Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

ELSEVIER

Neurocomputing 51 (2003) 277–301

NEUROCOMPUTING

www.elsevier.com/locate/neucom

BYY learning, regularized implementation, and model selection on modular networks with one hidden layer of binary units ☆

Lei Xu

Department of Computer Science Engineering, Chinese University of Hong Kong Shatin, N.T., Hong Kong, People's Republic of China

Received 16 July 2001; accepted 8 May 2002

Abstract

The BYY learning has been extended to a modular system, with developments on not only regularized implementation via either normalization or data smoothing, but also the least complexity based model selection. Moreover, both unsupervised and supervised learning have been specifically investigated on networks with one hidden layer of binary units. Adaptive EM-like learning algorithms are provided for implementing regularized learning with either automatic model selection during parameter learning or post-learning selection criteria for both the number of individual nets and the number of hidden units. Furthermore, discussions are made on application of rule extraction for tackling the paradox between conflict and redundancy.

Keywords: BYY modular system; Harmony learning; Binary hidden layer; Mixture-of-experts; Hidden unit number; Regularization; Model selection; Rule extraction; Conflict and redundancy

1. Introduction

The tasks of pattern recognition, feature extraction, independent data analysis and statistical regression as well as rule extraction all share a common key point that performs an expected input–output mapping. The specific type of mapping varies for different

☆ The work described in this paper was fully supported by a grant from the Research Grant Council of the Hong Kong SAR (Project No: CUHK 4383/99E).
E-mail address: lxu@cse.cuhk.edu.hk (L. Xu).

0925-2312/03/$ - see front matter
PII: S0925-2312(02)00622-7

network structures and various practical applications. One widely used structure is forward network with one hidden layer of binary units or approximately sigmoid units, trained via either supervised learning or unsupervised learning.

By supervised learning, the most popular example is the one trained by the back-propagation technique [28]. Another popular example is the mixture-of-expert model [12–14] which implements the entire input–output mapping by a number of local individual networks or called experts that are combined via a probabilistic controlling of the so called gating net. The EM algorithm [9,24] can be used to separate the learning of experts and the gating net. Then, each of them can be further learned by back-propagation. Moreover, an alternative gating model is proposed such that the entire learning can be made in the maximum likelihood sense by the EM algorithm in the case that each expert is described by a Gaussian with a linear regression [41,23,36]. Furthermore, adaptive EM algorithms have been proposed for fast learning of both the original and alternative mixture-of-expert models in help of the so called coordinated competition [36].

By unsupervised learning, one typical example is the multiple cause model that regards the observed samples being generated from binary hidden factors with mutually independent bits [29,8], trained by either cost minimization or maximum likelihood. One other example is the auto-association network that is trained by back-propagation via simply copying input as the desired output [5]. Another example is the *least mean square error reconstruction (LMSER)* learning [34] that was found experimentally to perform ICA with promising results [15]. Recently, it has been further shown in [38] that LMSER actually implements not simply ICA but a so called principal ICA that corresponds to the extension of PCA to ICA.

Regularization and model selection are two major issues that affect the performance of a network learned on a training set of finite number of samples. For a network with one hidden layer, the generalization performance is prone to the number of hidden units. Many studies have been made. On one hand, several theories and heuristic techniques for regularization have been proposed to impose constraints on the parameters in networks such that it is effectively equivalent to reducing the number of hidden units. Typical theories for this purpose include Tikhonov-type regularization theory [30,4], Bayesian theory [21] and MDL theory [25,26], 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, by enumerating a number of structures with different number of hidden units. Typical model selection theories include Bayesian theory [21], MDL [25,26,11], VC dimension based generalization theory [31], cross validation [27] and AIC [1] as well as its extensions [6]. However, these theories are usually difficult or expensive in computational implementation.

The Bayesian Ying-Yang (BYY) learning was proposed as a unified statistical learning framework firstly in 1995 and systematically developed in past years [38]. From the perspective of general learning framework, the BYY harmony 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 class of criteria for model selection, and a new family of algorithms that perform parameter learning with automated model selection. From the perspective of specific learning paradigms, the BYY

learning with specific structures applies to unsupervised learning, supervised learning, and state space approach for temporal modeling, with a number of new results. The details are referred to [38].

The model selection ability of the BYY learning can also be further understood from an information transfer perspective. As discussed in [39], the BYY harmony learning shares the common spirit of the minimum message length (MML) [32,33] or the minimum description length (MDL) [25,26], but with two key differences that make the BYY harmony learning avoid the difficulty of encoding the parameter set, as usually encountered by the MML/MDL approach. The relations of the BYY harmony learning to certain existing learning approaches have also been discussed in [39]. Moreover, the experimental results that demonstrate how model selection by the BYY learning works have been made in comparison with some existing approaches on several learning tasks, including the number of clusters on Guassian mixture and clustering analyses in [35,17], the hidden unit selection of three layer nets [18,22], the number of experts in the mixture-of-experts model [19], local PCA analyses [20], and temporal factor analysis [7].

In Section 2 of this paper, after introducing the fundamentals of BYY harmony learning, we further extend the harmony learning to the BYY modular system. In Sections 3 and 4, respectively, both unsupervised and supervised learning have been specifically investigated on networks with one binary hidden layer. Adaptive EM-like learning algorithms are provided for implementing regularized learning with either automatic model selection during parameter learning or post-learning selection criteria for the number of individual nets and the number of hidden units. Finally, discussions are made in Section 5 on applications of rule extraction for tackling the paradox between conflict and redundancy.

2. Bayesian Ying-Yang system and harmony learning

As shown in Fig. 1, we consider a world $\mathbf{X}$ with each object in observation represented by a stochastic notation $\mathbf{x} \in \mathbf{X}$. Corresponding to each $\mathbf{x}$, there is an inner representation $\mathbf{y} \in \mathbf{Y}$ in the representation domain $\mathbf{Y}$ of a learning system. We consider the joint distribution of $\mathbf{x}, \mathbf{y}$, which can be understood from two complement perspectives.

On one hand, we can interpret that each $\mathbf{x}$ is generated (or called reconstructed/decoded) from an invisible inner representation $\mathbf{y}$ via a backward path distribution $q(\mathbf{x}|\mathbf{y})$ or called a generative model by which $\mathbf{x}$ is generated from an inner distribution $q(\mathbf{y})$ in a structure that is designed according to the learning tasks. On the other hand, we can interpret that each $\mathbf{x}$ is mapped (or called encoded/recognized) into an invisible inner representation $\mathbf{y}$ via a forward path distribution $p(\mathbf{y}|\mathbf{x})$ to match the inner density $q(\mathbf{y})$ in a pre-specified structure.

The two perspectives reflect the two types of Bayesian decomposition of the joint density $q(\mathbf{x}|\mathbf{y})q(\mathbf{y}) = q(\mathbf{x},\mathbf{y}) = p(\mathbf{x},\mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$ on $X \times Y$. Without any constraint, the two decompositions should be theoretically identical. However, in a real consideration, the four components $p(\mathbf{y}|\mathbf{x})$, $p(\mathbf{x})$, $q(\mathbf{x}|\mathbf{y})$, $q(\mathbf{y})$ should be subject to certain

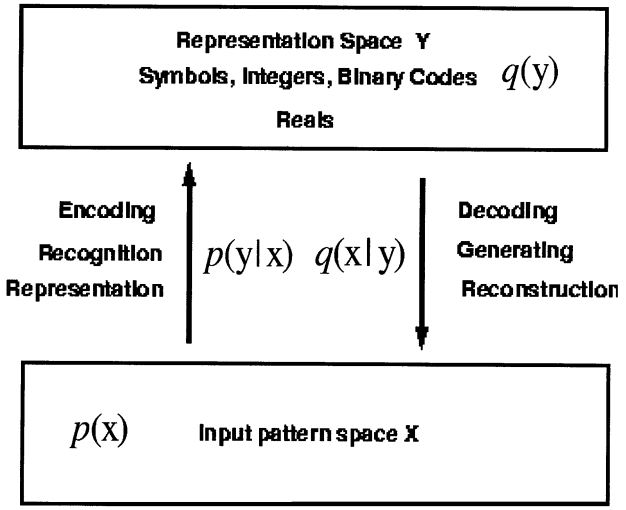

Fig. 1. Bayesian Ying-Yang system.

structural constraints. Thus, we usually have two different but complementary Bayesian representations:

$$p(\mathbf{x},\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}), \quad q(\mathbf{x},\mathbf{y}) = q(\mathbf{x}|\mathbf{y})q(\mathbf{y}), \tag{1}$$

which compliments to the famous Chinese ancient Ying-Yang philosophy [38] with $p(\mathbf{x},\mathbf{y})$ called Yang machine that consists of the observation space (or called Yang space) by $p(\mathbf{x})$ and the forward pathway (or called Yang pathway) by $p(\mathbf{y}|\mathbf{x})$, and with $q(\mathbf{x},\mathbf{y})$ called Ying machine that consists of the invisible state space (or Ying space) by $q(\mathbf{y})$ and the Ying (or backward) pathway by $q(\mathbf{x}|\mathbf{y})$. Such a pair of Ying-Yang models is called *Bayesian Ying-Yang* (*BYY*) *system*. The task of learning on a BYY system consists of specifying all the aspects of the system, which is implemented in three steps.

First, on a set X of samples from the observed world $\mathbf{X}$, the distribution $p(\mathbf{x})$ can be obtained by a non-parametric Parzen window estimate [10]:

$$p_{h_x}(x) = \frac{1}{N}\sum_{t=1}^{N} G(x|x_t, h_x^2 I), \tag{2}$$

where, and throughout this paper, $G(x|m,\Sigma)$ denotes a Gaussian density with mean vector m and covariance matrix Σ. Particularly, $p_{h_x}(x) = p_0(x)$ becomes the empirical density when $h_x = 0$.

Second, we need to design each structure of $p(\mathbf{y}|\mathbf{x})$, $q(\mathbf{x}|\mathbf{y})$, $q(\mathbf{y})$. The details will be discussed in the subsequent sections.

In a narrow sense, our learning task is mainly the *third* step that has two subtasks. One is called parameter learning for determining a value of the set θ that consists of all the unknown parameters in $p(\mathbf{y}|\mathbf{x})$, $q(\mathbf{x}|\mathbf{y})$, $q(\mathbf{y})$ as well as h_x (if any). The

other subtask is to select the scale of the representation domain **Y**, which is featured by a set **k** of several natural numbers. This subtask is usually called *model selection* since a collection of specific BYY systems with different values on **k** corresponds to a family of specific models that share a same system configuration but in different scales.

Our *fundamental learning principle* is to make the Ying machine and Yang machine be best harmony in a twofold sense:

- The difference between the two complementary Bayesian representations in Eq. (1) should be minimized.
- The resulting BYY system should be of the least complexity.

Its implementation is made via maximizing a so called harmony measure. Specially, we have two ways. One is making the above first point via minimizing the Kullback divergence and then making the above second point via maximizing the harmony measure [37]. The other way is to make both points jointly via maximizing the harmony measure [38]. This paper will focus on further studies on the second way.

