).
34. M. I. Jordan & L. Xu, "Convergence results for the EM approach to mixtures of experts", *Neural Networks*, **8**, 1409–1431 (1995).
35. I. King, et al, "Using rival penalized competitive clustering for feature indexing in Hong Kong's textile and fashion image database," *Proc. of 1998 IEEE WCCI*, **1**, 237-240 (1998).
36. S.Kirkpatrick, et al, "Optimization by simulated annealing", *Science*, **220**, 671-680 (1983).
37. T.K. Lau & I. King, "Performance analysis of clustering algorithms for information retrieval in image databases," *Proc. of 1998 IEEE World Congress on Computational Intelligence*, **2**, 932-937 (1998).
38. J. Lee, et al, "A practical radial basis function equalizer," *IEEE Trans. on Neural Networks*, **10**, 450-455 (1999).
39. X. Q. Li &. King, "Regression analysis for rival penalized competitive learning binary tree," *Proc. of IJCNN2000*, **6**, 290-295 (2000).
40. R. Li, et al, "Fast image vector quantization using a modified competitive learning neural network approach" *Intl J. of Imaging Systems and Technology*, **8**, 413-418 (1997).
41. D.J.C. Mackey, " Bayesian Interpolation", *Neural Computation*, **4**, 488-472 (1992),.
42. D.J.C. Mackey, "A practical Bayesian framework for backpropagation", *Neural Computation*, **4**, 415-447 (1992).
43. J. Makhoul, et al, "Vector quantization in speech coding," *Proc. IEEE*, **73**, 1551-1558 (1985).
44. A. Marazzi, et al, "Automatic selection of the number of clusters in multidimensional data problems", *Proc. of Intl. Conf. on Image Processing*, **3**, 631-634 (1996).
45. P. McCullagh & J.A. Nelder, *Generalized Linear Models*, Chapman and Hall (1983).
46. G.J. McLachlan & K.E. Basford, *Mixture Models: Inference and Application to Clustering*, Dekker (1988).
47. G. W. Millgan, " A monte carlo study of thirty internal criterion measures for cluster analysis", *PSYCHOMETRIKA*, **46**, 187-199 (1985).
48. G. W. Millgan & M.C. Copper, " An examination of procedures for determining the number of clusters in a data set", *PSYCHOMETRIKA*, **50**, 159-179 (1985).
49. J.Moody & J.Darken, "Fast learning in networks of locally-tuned processing units", *Neural Computation*, **1**, 281-294 (1989).
50. N. Nasrabadi & R. A. King, "Image coding using vector quantization: a review," *IEEE Trans. Commun.*, **36**, 957-971 (1988).
51. S.J.Nowlan, "Max likelihood competition in RBF networks", *Tech. Rep. CRG-Tr-90-2, Dept. of Computer Sci., U. of Toronto* (1990).
52. T.Poggio & F.Girosi, "Networks for approximation and learning", *Proc. of IEEE*, **78**, 1481-1497 (1990).
53. M.J.D.Powell, "Radial basis functions for multivariable interpolation: a review", eds, J.C.Mason and M.G.Cox, *Algorithms for Approximation*, Clarendon Press, Oxford (1987).
54. D.F.Specht, "Probabilistic neural networks", *Neural Networks*, **3**, 109-118 (1990).
55. V. Ramamurti & J. Ghosh, "Regularization and error bars for the mixture of experts network", *Proc. of IEEE ICNN97", 221-225 (1997.*
56. F.Ronsenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Washington D.C.: Spartan Books (1962).
57. D.E.Rumelhart, G.E.Hinton & R.J.Williams, "Learning internal representations by error propagation", *Parallel distributed processing*, **1**, MIT press (1986).
58. R.A. Redner & H.F. Walker, "Mixture densities, maximum likelihood, and the EM algorithm", *SIAM Review*, **26**, 195-239 (1984).
59. J.Rissanen, *Stochastic Complexity in Statistical Inquiry*, World Scientific: Singapore (1989).

60. J. Rissanen et al, "Unsupervised classification with stochastic complexity in multivariate statistical modeling", *Proc. The 1st US/Japan Conf. on the Frontiers of Statistical Modeling: An In formational Approach*, (1994).
61. D. E. Rumelhart & D. Zipser, "Feature discovery by competitive learning," *Cognitive Science*, **9**, 75-112 (1985).
62. I. Rivals & L. Personnaz, "On Cross Validation for Model Selection", *Neural Computation*, **11**, 863-870 (1999).
63. A. Roy, et al, "A neural-network learning theory and a polynomial time RBF algorithm", *IEEE Trans. on Neural Networks*, **8**, 1301-1313 (1997).
64. A.J. Scott & M.J. Symons, "Clustering methods based on likelihood ratio criteria", *Biometrics*, **27**, 387-397 (1971).
65. M. Stone, "Cross-validation: A review", *Math. Operat. Statist.* , 9, 127-140 (1978).
66. A.N.Tikhonov & V.Y. Arsenin, *Solutions of Ill-posed Problems*, V.H. Winston and Sons (1977).
67. V.N.Vapnik, *The Nature Of Statistical Learning Theory*, Springer-Verlag (1995).
68. L. Xu, " BYY Harmony Learning, Independent State Space and Generalized APT Financial Analyses", to appear on *IEEE Trans. on Neural Networks*, (2001).