The simplest harmony measure is the following cross entropy:

$$H(p\|q) = \sum_{t=1}^{N} p_t \ln q_t, \tag{3}$$

for two discrete densities both $p(u), q(u)$ in the form of

$$q(u) = \sum_{t=1}^{N} q_t \delta(u - u_t), \quad \sum_{t=1}^{N} q_t = 1, \tag{4}$$

with $\delta(u)$ being a δ-function.

Maximizing this $H(p\|q)$ has two interesting natures:

- *Matching nature* with p fixed, $\max_q H(p\|q)$ pushes q towards

$$q_t = p_t \quad \text{for all } t. \tag{5}$$

- *Least complexity nature* $\max_p H(p\|q)$ with q fixed pushes p towards

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

or equivalently $p_\tau = 1$, and $p_t = 0$ for other t, which is of the least complexity from the statistical perspective [38]. Thus, the maximization of the functional $H(p\|q)$ indeed implements the above harmony learning principle mathematically.

When $p(u), q(u)$ are discrete or continuous densities, a typical form of the harmony measure is

$$H(p\|q) = \int p(u) \ln q(u)\, \mathrm{d}u - \ln z_q, \tag{7}$$

where z_q depends on how $p(u)$ is estimated. For $p(u) = p_h(u)$ by Eq. (2), we have

$$z_q = \begin{cases} \sum_{t=1}^{N} q(u_t) & \text{(a) } h_x = 0, \\ \sum_{t=1}^{N} p_h(u_t) & \text{(b) } h_x \text{ to be learned,} \end{cases} \tag{8}$$

under a mild assumption $\sum_{t=1}^{N} q(u_t) = \sum_{t=1}^{N} p(u_t)$.

It can be observed that Eq. (7) degenerates to Eq. (3) when $q(u)$, $p(u)$ are discrete as in Eq. (4). Moreover, for the case (a), $H(p\|q)$ will degenerate to the likelihood function either when $q(u)$ is a discrete density with $z_q = \delta_u(0)$ or when $q(u)$ is a continuous density but we make a crude approximation $z_q = 1$. In other words, it becomes equivalent to the maximum likelihood (ML) learning.

The fact that maximizing a cross entropy like cost $\max_\theta \int p_0(u) \ln q(u)\, \mathrm{d}u$ leads to the ML learning is well known in the literature. However, what is new is that a general z_q by Eq. (8) introduces a regularization to ML learning via either a de-learning in the choice (a) or a Tikhonov-type in the choice (b) [38].

More interestingly, though the least complexity nature Eq. (6) is regarded as being useless in the conventional literature, it takes an essential role in learning on the *Bayesian Ying-Yang (BYY) system*, with $p(u) = p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$, $q(u) = q(\mathbf{x}, \mathbf{y}) = q(\mathbf{x}|\mathbf{y})q(\mathbf{y})$ inserted in Eq. (7). Instead of completely fixed as $p(u) = p_h(u)$ by a training set of samples, the Yang machine consists of not only $p(\mathbf{x}) = p_{h_x}(x)$ that is fixed from the training set, but also $p(\mathbf{y}|\mathbf{x})$ that is free or at least not completely fixed and thus will be pushed towards its least complexity form by the least complexity nature Eq. (6). In turn, the matching nature of harmony learning will further push $q(\mathbf{x}|\mathbf{y})$ and $q(\mathbf{y})$ towards their corresponding least complexity forms. In other words, the least complexity nature in a BYY system makes model selection possible.

Mathematically, the harmony learning is described as

$$\max_{\theta, \mathbf{k}} H(\theta, \mathbf{k}), \; H(\theta, \mathbf{k}) = H(p\|q), \tag{9}$$

where $H(p\|q)$ is obtained with $p = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$, $q = q(\mathbf{x}|\mathbf{y})q(\mathbf{y})$ in Eq. (7).

3. A new development on BYY harmony learning

3.1. BYY harmony learning with modular representation domains

The nature of a BYY system is featured by the structure of $q(\mathbf{y})$, which depends on the complexity of the representation domain $\mathbf{Y}$ that depends on the complexity of the learning task in consideration as well as the world $\mathbf{X}$ we observe.

The most simplified world $\mathbf{X}$ consists of only a single object which is observed via a set X of samples with each sample $\mathbf{x} = [x^{(1)}, \ldots, x^{(d)}]^{\mathrm{T}}$ from a same underlying probability density function (pdf) $p(\mathbf{x})$. Correspondingly, in the BYY system, the representation

domain $\mathbf{Y}$ consists of each $\mathbf{y}=[y^{(1)},\ldots,y^{(m)}]^{\mathrm{T}}$ as an inner representation of $\mathbf{x}$, subject to a parametric structure of $q(\mathbf{y})$. In the literatures of both biological neural system modeling [3] and statistical data analysis [2], a widely adopted principle for representing the observed data via an inner representation is assuming that $\mathbf{y}=[y^{(1)},\ldots,y^{(m)}]^{\mathrm{T}}$ comes from a distribution $p(\mathbf{y})$ that is independent among components. The details of such a type are referred to the previous papers [37,38].

Here, a new development is to consider a world $\mathbf{X}=\{X,L\}$ that consists of a number of individual objects to observe, with L denoting a set of labels and each $\ell\in L$ denoting an object. In this case, each $\mathbf{x}=\{x,\ell\}$ contains a feature vector $x=[x^{(1)},\ldots,x^{(d)}]^{\mathrm{T}}$ observed from the object ℓ, subject to a joint underlying pdf $p(\mathbf{x})=p(x,\ell)=p(x|\ell)p(\ell)$. Correspondingly, we consider a modular representation domain $\mathbf{Y}=\{Y,L\}$ in the BYY system, subject to a parametric structure of $p(y,\ell)=p(y|\ell)p(\ell)$ that describes the vector $y=[y^{(1)},\ldots,y^{(m_\ell)}]^{\mathrm{T}}$ and the label ℓ jointly, in the sense that each $p(y|\ell)$, $\ell\in L$ is independent among components, weighted by $p(\ell)$. That is, we have a modular independence space.

Given a set of samples with each sample x appended with a known label ℓ, the learning tasks degenerate simply into the tasks of learning on each object separately. Thus, we generally consider a set of samples of x with all the labels absented. Specifically, the learning tasks can be understood from two complementary perspectives as follows:

- It follows from $q(x|y,\ell)$ that $p(x)$ can be estimated via

$$q(x)=\sum_{\ell\in L}\int q(x|y,\ell)q(y,\ell)\,\mathrm{d}y=\int q(x|y)q(y)\,\mathrm{d}y, \tag{10}$$

with several $q(x|y,\ell)$, $\ell\in L$ structures engaged in action under the control of $q(y,\ell)$. We can interpret $q(x)$ as a finite mixture

$$q(x)=\sum_{\ell\in L}q(x|\ell)q(\ell),\quad q(x|\ell)=\int q(x|y,\ell)q(y|\ell)\,\mathrm{d}y, \tag{11}$$

where each $q(x|\ell)$ describes an object $\ell\in L$ that is generated from each independence pdf $p(y|\ell)$, $\ell\in L$ via each individual Ying path.
Moreover, $q(x)$ can also be interpreted as generated from a finite mixture

$$q(y)=\sum_{\ell=1}^{k}q(y,\ell)=\sum_{\ell=1}^{k}q(y|\ell)q(\ell) \tag{12}$$

via $q(x|y)$ in the form of the so called *mixture-of-expert model* [12–14], i.e.,

$$q(x|y)=\sum_{\ell=1}^{k}q(x|y,\ell)q(\ell|y),\quad q(\ell|y)=q(y|\ell)q(\ell)/q(y), \tag{13}$$

which is a weighted sum of each expert $q(x|y,\ell)$, gated by either the alternative gate $q(\ell|y)$ [41,36] or the original softmax gate [12–14].

- Alternatively, we can also use $p(y,\ell|x)$, $\ell \in L$ to get $p(y,\ell) = \int p(y,\ell|x)p(x)\,dx$ that matches an expected inner pdf $q(y,\ell)$. It can be further understood from

$$p(y,\ell) = \int p(y,\ell|x,\ell)p(x,\ell)\,dx = p(y|\ell)p(\ell),$$
$$p(y|\ell) = \int p(y,\ell|x,\ell)p(x|\ell)\,dx. \tag{14}$$

That is, each $p(y,\ell|x,\ell)$ will discover certain structure of the corresponding object $\ell \in L$. Moreover, from Eq. (14) we have

$$p(y) = \sum_{\ell \in L} p(y,\ell) = \int p(y|x)p(x)\,dx,$$
$$p(y|x) = \sum_{\ell \in L} p(y|x,\ell)p(\ell|x). \tag{15}$$

That is, the Yang path can be implemented again by a mixture-of-expert model [13,14] with each expert $p(y|x,\ell)$, gated by $p(\ell|x)$. In this case, each sample x is not only mapped to an inner representation via $p(y|x)$ but also classified into an object ℓ in help of $p(\ell|x)$.

3.2. Learning implementation with B-architecture and BI-architecture

The architecture of a BYY system is featured by a combination of the specific structures of $p(\mathbf{y}|\mathbf{x}), q(\mathbf{x}|\mathbf{y})$, which leads to three choices, namely, the backward architecture, the Bi-directional architecture, and the forward architecture (shortly B-architecture, BI-architecture, and F-architecture, respectively). The details are referred to [38]. In this paper, we extend to consider modular B-architecture and modular BI-architecture.

In a modular B-architecture, $p(\mathbf{y}|\mathbf{x}) = p(y,\ell|x)$ is structure-free and will be indirectly specified via Eq. (9) by the structures of $q(y,\ell)$ and $q(x|y,\ell)$. Similar to a B-architecture in [38], the *least complexity nature* Eq. (6) of the harmony learning Eq. (9) will push

$$p(y,\ell|x) = p(y|x,\ell)p(\ell|x) \tag{16}$$

towards its least complexity form via a winner-take-all (WTA) competition

$$p(y|x,\ell) = \delta(y - y_\ell(x)), \quad p(\ell|x) = \delta(\ell - \ell(x)),$$
$$y_\ell(x) = \arg\min_y d(\ell,x,y), \quad \ell(x) = \arg\min_\ell d(\ell,x,y_\ell(x)),$$
$$d(\ell,x,y) = -\ln[q(x|y,\ell)q(y|\ell)q(\ell)] + \ln z_q, \tag{17}$$

where z_q does not vary with l,x,y and thus has no affect on the outcome of the minimization, but being conceptually useful. In other words, $q(\ell)$ is a δ pdf, and either or both of $q(x|y,\ell), q(y|\ell)$ may also be δ pdf, which makes the first part of $d(\ell,x,y)$ contains an infinite term $-\ln\delta(0)$. The affect of this infinite is canceled by $\ln z_q$ because it correspondingly contains a $\ln\delta(0)$.

In a modular BI-architecture, we consider a specific parametric model of Eq. (16) with

$$p(y|x,\ell) = G(y|y_\ell(x), h_y^2 I), \quad y_\ell(x) = f_\ell(x|\theta_{y|x,\ell}), \tag{18}$$

with $p(\ell|x)$ being structure-free. Putting it into Eq. (9), we get $p(\ell|x)$ as in Eq. (17) again but with

$$\ell(x) = \arg\max_{\ell} [q(x|y_\ell(x),\ell)q(y_\ell(x),\ell)]. \tag{19}$$

Actually, $p(y|x,\ell)$ in Eq. (17) can be regarded as a special case of Eq. (18) at $h_y = 0$. Putting $p(y|x,\ell)$ by Eq. (18) and $p(\ell|x)$ by Eq. (17) into Eq. (9), with Eq. (2) we get

$$H(p\|q) = \frac{1}{N}\sum_{t=1}^{N} H(x_t, y_{t,\ell_t}) - \ln z_q, \quad y_{\ell,t} = y_\ell(x_t), \ \ \ell_t = \ell(x_t),$$

$$H(x_t, y_{t,\ell_t}) = \int G(x|x_t, h_x^2 I) G(y|y_{t,\ell_t}, h_y^2 I) \ln[q(x|y,\ell_t)q(y,\ell_t)]\,\mathrm{d}x\,\mathrm{d}y. \tag{20}$$

Using v denoting either x or y, we have that $h_v^2 = 0$ when v comes from a discrete distribution or equivalently $G(v|v_t, h_v^2 I)$ degenerates to $\delta(v - v_t)$. In these cases, the integral in $H(x_t, y_{t,\ell_t})$ will disappear automatically.

In help of a trick used in [38], i.e., making Taylor expansion of $\ln q(x|y,\ell)$ and $\ln q(y,\ell)$ around x_t, y_t up to the second order, we can generally avoid the integral by the following approximation:

$$\begin{aligned} H(x_t, y_{t,\ell_t}) = {} & \ln[q(x_t|y_{t,\ell_t},\ell_t)q(y_{t,\ell_t}|\ell_t)q(\ell_t)] + 0.5h_x^2 Tr[H_q^x(x_t|y_{t,\ell_t},\ell_t)] \\ & + 0.5h_y^2 Tr[H_q^y(x_t|y_{t,\ell_t},\ell_t)] + 0.5h_y^2 Tr[H_q^y(y_{t,\ell_t},\ell_t)], \end{aligned} \tag{21}$$

$$H^x_{q(x|y,\ell)} = \frac{\partial^2 \ln q(x|y,\ell)}{\partial x \partial x^{\mathrm{T}}}, \quad H^y_{q(x|y,\ell)} = \frac{\partial^2 \ln q(x|y,\ell)}{\partial y \partial y^{\mathrm{T}}}, \quad H^y_{q(y,\ell)} = \frac{\partial^2 \ln q(y,\ell)}{\partial y \partial y^{\mathrm{T}}},$$

where, and throughout this paper, $Tr[A]$ denotes the trace of the matrix A.

Moreover, it follows from Eq. (8) that

$$z_q = \begin{cases} \displaystyle\sum_{t=1}^{N}\sum_{\ell=1}^{k} q(x_t|y_{t,\ell},\ell)q(y_{t,\ell},\ell) & \text{(a) } h_x = 0,\ h_y = 0, \\ \displaystyle\sum_{t=1}^{N} p_{h_x}(x_t)\sum_{\ell=1}^{k} q(y_{t,\ell},\ell) & \text{(b) } h_y = 0,\ h_x \text{ to be learned}, \\ \displaystyle\sum_{t=1}^{N}\sum_{\ell=1}^{k} \frac{q(x_t|y_{t,\ell},\ell)q(\ell)}{(2\pi h_y)^{0.5m_\ell}} & \text{(c) } h_x = 0,\ h_y \text{ to be learned}, \\ \displaystyle\sum_{t=1}^{N} p_{h_x}(x_t)\sum_{\ell=1}^{k} (2\pi h_y)^{-0.5m_\ell} & \text{(d) both } h_x, h_y \text{ to be learned}. \end{cases} \tag{22}$$

Similar to Eq. (8), (b) comes from (b) subject to $\sum_{t=1}^{N}\sum_{\ell=1}^{k} q(x_t|y_{t,\ell},\ell)q(y_{t,\ell},\ell) = \sum_{t=1}^{N}\sum_{\ell=1}^{k} p(y_{t,\ell},\ell|x_t)p_h(x_t)$. Similarly, (b) and (c) come from (a).

Putting the above z_q and Eq. (21) into Eq. (20), we can implement Eq. (9) both for parameter learning and model selection.

The task of parameter learning is to determine the parameter set θ that consists of all parameters in $q(x|y,\ell)$ and $q(y|\ell)$, plus (a) $\theta_{y|x,\ell}$ for a BI-architecture with $p(y|x,\ell)$ given in Eqs. (18) and (b) each of unknown h_x, h_y (if any).

The task of model selection is to select the values of $\mathbf{k} = \{k, \{m_\ell\}_{\ell=1}^{k}\}$, which can be further understood by observing that the least complexity nature Eq. (6) pushes $p(y,\ell|x)$ towards its least complexity form Eq. (17). Then, the matching nature of harmony learning in turn further pushes $q(x|y,\ell)$ and $q(y,\ell)$ towards their corresponding least complexity forms.

Specifically, parameter learning and model selection can be implemented either in the conventional two-phase style or in a parallel style in that model selection is made automatically during parameter learning.

In the two-phase style, we enumerate $k, \{m_\ell\}_{\ell=1}^{k}$ from small values incrementally, and at each specific setting of $\mathbf{k} = \{k, \{m_\ell\}_{\ell=1}^{k}\}$ we perform parameter learning

$$\max_{\theta} H(\theta), \quad H(\theta) = H(\theta, \mathbf{k}) \tag{23}$$

for a best value θ^*, during which we can simply set

$$q(\ell) = 1/k, \quad \int (y-\mu_{y,\ell})(y-\mu_{y,\ell})^{\mathrm{T}} q(y|\ell)\,\mathrm{d}y = b_0 I, \quad \mu_{y,\ell} = \int y q(y|\ell)\,\mathrm{d}y, \tag{24}$$

where $b_0 > 0$ is a given constant. For example, $b_0 = 0.25$ when y comes from Bernouilli and $b_0 = 1$ when y comes from Gaussian.

Then, we select best setting $\mathbf{k}^*$ by

$$\min_{\mathbf{k}} J(\mathbf{k}), \quad J(\mathbf{k}) = -H(\theta^*, \mathbf{k}). \tag{25}$$

Considering $q(y,\ell) = q(y|\ell)q(\ell)$, it is not difficult to observe that letting $p(\ell)$ to be zero is equivalent to reducing k by one. Moreover, we also observe that

$$q(y|\ell) = \delta(y^{(j)} - c_0)\prod_{r\neq j} q(y^{(r)}|\ell), \tag{26}$$

with c_0 being a constant, implies that the dimension $y^{(j)}$ can be removed and m_ℓ is reduced by one. That is, a value of θ with such settings is equivalent to forcing k, m_ℓ effectively to be reduced to appropriate scales.

With Eq. (17) inserted into Eq. (9), we can observe that the on-line implementation of $\max H(p\|q)$ is equivalent to maximizing $\ln q(x|y,\ell)$, $\ln q(y|\ell)$ and $\ln q(\ell)$. Specifically, maximizing $\ln q(\ell)$ leads to that

$$\text{all extra } q(\ell) \text{ are pushed towards zero.} \tag{27}$$

While maximizing $\ln q(y|\ell)$ leads to the nature that

$$q(y^{(j)}|\ell) \text{ on each extra dimension is pushed towards } \delta(y^{(j)} - c_0). \tag{28}$$

Therefore, the least complexity nature Eq. (6) does imply model selection during parameter learning. In other words, setting the scales m_ℓ, k large enough, we can implement parameter learning by Eq. (23) during which model selection is made automatically.

More specifically, parameter learning Eq. (23) is implemented via gradient ascending, by updating parameters in $q(x|y,\ell)$, $q(y|x,\ell)$, $q(y|\ell)$, an $p(\ell)$, separately. The details are referred to Table 1.

4. Unsupervised modular binary hidden layer networks

4.1. Modular B-architecture

We consider a B-architecture with

$$q(x|y,\ell) = G(x|A_\ell y + c_\ell, \sigma_\ell^2 I),$$

$$q(y|\ell) = \prod_{j=1}^{m_\ell} [q_{\ell,j}\delta(y^{(j)}) + (1 - q_{\ell,j})\delta(1 - y^{(j)})], \tag{29}$$

from which we get a mixture $q(x) = \sum_\ell q(x|\ell)q(\ell)$ with each $q(x|\ell) = \sum_y q(y|\ell)$ $q(x|y,\ell)$ itself consisting of 2^{m_ℓ} Gaussians $G(x|A_\ell y + c_\ell, \sigma_\ell^2 I)$, where the variance for each ℓ is a constant σ_ℓ^2 but the mean vector $A_\ell y + c_\ell$ changes as y varies according to $q(y|\ell)$. These Gaussians are mixed by $q(y|\ell)$ in an organization formed by locating a Gaussian with the zero mean and the covariance $\sigma_\ell^2 I$ at every vertex of a polyhedra that is obtained by an arbitrary affine transformation from a m_ℓ dimensional hypercubic in the R^d space of x.

Learning on the B-architecture Eq. (29) can be implemented as a special case of what discussed in the previous section. Specifically, a structure-free $p(y,\ell|x)$ will still lead to Eq. (17) with

$$d(\ell,x,y) = 0.5\frac{\|x - A_\ell y - c_\ell\|^2}{\sigma_\ell^2} + 0.5\ln\sigma_\ell^2$$
$$- \ln q(\ell) - \sum_{j=1}^{m_\ell} [y^{(j)} \ln q_{\ell,j} + (1 - y^{(j)}) \ln(1 - q_{\ell,j})]. \tag{30}$$

Also, it follows from Eq. (21) that

$$H^x_{q(x|y,\ell)} = -0.5\sigma_\ell^{-2} I, \quad H^y_{q(x|y,\ell)} = -0.5\sigma_\ell^{-2} A_\ell^{\mathrm{T}} A_\ell, \quad H_{q(y,\ell)} = 0. \tag{31}$$

Putting Eqs. (29), (30), and (31) into Table 1, learning can be implemented, with its Step 4 substituted by

4(a): $q^{\text{new}}_{\ell,j} = b^{\text{new }2}_{\ell,j}$, $b^{\text{new}}_{\ell,j} = b^{\text{old}}_{\ell,j} + \gamma_{t,\ell_t} b^{\text{old}}_{\ell,j}(y^{(j)}_{\ell,t} - q^{\text{old}}_{\ell,j})/[q^{\text{old}}_{\ell,j}(1 - q^{\text{old}}_{\ell,j})]$;

if either $q^{\text{new}}_{\ell,j} \to 0$ or $q^{\text{new}}_{\ell,j} \to 1$, discard the jth dimension

according to Eq. (28).