69. L. Xu, " Temporal BYY Learning for State Space Approach, Hidden Markov Model and Blind Source Separation", *IEEE Trans on Signal Processing*, **48**, 2132-2144 (2000). A part of its preliminary version on *Proc. IJCNN99*, **2**, 949-954 (1999).
70. L. Xu, " Best Harmony Learning ", *Proc. IDEAL2000: Lecture Notes in Computer Science*, **1983**, Springer-Verlag, 116-125(2000).
71. L. Xu, "Best Harmony Learning for BYY Σ-Π Factor Systems", *Proc. 7th Intl. Conf. On Neural Information Processing* , bf 1, 548-558(2000).
72. L. Xu, "BYY System and Theory for Statistical Learning: Best Harmony, Data Smoothing, and Model Selection", *Proc. of 1999 Chinese Conf. on Neural Networks and Signal Processing*, Nov., Shantou, China, (12-29) 1999.
73. L. Xu, "Bayesian Ying-Yang Supervised Learning, Modular Models, and Three Layer Nets", *Proc. 1999 IJCNN*, **1**, 540-545(1999).
74. L. Xu, "RBF Nets, Mixture Experts, and Bayesian Ying-Yang Learning", *Neurocomputing*, **19**, 223-257 (1998).
75. L. Xu, " Bayesian Kullback Ying-Yang Dependence Reduction Theory", *Neurocomputing*, **22**, 81-112 (1998).
76. L. Xu, "Rival penalized competitive learning, finite mixture, and multisets clustering," *Proc. of 1998 IEEE IJCNN*, **3**, 251-2530 (1988).
77. L. Xu, " Bayesian Ying-Yang Learning Theory For Data Dimension Reduction and Determination ", *J. of Computational Intelligence in Finance*, **6**, 6-18 (1998).
78. L. Xu, C.C.Cheung, & S.-I.Amari, " Learned Parametric Mixture Based ICA Algorithm ", *Neurocomputing*, **22**, 69-80 (1998).
79. L. Xu, "Bayesian Ying-Yang Learning Based ICA Models" , *Proc. 1997 IEEE Workshop on Neural Networks for Signal Processing*, 476-485 (1997).
80. L. Xu, "Bayesian Ying-Yang Machine, Clustering and Number of Clusters", *Pattern Recognition Letters*, **18**, 1167-1178 (1997).
81. L. Xu, "Bayesian-Kullback YING-YANG Machines for Supervised Learning", *Proc. of 1996 World Congress On Neural Networks*, 193-200 (1996).
82. L. Xu, & M. I. Jordan, "On Convergence Properties of the EM Algorithm for Gaussian Mixtures", *Neural Computation*, **8**, 129-151 (1996).
83. L. Xu, "Bayesian-Kullback YING-YANG Machine: Reviews and New Results", *Progress in Neural Information Processing: Proc. ICONIP96*, Springer-Verlag, 59-67 (1996).
84. L. Xu, "Vector Quantization, Cluster Number Selection and The EM Algorithms", *Proc. of 1995 IEEE Intl Conf. on Neural Networks and Signal Processing*, **II**, 1499-1502 (1995).
85. L. Xu, "A Unified Learning Scheme: Bayesian-Kullback YING-YANG Machine", *Advances in Neural Information Processing Systems*, **8**, 444-450 (1996). A part of its preliminary version on *Proc. ICONIP95-Peking*, 977-988(1995).
86. L. Xu, M.I. Jordan, & G.E. Hinton, "An Alternative Model for Mixtures of Experts", *Advances in Neural Information Processing Systems*, **7**, 633-640 (1995). Its preliminary version on *Proc. of WCNN'94*, **2**, 405-410 (1994).
87. L. Xu "A Unified Learning Framework: Multisets Modeling Learning", *Proc. of 1995 World Congress on Neural Networks*, **1**, 35-42 (1995). A part of its preliminary version on *Proc. 1994 IEEE Intl. Conf. on Neural Networks* , **1**, 315-320 (1994).
88. L. Xu, A.Krzyzak, & A.L.Yuille, "On Radial Basis Function Nets and Kernel Regression: Statistical Consistency, Convergence Rates and Receptive Field Size", *Neural Networks*, **7**, 609-628 (1994).
89. L. Xu, A.Krzyzak, & E.Oja, " Rival Penalized Competitive Learning for Clustering Analysis, RBF net and Curve Detection", *IEEE Trans. on Neural Networks*, **4**, 636-649 (1993). Its early version on *Proc. of 11th Intl. Conf. on Pattern Recognition*, **1**, 672-675 (1992).
90. L. Xu, "Least mean square error reconstruction for self-organizing neural-nets", *Neural Networks*, **6**, 627-648 (1993). Its early version on *Proc. 1991 Intl. Joint Conf. on Neural Networks (IJCNN91'Singapore)*, 2363-2373 (1991).
91. G. L. Zheng & S. A. Billings, "Radial basis function network configuration using mutual information and the orthogonal least squares algorithm", *Neural Networks*, **9**, 1619-1637(1996).