Table 1
A unified learning procedure for both B-architecture and BI-architecture

Set $z_q(t)=0,\ q(\ell)=1/k.\ \theta_{x|y,\ell},\theta_{y|\ell}$ denote parameters in $q(x|y,\ell),q(y|\ell)$, respectively. $\gamma_0>0$ denotes a constant that may take different values in updating different parameters

Step 1: $y_{\ell,t}=y_\ell(x_t),\ \ell_t=\ell(x_t)$, by $\begin{cases}\text{(a) Eq. (17) for a BI-architecture,}\\ \text{(b) Eq. (18) and Eq. (19) for a B-architecture.}\end{cases}$

Step 2:
$$z_q(t)=z_q(t-1)+\begin{cases}\sum_{\ell=1}^{k} q(x_t|y_{\ell,t},\ell)q(y_{\ell,t},\ell) & \text{(a) } h_x=0,\ h_y=0,\\ \sum_{\ell=1}^{k} q(y_{\ell,t},\ell) & \text{(b) } h_y=0,\ \text{but } h_x\neq 0 \text{ to be learned},\\ \sum_{\ell=1}^{k} q(x_t|y_{\ell,t},\ell)q(\ell) & \text{(c) } h_x=0,\ \text{but } h_y\neq 0 \text{ to be learned},\\ 0 & \text{(d) both } h_x,h_y \text{ to be learned.}\end{cases}$$
$$\gamma_{t,\ell}=\gamma_0[\tfrac{\delta_{\ell,\ell_t}}{t^{\text{new}}}-\gamma^d_{t,\ell}],$$
$$\delta_{\ell,\ell_t}=\begin{cases}1 & \ell=\ell_t,\\ 0 & \text{otherwise;}\end{cases}$$
$$\gamma^d_{t,\ell}=\begin{cases}\dfrac{q(x_t|y_{\ell,t},\ell)q(y_{\ell,t}|\ell)q(\ell)}{z_q(t)} & \text{(a) } h_x=0,\ h_y=0,\\ \dfrac{q(y_{\ell,t}|\ell)q(\ell)}{z_q(t)} & \text{(b) } h_y=0,\ \text{but } h_x\neq 0 \text{ to be learned},\\ \dfrac{q(x_t|y_{\ell,t},\ell)q(\ell)}{z_q(t)} & \text{(c) } h_x=0,\ \text{but } h_y\neq 0 \text{ to be learned},\\ 0 & \text{(d) both } h_x\neq 0,\ h_y\neq 0 \text{ to be learned.}\end{cases}$$

Note: $-\ln z_q$ *results in a de-learning via* $\gamma^d_{t,\ell}$ *in effect when either or both of* h_x,h_y *are zero.*

Step 3: Skip this step for the two-phase style implementation, otherwise update
$$q^{\text{new}}(\ell)=\begin{cases}\dfrac{q^{\text{old}}(\ell)+\gamma_0}{1+\gamma_0}, & \ell=\ell_t,\\ \dfrac{q^{\text{old}}(\ell)}{1+\gamma_0} & \text{otherwise.}\end{cases}$$
if $q^{\text{new}}(\ell)\to 0$, we discard the corresponding cluster ℓ according to Eq. (27).

Step 4: $\theta^{\text{new}}_{y|\ell}=\theta^{\text{old}}_{y|\ell}+\gamma_{t,\ell}\nabla_{\theta_{y|\ell}}J(y_{\ell,t},\theta_{y|\ell}),\ J(y,\theta_{y|\ell})=\ln q(y|\ell)+0.5h_y^2Tr[H_{q(y|\ell)}]$.
(which is subject to the constraint Eq. (24) for the two-phase style implementation.)

Step 5: $\theta^{\text{new}}_{x|y,\ell}=\theta^{\text{old}}_{x|y,\ell}+\gamma_{t,\ell_t}\nabla_{\theta_{x|y,\ell}}J(x_t,y_{\ell,t},\theta_{x|y,\ell})$,
$J(x,y,\theta_{x|y,\ell})=\ln q(x|y,\ell)+0.5Tr[h_x^2H^x_{q(x|y,\ell)}+h_y^2H^y_{q(y|\ell)}]$.

Table 1 (*continued*)

Step 6:	Skip this step for a B-architecture, otherwise via $y_\ell(x_t) = f_\ell(x_t\|\theta_{y\|x,\ell})$ to update $\theta^{\text{new}}_{y\|x,\ell} = \theta^{\text{old}}_{y\|x,\ell} + \gamma_{t,\ell_t}\nabla_{\theta_{y\|x,\ell}}[J(x_t, y_\ell(x_t), \theta_{x\|y,\ell}) + J(y_\ell(x_t), \theta_{y\|\ell})]$.
Step 7(a):	For $h_x \neq 0$ to be learned, it is updated via gradient ascent, i.e., $h_x^{\text{new}} = h_x^{\text{old}} + \gamma_0 g_x(h_x),\ g_x(h_x) = \frac{d}{h_x} + h_x Tr[H^x_{q(x_t\|y_t,\ell_t)}] - \frac{\mathrm{d}h^2_{x,0}}{h_x^3}$, $h^{2\ \text{new}}_{x,0} = (1-\gamma_0)h^{2\ \text{old}}_{x,0} + \gamma_0 \sum_{t=1}^n \sum_{t'=1}^n p_{t,t'}\|x_t - x_{t'}\|^2$, $p_{t,t'} = \frac{\mathrm{e}^{-0.5\|x_t - x_{t'}\|^2/h_x^2}}{z_q^D(t)},\ z_q^D(t) = (1-\gamma_0)z_q^D(t-1) + \gamma_0 \mathrm{e}^{-0.5\|x_t - x_{t'}\|^2/h_x^2}$. Initially, set $z_q^D(0) = 0$. Also, h_x can be set at the roots of $g_x(h_x) = 0$.
Note:	*The iterative process adaptively provides as approximations of* $h^2_{x,0} = (1/d)\sum_{t=1}^N \sum_{t'=1}^N p_{t,t'}\|x_t - x_{t'}\|^2,\ p_{t,t'} = \mathrm{e}^{-0.5\|x_t - x_{t'}\|^2/h_x^{\text{old}\ 2}}/z_q^N(h_x^{\text{old}}, d)$.
Step 7(b):	For $h_y \neq 0$ to be learned, it is also updated via gradient ascent, i.e., $h_y^{\text{new}} = h_y^{\text{old}} + \gamma_0 g_y(h_y),\ g_y(h_y) = \frac{m}{h_y} + h_y\, Tr[H^y_{q(x_t\|y_t,\ell_t)} + H^y_{q(y_t,\ell_t)}]$, Similarly, h_y can be initially set at the root of $g_y(h_y) = 0$.

$$4(\text{b}):\ e_{t,\ell} = x_t - A_{\ell_t} y_{\ell,t} - c_{\ell_t},\ A_\ell^{\text{new}} = A_\ell^{\text{old}} + \gamma_{t,\ell} e_{t,\ell}^{\text{old}} y_t^{\text{T}},\ c_\ell^{\text{new}} = c_\ell^{\text{old}} + \gamma_{t,\ell} e_{t,\ell}^{\text{old}},$$

$$\sigma_\ell^{\text{new}} = \sigma_\ell^{\text{old}} + \gamma_{t,\ell}\sigma_\ell^{\text{old}}(\|e_{t,\ell}^{\text{old}}\|^2 - \sigma_\ell^{2\ \text{old}}),\ \sigma_\ell^{\text{new}\ 2} = \sigma_\ell^{\text{new}}\sigma_\ell^{\text{new}}. \tag{32}$$

During learning, not only automatic selection on clusters is in action via Step 3, but also automatic determination on each dimension m_ℓ is made via Step 4(a) by checking if $q_{\ell,j}$ is pushed towards 1 or 0.

In the two-phase style implementation, we can simply fix $q(\ell) = 1/k, q_{\ell,j} = 1/2$. Putting Eqs. (22), (30), and (31) into Eqs. (20) and (21), we get a specific criterion from Eq. (25):

$$J(k, \{m_\ell\}) = \ln k + \frac{\sum_{\ell=1}^k m_\ell}{k}\ln 2 + \frac{0.5d}{k}\sum_{\ell=1}^k \left(\ln \sigma_\ell^2 + \frac{h_x^2}{\sigma_\ell^2}\right) + \ln z_q, \tag{33}$$

$$z_q = \begin{cases} \dfrac{1}{k}\displaystyle\sum_{t=1}^N \sum_{\ell=1}^k 2^{-m_\ell} G(x_t|A_\ell y_{\ell,t} + c_\ell, \sigma_\ell^2 I) & \text{(a) } h_x = 0, \\ \dfrac{z_q(h_x)}{N(2\pi h_x^2)^{d/2}},\ z_q(h_x) = \displaystyle\sum_{\tau=1}^N \sum_{t=1}^N \mathrm{e}^{-0.5\|x_t - x_\tau\|^2/h_x^2} & \text{(b) } h_x \neq 0. \end{cases}$$

Particularly, when $k = 1$, removing the label ℓ, it is simplified into

$$J(m) = \begin{cases} \ln + \ln \displaystyle\sum_{t=1}^N \mathrm{e}^{-0.5\frac{\|x_t - A y_t - c\|^2}{\sigma^2}} & \text{(a) } h_x = 0, \\ m\ln 2 + 0.5d\ln\sigma^2 + 0.5\dfrac{\mathrm{d}h_x^2}{\sigma^2} + \ln\frac{z_q(h_x)}{N(2\pi h_x^2)^{d/2}} & \text{(b) } h_x \neq 0. \end{cases}$$

4.2. Modular BI-architecture

In a B-architecture discussed above, the task of getting $y_{\ell,t}$ is a combinatorial optimization problem on the quadratic cost $d(\ell, x, y)$ in Eq. (30), which should be solved per sample x_t. This computing cost can be very expensive. In a BI-architecture, this cost can be avoided when $p(y|x,\ell)$ is given by a special case of Eq. (18) as follows:

$$y_\ell(x) = s(\hat{y}_\ell), \quad \hat{y}_\ell = W_\ell x + d_\ell, \tag{34}$$

where and hereafter in this paper, $s(r)$ denotes a sigmoid function $s(r) = (1 + e^{-\beta r})^{-1}$. Also, for a vector $u = [u^{(1)}, \ldots, u^{(m)}]^T$, we denote $s(u) = [s(u^{(1)}), \ldots, s(u^{(m)})]^T$.

In this case, Table 1 is implemented with Eq. (32). Moreover, Step 5 takes the following detailed form:

$$\begin{aligned} &Step\ 5: e^0_{t,\ell} = A_\ell^{\text{old } T} e^x_{t,\ell} + \sigma_\ell^{\text{old } 2} e^y_{t,\ell},\ e^x_{t,\ell} = x_t - A_\ell y_{\ell,t} - c_\ell, \\ &\qquad e^y_{t,\ell} = \left[\ln \frac{q_{\ell,1}}{1-q_{\ell,1}}, \ldots, \ln \frac{q_{\ell,m}}{1-q_{\ell,m}}\right]^T,\ d_\ell^{\text{new}} = d_\ell^{\text{old}} + \gamma_{t,\ell} D_s(\hat{y}_{\ell,t}) e^0_{t,\ell}, \\ &\qquad W_\ell^{\text{new}} = W_\ell^{\text{old}} + \gamma_{t,\ell} D_s(\hat{y}_\ell) e^0_{t,\ell} x_t^T,\ \hat{y}_{\ell,t} = W_\ell x_t + d_\ell, \end{aligned} \tag{35}$$

where $e^y_{t,\ell}$ is the correcting term from the part of $q(y|\ell)$. Here and hereafter, the notation $D_s(u)$ denotes a diagonal matrix with its diagonal elements $[s'(u^{(1)}), \ldots, s'(u^{(m)})]^T$ and $s'(r) = ds(r)/dr$.

To get a further insight, we consider Eq. (30) at the special case that $k = 1$, $q_j = q_{\ell,j} = 0.5$, $c = c_\ell = 0$, $d = d_\ell = 0$, $h_x = 0$, and $z_q = 1$. It follows that the harmony learning Eq. (23) becomes equivalent to minimize

$$\sigma^2 = \frac{1}{N} \sum_{t=1}^{N} \|x_t - As(Wx_t)\|^2, \tag{36}$$

which is called auto-association learning via three layer net and can be trained in the same way as training a three layer net by the Back-propagation technique [28]. Under the constraint that $A = W^T$, it further becomes the so called *least mean square error reconstruction (LMSER)* learning that was firstly proposed in 1991 [34] with both batch and adaptive gradient algorithms provided. Also, it was found that a sigmoid non-linearity $s(r)$ leads to an automatic breaking on the symmetry of the components in the subspace. Three years later, the LMSER learning and its adaptive algorithm have been directly adopted in [15] to implement ICA with promising results under the name of nonlinear PCA.

Why LMSER performs such an ICA task can be understood from the perspective of Eqs. (29) and (34). By Eq. (29), each bit of y is expected to take value 1 with probability 0.5, independent from other bits. By Eq. (34), each mapping from $s(\hat{y}_\ell)$ to y has no cross-talk among components. Thus, the independence of y among components

means the independence of $\hat{y}=Wx$, i.e., it implements ICA. One key feature of this ICA is that the observation noise is considered via minimizing σ^2, so it actually implements a noisy ICA problem.

Moreover, the BI-architecture Eqs. (29) and (34) with the learning algorithm Eq. (35) extends the LMSER learning with the following new results:

(a) *Criterion for hidden unit number and automatic selection.* With $k=1$ and $q_j=q_{\ell,j}=0.5$, we can use Eq. (33) for deciding the number m of hidden unit, with σ^2 given in Eq. (36). Moreover, with $q_j=q_{\ell,j}$ not fixed, but learned by Step 4(a) in Eq. (32), automatic selection on these hidden units will be made during learning.

(b) *ICA that works on both super-Gaussian and sub-Gaussian sources.* Instead of fixing $q_j=q_{\ell,j}=0.5$, making learning with q_j updated via Eq. (32) will let q_j to adapt the distribution of y such that the above discussed ICA works on a noisy observation x that is generated from y with components coming from either or both of super-Gaussian and sub-Gaussian sources. Thus, it not only acts as an alternative of the LPM-ICA [40] with a much simplified computation, but also makes the sources selected via either Eq. (33) or automatically during learning.

(c) *Local LMSER for structural clustering and competitive ICA.* With $k>1$, a local LMSER learning is made locally on each cluster of data via the competition Eq. (17), i.e., $\ell_t=\arg\min_\ell d(\ell,x_t,y_\ell(x_t))$. In other words, it implements a competitive ICA, with the number k selected via either Eq. (33) or automatically during learning.

4.3. Bernoulli LMSER and automatic selection

We further consider the cases that $x=[x^{(1)},\ldots,x^{(d)}]^{\mathrm{T}}$ with each $x^{(j)}=0$ or 1. That is, we regard each binary code x generated from an inner binary code y. One example that explores along this direction is the early work called multiple cause mixture [29]. It models each bit $\hat{x}^{(j)}=1-\prod_i(1-y_i a_{ij})$ via binary a_{ij} and then matches the observed bit $x^{(j)}$ with a cost function. Learning becomes a combinatorial optimization problem that searches the values of $a_{i,j}$. Also, the maximum likelihood learning is proposed on this model [8], with each binary code x interpreted as Bernoulli via defining the probability that $x^{(j)}=1$ in help of the generating model $1-\prod_i(1-y_i a_{ij})$. Many studies have been made along this line in literature.

Alternatively, here we consider BYY learning on either a B-architecture with

$$q(x|y)=\prod_{j=1}^{d}[s(\hat{x}^{(j)})\delta(x^{(j)})+(1-s(\hat{x}^{(j)}))\delta(1-x^{(j)})],\quad \hat{x}=Ay+c,$$

$$q(y)=\prod_{j=1}^{m}[q_j\delta(y^{(j)})+(1-q_j)\delta(1-y^{(j)})], \tag{37}$$

or a Bi-architecture with $q(y|x)=\delta(y-s(\hat{y}))$, $\hat{y}=Wx+d$ added in.

Table 2
Simplified learning procedure for logistic LMSER

Step 1:	$y_t = \arg\max_y \{\sum_{j=1}^{m} [y^{(j)} \ln q_j + (1-y^{(j)}) \ln(1-q_j)] + \sum_{j=1}^{d} [x_t^{(j)} \ln s(\hat{x}^{(j)}) + (1-x_t^{(j)}) \ln(1-s(\hat{x}^{(j)}))]\}$;
Step 2:	For the two-phase style implementation, simply fix $q_j = 0.5$, $\forall j$, otherwise update $q_j^{\text{new}} = \begin{cases} \dfrac{q_j^{\text{old}} + \gamma_0}{1+\gamma_0} & \text{if } y_t^{(j)} = 1, \\ \dfrac{q_j^{\text{old}}}{1+\gamma_0} & \text{if } y_t^{(j)} = 0; \end{cases}$ if either $q_j^{\text{new}} \to 0$ or $q_j^{\text{new}} \to 1$, discard the corresponding jth dimension according to Eq. (28).
Step 3:	$e_{x\|y} = x_t - s(\hat{x}_t),\ c^{\text{new}} = c^{\text{old}} + \gamma_0 e_{x\|y},\ A^{\text{new}} = A^{\text{old}} + \gamma_0 e_{x\|y} y_t^{\mathrm{T}}$.
Step 4:	Skip this step for a B-architecture, otherwise for a BI-architecture, update $e_t = \beta A^{\text{old}} e_{x\|y} + [\ln \frac{q_1}{1-q_1}, \ldots, \ln \frac{q_m}{1-q_m}]^{\mathrm{T}}, W^{\text{new}} = W^{\text{old}} + \gamma_0 D_s(\hat{y}_t) e_t x_t^{\mathrm{T}},\ \hat{y}_t = W_{x_t} + d$ $d^{\text{new}} = d^{\text{old}} + \gamma_0 D_s(\hat{y}_t) e_t,\ D_s(\hat{y}_t) = diag[s'(\hat{y}_t^{(1)}), \ldots, s'(\hat{y}_t^{(m)})],\ s'(r) = \frac{\mathrm{d}s(r)}{\mathrm{d}r}$.

With $k = 1$, learning is made by Table 2 that is simplified from Table 1. The dimension m is determined either automatically by Step 2 or fixing $q_j = 0.5$ in help of Eq. (25) that is simplified into

$$J(m) = -\frac{1}{N}\sum_{t=1}^{N}\sum_{j=1}^{d}[x_t^{(j)} \ln s(\hat{x}_t^{(j)}) + (1-x_t^{(j)}) \ln(1-s(\hat{x}_t^{(j)}))] + \ln \sum_{t=1}^{N}\prod_{j=1}^{d} s(\hat{x}_t^{(j)})^{x_t^{(j)}} (1-s(\hat{x}_t^{(j)}))^{1-x_t^{(j)}}. \tag{38}$$

5. Supervised modular binary hidden layer networks

5.1. Forward mapping via a specific B-architecture

We start at re-considering Eq. (29) by dividing $x=[\xi,\eta]$, which leads to $q(\xi,\eta|y,\ell)= q(\eta|\xi,y,\ell)q(\xi|y,\ell)$ while $q(y|\ell)$ remains unchanged.

Specifically, $q(\xi|y,\ell)$ can be further modeled via the following Bayesian inverse:

$$q(\xi|y,\ell) = \frac{q(y|\xi,\ell)\, p_{h_\xi}(\xi)}{p(y|\ell)}, \quad p(y|\ell) = \int q(y|\xi,\ell)\, p_{h_\xi}(\xi)\, \mathrm{d}\xi, \tag{39}$$

with $p_h(\xi)$ estimated by Eq. (2) from a training set $\{\xi_t\}_{t=1}^N$, and $q(y|\xi,\ell)$ given by a logistic layer as follows:

$$q(y|\xi,\ell)=\delta(y-s(\hat{y}_\ell)),\quad \hat{y}_\ell=W_\ell\xi+d_\ell. \tag{40}$$

Moreover, considering η being Gaussian and assuming that η is independent from ξ upon knowing y, we have $q(\eta|\xi,y,\ell)=q(\eta|y,\ell)=G(\eta|A_\ell y+c_\ell,\sigma_\ell^2 I)$. Then, for each ℓ we can get a cascaded mapping $\xi\to y\to\eta$ by

$$\begin{aligned} q(\eta|\xi,\ell)&=\int G(\eta|A_\ell y+c_\ell,\sigma_\ell^2 I)q(y|\xi,\ell)\,\mathrm{d}y\\ &=G(\eta|A_\ell s(W_\ell\xi+d_\ell)+c_\ell,\sigma_\ell^2 I), \end{aligned} \tag{41}$$

where its regression $A_\ell s(W_\ell\xi+d_\ell)+c_\ell$ implements a forward mapping $\xi\to y\to\eta$ by a conventional three layer net with sigmoid hidden units, which has been widely used in literature and trained by the well known back-propagation technique [28].

Furthermore, we can also get $q(\eta|\xi)=\sum_{\ell=1}^k q(\ell|\xi)q(\eta|\xi,\ell)$ with

$$q(\ell|\xi_t)=q(\ell|y_{\ell,t})=\frac{q(y_{\ell,t}|\ell)q(\ell)}{\sum_{\ell=1}^k q(y_{\ell,t}|\ell)q(\ell)},\quad y_{\ell,t}=s(W_\ell\xi_t+d_\ell). \tag{42}$$

Thus, $E(\eta|\xi)=\sum_{\ell=1}^k q(\ell|\xi)[A_\ell s(W_\ell\xi+d_\ell)+c_\ell]$ implements a mixture of several three layer nets, with a Bayesian gate $q(\ell|\xi)$. That is, we get a variant of the alternative ME model [41,36], with $q(\ell|\xi)=q(\ell|y)$ that is indirectly modeled by $q(y|\ell)$ given in Eq. (29).

The components $q(y|\xi,\ell)$, $G(\eta|A_\ell y+c_\ell,\sigma_\ell^2 I)$, $q(y|\ell)$ and $q(\ell)$ are learned from a set of sample pairs $\{\xi_t,\eta_t\}_{t=1}^N$ such that the above forward mapping performs as desired. In order to apply the BYY harmony learning on this learning task in a way similar to what made in Section 4, by summing up the above discussions we consider the following B-architecture:

$$\begin{aligned} &q(y|\ell)=\prod_{j=1}^{m_\ell}[q_{\ell,j}\delta(y^{(j)})+(1-q_{\ell,j})\delta(1-y^{(j)})],\quad q(\ell)\geqslant 0,\ \sum_{\ell=1}^k q(\ell)=1,\\ &q(\xi,\eta|y,\ell)=G(\eta|A_\ell y+c_\ell,\sigma_\ell^2 I)\frac{q(y|\xi,\ell)\,p_{h_\xi}(\xi)}{p(y|\ell)}, \end{aligned} \tag{43}$$

where $p(y|\ell)$ is obtained via the integral given in Eq. (39).

We can further get an easy implementing representation for $p(y|\ell)$. With $p_h(\xi)$ given by Eq. (2), it follows from Eq. (39) that $p(y|\ell)=1/N\sum_{t=1}^N p_t(y|\ell)$, where each $p_t(y|\ell)$ is mapped from a Gaussian $G(\xi|\xi_t,h_\xi^2 I)$ that undergoes firstly a linear map $\hat{y}_\ell=W_\ell\xi+d_\ell$ and then a sigmoid map $y_\ell=s(\hat{y}_\ell)$. Specifically, the linear map $\hat{y}_\ell=W_\ell\xi+d_\ell$ results in a Gaussian $p_t(\hat{y}_\ell|\ell)=G(\hat{y}_\ell|\hat{y}_{\ell,t},h_\xi^2 W_\ell W_\ell^{\mathrm{T}})$, $\hat{y}_{\ell,t}=W_\ell\xi_t+d_\ell$, that is further mapped by $y_\ell=s(\hat{y}_\ell)$ into $p(y_\ell|\ell)=p_t(y_\ell|\ell)=p_t(\hat{y}_\ell|\ell)/|D_s(\hat{y}_\ell)|$. Also, it follows from $\bar{y}_\ell=(W_\ell W_\ell^{\mathrm{T}})^{-0.5}\hat{y}_\ell$ that $p_t(\hat{y}_\ell|\ell)=p_t(\bar{y}_\ell|\ell)|W_\ell W_\ell^{\mathrm{T}}|^{-0.5}$ and

$p_t(\bar{y}_\ell|\ell) = G(\bar{y}_\ell|\bar{y}_{\ell,t}, h_\xi^2 I)$. Thus, we have

$$p_{h_\xi}(y_\ell|\ell) = \frac{1}{N}\sum_{\tau=1}^{N} \frac{G(\bar{y}_\ell|\bar{y}_{\ell,t}, h_\xi^2 I)}{|W_\ell W_\ell^{\mathrm{T}}|^{0.5}|D_s(\hat{y}_{\ell,t})|}, \quad \bar{y}_\ell = (W_\ell W_\ell^{\mathrm{T}})^{-0.5}\hat{y}_\ell,$$

$$y_\ell = s(\hat{y}_\ell), \quad \hat{y}_{\ell,t} = W_\ell \xi_t + d_\ell, \quad \bar{y}_{\ell,t} = (W_\ell W_\ell^{\mathrm{T}})^{-0.5}\hat{y}_{\ell,t}. \tag{44}$$

Particularly, as $p_h(\xi)$ becomes an empirical density with $h_\xi \to 0$, we have

$$p(y|\ell) = p_0(y|\ell) = \frac{1}{N}\sum_{\tau=1}^{N} \frac{\delta(\bar{y}_\ell - \bar{y}_{\ell,t})}{|W_\ell W_\ell^{\mathrm{T}}|^{0.5}|D_s(\hat{y}_{\ell,t})|}. \tag{45}$$

5.2. Adaptive learning, regularization, and model selection

We start at considering the empirical density $p_0(x)$, $x = [\xi, \eta]$. It follows from Eqs. (43) and (45) that we can modify Eqs. (17) and (20) into

$$\begin{aligned}
&H(p\|q) = \frac{1}{N}\sum_{t=1}^{N} \ln Q_0(\xi_t, \eta_t, \ell_t) - \ln z_q, \quad \ell_t = \arg\max_\ell Q_0(\xi_t, \eta_t, \ell),\\
&Q_0(\xi, \eta, \ell) = G(\eta|A_\ell y_\ell + c_\ell, \sigma_\ell^2 I) q(y_\ell|\ell) q(\ell) |W_\ell W_\ell^{\mathrm{T}}|^{0.5} |D_s(\hat{y}_\ell)|, \quad y_\ell = s(\hat{y}_\ell),\\
&\hat{y}_\ell = W_\ell \xi + d_\ell, \quad z_q = \begin{cases} 1 & \text{(a) a crude approximation,} \\ \displaystyle\sum_{t=1}^{N} Q_0(\xi_t, \eta_t, \ell_t) & \text{(b) normalization.} \end{cases}
\end{aligned} \tag{46}$$

Similar to Table 1, putting Eq. (46) into Eq. (9), learning can be implemented by the adaptive learning procedure given in Table 3. Similarly, automatic selection is made on k by Step 3 and on m_ℓ by Step 4, respectively. Alternatively, simply setting $q(\ell) = 1/k$, $q_{\ell,j} = 0.5$, we can also make model selection by Eq. (25).

For the choice of $z_q = 1$, we have $J(\mathbf{k})$ as follows:

$$J(\mathbf{k}) = \ln k + \frac{1}{k}\sum_{\ell=1}^{k} [0.5d \ln \sigma_\ell^2 + m_\ell \ln 2 + R_\ell],$$

$$R_\ell = -0.5 \ln |W_\ell W_\ell^{\mathrm{T}}| - \sum_{t=1}^{N} \ln |D_s(\hat{y}_{\ell,t})|. \tag{47}$$

For the choice of z_q given by (b) in Eq. (46), we have

$$J(\mathbf{k}) = \ln \hat{z}_q + \frac{1}{k}\sum_{\ell=1}^{k} [m_\ell \ln 2 + R_\ell],$$

$$\hat{z}_q = \sum_{\ell=1}^{k} \frac{|W_\ell W_\ell^{\mathrm{T}}|}{2^{m_\ell}} \sum_{t=1}^{N} |D_s(\hat{y}_{\ell,t})| \mathrm{e}^{-0.5\frac{\|\eta_t - A_\ell y_{\ell,t} - c_\ell\|^2}{\sigma_\ell^2}}. \tag{48}$$

Table 3
Learning procedure for modular sigmoid hidden layer net

Set $z_q(t)=0,\ q(\ell)=1/k,\ q_{\ell,j}=0.5$.

Step 1: $y_{\ell,t}=y_\ell(\xi_t),\ y_\ell(\xi)=s(W_\ell\xi+d_\ell),\ \ell_t=\arg\max_\ell Q_0(\xi_t,\eta_t,\ell)$,
$Q_0(\xi,\eta,\ell)=G(\eta|A_\ell y_\ell(\xi)+c_\ell,\sigma_\ell^2 I)q(y_\ell|\ell)q(\ell)|W_\ell W_\ell^{\mathrm{T}}|^{0.5}|D_s(\hat{y}_\ell)|,\ \hat{y}_\ell=W_\ell\xi+d_\ell$.

Step 2: $z_q(t)=z_q(t-1)+\sum_{\ell=1}^{k}Q_0(\xi_t,\eta_t,\ell),\ \gamma_{t,\ell}=\gamma_0[\frac{\delta_{\ell,\ell_t}}{t^{\mathrm{new}}}-\gamma_{t,\ell}^d],\ \gamma_{t,\ell}^d=\begin{cases}0 & \text{(a) } z_q=1,\\ Q_0(\xi_t,\eta_t,\ell)/z_q(t) & \text{(b) normalization.}\end{cases}$

Step 3: For the two-phase style implementation, fix $q(\ell)=\frac{1}{k}$ and skip this step, otherwise update

$$q^{\mathrm{new}}(\ell)=\begin{cases}\dfrac{q^{\mathrm{old}}(\ell)+\gamma_0}{1+\gamma_0}, & \ell=\ell_t,\\ \dfrac{q^{\mathrm{old}}(\ell)}{1+\gamma_0} & \text{otherwise.}\end{cases}$$

If $q^{\mathrm{new}}(\ell)\to 0$, we discard the corresponding network ℓ according to Eq. (27).

Step 4: For the two-phase style implementation, fix $q_j=0.5,\ \forall j$ and skip this step, otherwise update

$$q_{\ell,j}^{\mathrm{new}}=\begin{cases}\dfrac{q_{\ell,j}^{\mathrm{old}}+\gamma_0}{1+\gamma_0} & \text{if } y_{\ell,t}^{(j)}=1\\ \dfrac{q_{\ell,j}^{\mathrm{old}}}{1+\gamma_0} & \text{if } y_{\ell,t}^{(j)}=0\end{cases}\qquad \text{If either } q_{\ell,j}^{\mathrm{new}}\to 0 \text{ or } q_{\ell,j}^{\mathrm{new}}\to 1,$$

discard the corresponding jth dimension in the network ℓ according to Eq. (28).

Step 5: $e_{t,\ell}^{\eta}=\eta_t-A_\ell y_{\ell,t}-c_\ell,\ c_\ell^{\mathrm{new}}=c_\ell^{\mathrm{old}}+y_{\ell,t}g_{c_\ell},\ g_{c_\ell}=e_{t,\ell}^{\eta},\ A_\ell^{\mathrm{new}}=A_\ell^{\mathrm{old}}+\gamma_{t,\ell}g_{c_\ell}y_{t,\ell}^{\mathrm{T}}$,
$\sigma_\ell^{\mathrm{new}}=\sigma_\ell^{\mathrm{old}}+\gamma_{t,\ell}\sigma_\ell^{\mathrm{old}}(\|e_{t,\ell_t}^{\eta}\|^2-\sigma_\ell^{2\ \mathrm{old}}),\ \sigma_\ell^{\mathrm{new}\ 2}=\sigma_\ell^{\mathrm{new}}\sigma_\ell^{\mathrm{new}}$.

Note: The updating direction g_{c} comes from the gradient direction multiplied by a positive scalar σ_ℓ^2.

Step 6: $d_\ell^{\mathrm{new}}=d_\ell^{\mathrm{old}}+\gamma_{t,\ell}g_{d_\ell},\ g_d=D_s(\hat{y}_{\ell,t})\,(\frac{A_\ell e_{t,\ell}^{\eta}}{\sigma_\ell^{\mathrm{new}}}+e_{t,\ell}^{y})+e_{t,\ell}^{s}$,

$$W_\ell^{\mathrm{new}}=W_\ell^{\mathrm{old}}+\gamma_{t,\ell}g_{w_\ell},\ g_{w_\ell}=\begin{cases}g_{d_\ell}\xi^{\mathrm{T}}+(W_\ell^{\mathrm{old}}W_\ell^{\mathrm{old}^{\mathrm{T}}})^{-1}W_\ell^{\mathrm{old}}, & \text{(a) gradient,}\\ [g_{d_\ell}(W_\ell^{\mathrm{old}}\xi)^{\mathrm{T}}+I]W_\ell^{\mathrm{old}}, & \text{(b) natural gradient,}\end{cases}$$

$s'(r)=\frac{\mathrm{d}s(r)}{\mathrm{d}r},\ \hat{y}_{\ell,t}=W_\ell\xi_t+d_\ell,\quad e_{t,\ell}^{y}=[\ln\frac{q_1}{1-q_1},\ldots,\ln\frac{q_m}{1-q_m}]^{\mathrm{T}},\ e_{t,\ell}^{s}=[1-2y_{\ell,t}^{(1)},\ldots,1-2y_{\ell,t}^{(m)}]$.

Note: (a) $e_{t,\ell}^{s}$ is resulted from $d\ln|D_s(\hat{y}_\ell)|=Tr[D_s^{-1}(\hat{y}_\ell)\,\mathrm{d}D_s(\hat{y}_\ell)]$ via a sigmoid function $s(r)$ with a feature $s'(r)=-\beta s(r)(1-s(r))$, e.g., $s(r)=1/(1+\mathrm{e}^{-\beta r})$;
(b) The second term of g_w comes from $d\ln|W_\ell W_\ell^{\mathrm{T}}|^{0.5}$.

where $\ln\hat{z}_q$ comes from $\ln z_q$ after releasing two terms that cancel $\ln k$ and $0.5\ln\sigma_\ell^2$.

In the particular case at $k=1$, Table 3 provides *a new adaptive learning algorithm* for training a conventional three layer net as a replacement of the back-propagation

technique, with a feature that regularization is in action via a de-learning rate $\gamma_{t,\ell}^{d}$ in Step 2 and selection of hidden units takes places automatically during learning by Step 4.

Also, in this special case, $J(\mathbf{k})$ is simplified into

$$J(\mathbf{k})=\begin{cases}0.5d\ln\sigma^2+m\ln 2-0.5\ln|WW^{\mathrm{T}}|-\sum_{t=1}^{N}\ln|D_s(\hat{y}_{\ell,t})| & \text{for Eq. (47)},\\ \ln\sum_{t=1}^{N}\left[|D_s(\hat{y}_{\ell,t})|\mathrm{e}^{-0.5\frac{\|\eta_t-A_\ell y_{\ell,t}-c_\ell\|^2}{\sigma_\ell^2}}\right]-\sum_{t=1}^{N}\ln|D_s(\hat{y}_{\ell,t})| & \text{for Eq. (48)}.\end{cases}$$

Another special case is that $A_\ell=0,\sigma_\ell^2=0$ for all ℓ, with the mapping $y\to\eta$ broken. In this case, Table 3 is simplified with Step 5 discarded and $Q_0(\xi,\eta,\ell)$ in Steps 1 and 2 replaced by $Q_0(\xi,\ell)=q(y_\ell|\ell)q(\ell)|W_\ell W_\ell^{\mathrm{T}}|^{0.5}|D_s(\hat{y}_\ell)|$. Interestingly, this simplified learning procedure actually makes the mapping $\xi\to y$ implement a *competitive ICA*, that is, implementing ICA at different locations via the competitive allocation by Step 1. Particularly, when $k=1$ and $d_\ell=0$, the updating on W_ℓ by the natural gradient becomes the same format as the popular Amari's natural gradient ICA algorithm, but with a different g_d here. The difference makes this ICA learning applicable to the cases that consist of both super-Gaussian and sub-Gaussian sources by adapting q_j via Step 4. This q_j can be used as an identifier on whether this source is super-Gaussian or sub-Gaussian. Since

$$k_j=E(\bar{y}^{(j)}-q_j)^4-3[E(\bar{y}^{(j)}-q_j)^2]^2=(1-q_j)q_j\left[q_j-\frac{3+\sqrt{3}}{6}\right]\left[q_j-\frac{3-\sqrt{3}}{6}\right],$$

we have

$$k_j>0 \text{ if } q_j>\frac{3+\sqrt{3}}{6} \text{ or } q_j<\frac{3-\sqrt{3}}{6} \text{ and } k_j<0 \text{ if } \frac{3+\sqrt{3}}{6}\geqslant q_j\geqslant\frac{3-\sqrt{3}}{6}.$$

5.3. Logistic outputs, smoothed learning, and Bernoulli hidden layer

The above results can be extended in three aspects as follows.

First, we can consider the cases that η is binary instead of Gaussian, by replacing $G(\eta|A_\ell y+c_\ell,\sigma_\ell^2 I)$ in both Eq. (43) and Step 1 of Table 3 with

$$q(\eta|y)=\prod_{j=1}^{d_\eta}[s(\hat{\eta}^{(j)})\delta(\eta^{(j)})+(1-s(\hat{\eta}^{(j)}))\delta(1-\eta^{(j)})],\quad \hat{\eta}=By+b, \tag{49}$$

where d_η is the dimension of η.

Correspondingly, Step 5 of Table 3 is replaced with

$$\textit{Step } 5:\ \varepsilon_t=\eta_t-s(\hat{\eta}_t),\ \ B^{\text{new}}=B^{\text{old}}+\gamma_0\varepsilon_t y_{\ell,t}^{\mathrm{T}},\ \ b^{\text{new}}=b^{\text{old}}+\gamma_0\varepsilon_t. \tag{50}$$

Also, for $J(\mathbf{k})$ in Eq. (47), the term $0.5d\ln\sigma_\ell^2$ is replaced with

$$J_{\eta,y}(\mathbf{k})=-\frac{1}{N}\sum_{t=1}^{N}\sum_{j=1}^{d_\eta}[\eta_t^{(j)}\ln s(\hat{\eta}_t^{(j)})+(1-\eta_t^{(j)})\ln(1-s(\hat{\eta}_t^{(j)}))]. \tag{51}$$

Second, we can consider the smoothing learning by modeling $x=[\xi,\eta]$ via the Parzen window estimate Eq. (2), i.e., $p([\xi,\eta])=1/N\sum_{t=1}^{N}G(\xi|\xi_t,h_\xi^2 I)G(\eta|\eta_t,h_\eta^2 I)$. Similar to Eq. (46), we can modify Eqs. (17) and (20) into

$$H(p\|q)=\frac{1}{N}\sum_{t=1}^{N}H(\xi_t,\eta_t,\ell_t)-\ln z_q,\quad \ell_t=\arg\max_{\ell}Q(\xi_t,\eta_t,\ell),$$

$$\tilde{Q}(\xi,\eta,\ell)=Q_0(\xi,\eta,\ell)\frac{p_{h_\xi}(\xi)}{p_{h_\xi}(y_\ell|\ell)},\quad z_q=\sum_{t=1}^{N}Q(\xi_t,\eta_t,\ell_t),$$

$$\tilde{Q}(\xi,\eta,\ell)=G(\eta|A_\ell y_\ell+c_\ell,\sigma_\ell^2 I)q(y_\ell|\ell)q(\ell)$$

$$H(\xi_t,\eta_t,\ell_t)=\int G(\xi|\xi_t,h_\xi^2 I)G(\eta|\eta_t,h_\eta^2 I)\ln[Q(\xi,\eta,\ell_t)]\,\mathrm{d}\xi\,\mathrm{d}\eta,$$

$$\approx \ln Q(\xi_t,\eta_t,\ell_t)-\ln p_{h_\xi}(y_{\ell,t}|\ell)+0.5h_\xi^2 Tr[H_H]-0.5h_\eta^2\sigma_\ell^{-2}Tr[I_\eta],$$

$$H_H=H_q^\xi(\eta_t|y_{\ell,t},\ell_t)+H_q^\xi(y_{\ell,t}|\ell_t)+H_{p_{h_\xi}}^\xi(\xi_t)-H_p^\xi(y_{\ell,t}|\ell_t),\tag{52}$$

where $H_q^\xi(\eta|y_\ell,\ell)$, $H_q^\xi(y_\ell|\ell)$, $H_{p_{h_\xi}}^\xi(\xi_t)$, and $H_p^\xi(y_\ell|\ell)$ are the Hessian matrices of $\ln G(\eta|A_\ell y_\ell+c_\ell,\sigma_\ell^2 I)$, $\ln q(y_\ell|\ell)$, $\ln p_{h_\xi}(\xi)$, and $\ln p_{h_\xi}(y_\ell|\ell)$, respectively, all with respect to ξ.

Putting Eq. (52) into Eq. (9), learning can be implemented by an adaptive learning procedure similar to Table 3, with three key modifications. First, $Q_0(\xi_t,\eta_t,\ell_t)$ is replaced by $Q(\xi_t,\eta_t,\ell_t)$ in Steps 1 and 2. Second, in Steps 5 and 6, the updating on the parameter set θ should add $0.5h_\xi^2\partial Tr[H_H]/\partial\theta$ into its updating direction. Moreover, the updating direction on σ_ℓ should also be added in with $h_\xi^2\sigma_\ell^{-3}$. Third, Step 7 should be added as in Table 1 with the part for updating h_ξ,h_η. Finally, we can obtain $J(k)$ for model selection in a way similar to Eq. (48).

Third, we can also replace the logistic hidden layer Eq. (40) by the following Bernoulli hidden layer:

$$q(y|\xi,\ell)=\prod_{j=1}^{m_\ell}[s(\hat{y}_\ell^{(j)})\delta(y_\ell^{(j)})+(1-s(\hat{y}_\ell^{(j)}))\delta(1-y_\ell^{(j)})],\quad \hat{y}_\ell=W_\ell\xi+d_\ell.\tag{53}$$

In this case, similar to Eq. (46) we modify Eqs. (17) and (20) into

$$H(p\|q)=\frac{1}{N}\sum_{t=1}^{N}\ln Q_B(\xi_t,\eta_t,y_{\ell_t},\ell_t)-\ln z_q,\quad y_{\ell,t}=y_{\ell_t}(\xi_t),\quad \ell_t=\ell(\xi_t),$$

$$y_\ell(\xi)=\arg\max_{y}Q_B(\xi,\eta,y,\ell),\quad \ell(\xi)=\arg\max_{\ell}Q_B(\xi,\eta,y_\ell(\xi),\ell),$$

$$Q_B(\xi,\eta,y_{\ell,t},\ell_t)=G(\eta|A_\ell y_\ell+c_\ell,\sigma_\ell^2 I)q(y_\ell|\ell)q(\ell)\frac{q(y|\xi,\ell)}{p(y|\ell)},$$

$$p(y|\ell)=\frac{1}{N}\sum_{t=1}^{N}\prod_{j=1}^{m_\ell}s(\hat{y}_\ell^{(j)})^{y_\ell^{(j)}}(1-s(\hat{y}_\ell^{(j)}))^{1-y_\ell^{(j)}},\quad \hat{y}_\ell=W_\ell\xi+d_\ell. \tag{54}$$

Again, learning can be implemented by an adaptive learning procedure similar to Table 3, with $Q_0(\xi_t,\eta_t,\ell_t)$ replaced by $Q_B(\xi_t,\eta_t,y_{\ell,t},\ell_t)$ and Step 1 replaced by $y_\ell(\xi)$, $\ell(\xi)$ as given in Eq. (54), while Steps 2–5 can be directly adopted, with $r_{t,\ell}^d$, g_{d_ℓ}, g_{w_ℓ} given as follows:

$$g_{d_\ell}=\nabla_{g_{d_\ell}}\ln q(y|\xi,\ell)_{\xi=\xi_t}=y_{\ell,t}-s(\hat{y}_{\ell,t}),$$

$$g_{W_\ell}=\nabla_{W_\ell}\ln q(y|\xi,\ell)_{\xi=\xi_t}=g_{d_\ell}\xi_t^{\mathrm{T}},$$

$$r_{t,\ell}^d=\frac{q(y_{\ell_t}|\xi_t,\ell_t)}{S_{py}(t)},\quad S_{py}(t)=S_{py}(t-1)+\sum_{\ell=1}^{k}q(y_{\ell,t}|\xi_t,\ell), \tag{55}$$

Moreover, for $J(\mathbf{k})$ in Eq. (48), the term R_ℓ is replaced with

$$B_\ell=-\sum_{t=1}^{N}\sum_{j=1}^{m_\ell}[y_{\ell,t}^{(j)}\ln s(\hat{y}_{\ell,t}^{(j)})+(1-y_{\ell,t}^{(j)})\ln(1-s(\hat{y}_{\ell,t}^{(j)}))]. \tag{56}$$

6. Discussions on application to production rule mining

As mentioned in the beginning of this paper, forward network with one sigmoid hidden layer has been widely used on the tasks of pattern recognition, feature extraction, independent data analysis and statistical regression. Sharing the common key point that performs an expected input–output mapping, the results in the previous sections are directly applicable to these tasks.

In recent years, there have been increasing interests on automatic extracting knowledge in the form of production rules from data. One main stream of these studies bases on the feature that a sigmoid unit $s(\hat{y}^{(j)})$ with $y=W\xi+d$, $s(r)=1/(1+\mathrm{e}^{-\beta r})$ with $\beta\to\infty$ acts as a primitive predicate statement $p_j(\xi)$ that takes '1' for true and '0' for false, according to the value of ξ. Such a predicate statement can also be understood as an IF–THEN type rule. In a three layer net, each of the hidden sigmoid unit acts as such a primitive predicate and subsequently a sigmoid output unit further lumps them into a compound statement. Thus, under the control of $\beta\to\infty$, a three layer net trained by a supervised learning algorithm (e.g., back-propagation), results in a rule system.

For a hidden layer of m sigmoid units we have m primitive predicate statements in the form of $p_j(\xi)$. Theoretically, a compound statement on the m primitive statements encounters 2^m different situations since a logical conjunction $p_1(\xi)\cap\cdots\cap p_m(\xi)$ can have 2^m different status, which geometrically locate at the 2^m vertexes of a m dimension

hypercubic. However, a sigmoid output unit only forms a hyperplane that separates vertexes of two linear separable groups but be not able to separate vertexes in any arbitrary two groups. Thus, for a rule system to cover all the situations, the problem is that the extraction of the m primitive statements should be able to map two groups of samples of x into two sets of linear separable vertexes of the m dimension hypercubic. For this purpose, we have to use a large number m of hidden units.

However, a rule system empirically obtained on a set of training samples cannot avoid a paradox between conflict and redundancy. When m is too small, there will be certain conflicts among the obtained rules. As m increases, even when conflicts have been eliminated and the obtained rules demonstrate a desired performance on a given training set of finite number of samples, the problem cannot be really solved. There are two reasons. First, when there are noises added to samples, there will be always certain rules in conflict. Reducing conflicts via increasing m will not improve the performance but instead wastes resource on encoding noise. Second, the rules with a desired performance on a training set may still encounter new conflicts on new samples, i.e., the rules may not generalize well, especially when m is too large.

We believe that the paradox between conflict and redundancy is actually equivalent to the problem of regularization that prevents an over-fitting on a training set and the problem of model selection that decides an appropriate number of production rules. Thus, the BYY harmony learning provides a solution to the problems, not only via regularization in help of either normalization or data smoothing, but also via model selection made either automatically during learning by one of Tables 2 and 3 or through the selection criterion by one of Eqs. (33), (38), (34), (47), and (48).

The BYY unsupervised learning given in Section 4 also provides a solution on unsupervised rule discovery. With $\beta \to \infty$, m primitive statements are discovered via $s(\hat{y}_\ell)$ in Eq. (34). Moreover, these statements become independent from each other, which thus minimizes the dependence and conflict among them. With these obtained primitive statements, we can form certain compound statements in a subsequent processing, and interpret how each sample of x is generated. As a group of independent primitive statements is obtained for each ℓ, we got k groups equivalently in a logical disjunction. Moreover, it follows from the discussion made at the end of Section 5.2 that the above arguments apply to the supervised case, too. That is, the paradox between conflict and redundancy is also being tackled by making the statements of the hidden units become independent.

Technically, the larger of the slant parameter $\beta > 0$, the better a sigmoid unit acts as a logical predicate statement. However, the learning may be trapped into a local optimal solution for a large $\beta > 0$. One solution to this problem is a simulated annealing procedure with $1/\beta$ acting as a temperature [16]. Moreover, we also impose the constraint $\sqrt{\|a\|^2 + b^2} = 1$ on a sigmoid unit $s(a^{\mathrm{T}}x + b)$ to remove the indeterminacy $a^{\mathrm{T}}x + b = c(a'^{\mathrm{T}}x + b')$.

Furthermore, for the rule extraction purpose, in Table 3 we can use $q(\eta|y)$ by Eq. (49) in Step 1 and use Eq. (50) as Step 5, with Eq. (51) put in Eqs. (47) and (49). Similarly, when each component $x^{(j)}$ takes the value of 'true' or 'false', we can make learning as described in Section 4.3.

Acknowledgements

The author would like to express thanks to the reviewers for their comments that help a lots on improving the original manuscript.

References

[1] H. Akaike, A new look at the statistical model identification, IEEE Trans. Automat. Control 19 (1974) 714–723.
[2] T.W. Anderson, H. Rubin, Statistical inference in factor analysis, Proceedings of the Berkeley Symposium on Mathematical and Statistical Probability 3rd 5, UC Berkeley, 1956, pp. 111–150.
[3] H.B. Barlow, Unsupervised learning, Neural Comput. 1 (1989) 295–311.
[4] C.M. Bishop, Training with noise is equivalent to Tikhonov regularization, Neural Comput. 7 (1995) 108–116.
[5] H. Bourlard, Y. Kamp, Auto-association by multilayer perceptrons and singular value decomposition, Biol. Cybernet. 59 (1988) 291–294.
[6] H. Bozdogan, Model selection and akaike's information criterion: the general theory and its analytical extension Psychometrika 52 (1987) 345–370.
[7] K.C. Chiu, L. Xu, A comparative study of Gaussian TFA learning and statistical tests for determination of factor number in APT, Proceedings of IJCNN '02, Honolulu, Hawaii, USA, May 12–17, 2002, pp. 2243–2248.
[8] P. Dayan, R.S. Zemel, Competition and multiple cause models, Neural Comput. 7 (1995) 565–579.
[9] A.P. Dempster, et al., Maximum-likelihood from incomplete data via the EM algorithm, J. Roy. Statist. Soc. B39 (1977) 1–38.
[10] L. Devroye, et al., A Probability Theory of Pattern Recognition, Springer, Berlin, 1996.
[11] G.E. Hinton, R.S. Zemel, Autoencoders, minimum description length and Helmholtz free energy, Adv. NIPS 6 (1994) 3–10.
[12] R.A. Jacobs, et al., Adaptive mixtures of local experts, Neural Comput. 3 (1991) 79–87.
[13] M.I. Jordan, R.A. Jacobs, Hierarchical mixtures of experts and the EM algorithm, Neural Comput. 6 (1994) 181–214.
[14] M.I. Jordan, L. Xu, Convergence results for the EM approach to mixtures of experts, Neural Networks 8 (1995) 1409–1431.
[15] J. Karhunen, J. Joutsensalo, Representation and separation of signals using nonlinear PCA type learning, Neural Networks 7 (1994) 113–127.
[16] S. Kirkpatrick, et al., Optimization by simulated annealing, Science 220 (1983) 671–680.
[17] Z.B. Lai, P. Guo, T.J. Wang, L. Xu, Comparison on Bayesian Ying-Yang theory based clustering number selection criterion with information theoretical criteria, Proceedings of IJCNN98, Vol. I, Anchorage, Alaska, May 5–9, 1998, pp. 725–729.
[18] W.K. Lam, L. Xu, An experimental comparison of the Bayesian YING-YANG criteria and cross validation for selection on number of hidden units in feedforward networks, Proceedings of ICASSP98, Vol. 2, Seattle, WA, May 12–15, 1998, pp. 1189–1192.
[19] W.K. Lam, L. Xu, An experimental comparison of the Bayesian Ying-Yang criteria and cross validation on experts number selection in original and alternative model for mixture of experts. Proceeding of ICONIP'98, Vol. 1, Kitakyushu, Japan, October 21–23, 1998, pp. 71–74.
[20] Z.Y. Liu, K.C. Chiu, L. Xu, Local subspace analysis for astronomical object detection and star/galaxy classification, Proceedings of IJCNN'02, Honolulu, Hawaii, USA, May 12–17, 2002, pp. 962–967.
[21] D.J.C. Mackey, A practical Bayesian framework for backpropagation, Neural Comput. 4 (1992) 415–447.
[22] O. Ning, W.K. Lam, K. Yamauchi, L. Xu, Using an improved back propagation learning method to diagnose the sites of cardiac, hypertrophy, MD Comput. 16 (1) (1999) 79–81.
[23] V. Ramamurti, J. Ghosh, Regularization and error bars for the mixture of experts network, Proceedings of IEEE ICNN97, 1997, pp. 221–225.

[24] R.A. Redner, H.F. Walker, Mixture densities, maximum likelihood, and the EM algorithm, SIAM Rev. 26 (1984) 195–239.
[25] J. Rissanen, Stochastic complexity, J. Roy. Statist. Soc. 49 (3) (1987) 223–239.
[26] J. Rissanen, Stochastic Complexity in Statistical Inquiry, World Scientific, Singapore, 1989.
[27] I. Rivals, L. Personnaz, On cross validation for model selection, Neural Comput. 11 (1999) 863–870.
[28] D.E. Rumelhart, et al., Learning internal representations by error propagation, Parallel Distributed Processing, Vol. 1, MIT Press, Cambridge, MA, 1986.
[29] E. Saund, A multiple cause mixture model for unsupervised learning, Neural Comput. 7 (1995) 51–71.
[30] A.N. Tikhonov, V.Y. Arsenin, Solutions of Ill-posed Problems, V.H. Winston and Sons, Washington, DC, 1977.
[31] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, Berlin, 1995.
[32] C.S. Wallace, D.M. Boulton, An information measure for classification, Comput. J. 11 (1968) 185–194.
[33] C.S. Wallace, D.R. Dowe, Minimum message length and Kolmogorov complexity, Comput. J. 42 (4) (1999) 270–280.
[34] L. Xu, Least mean square error reconstruction for self-organizing neural-nets, Neural Networks 6 (1993) 627–648. Its early version on Proceedings of 1991 International Joint Conference on Neural Networks (IJCNN91' Singapore), 1991, pp. 2363–2373.
[35] L. Xu, Bayesian Ying-Yang machine, clustering and number of clusters, Pattern Recognition Lett. 18 (11–13) (1997) 1167–1178.
[36] L. Xu, RBF nets, mixture experts, and Bayesian Ying-Yang learning, Neurocomput. 19 (1–3) (1998) 223–257.
[37] L. Xu, Temporal BYY learning for state space approach, hidden Markov model and blind source separation, IEEE Trans. Signal Process. 48 (7) (2000) 2132–2144.
[38] L. Xu, BYY harmony learning, independent state space and generalized APT financial analyses, IEEE Trans. Neural Networks 12 (4) (2001) 822–847.
[39] L. Xu, Bayesian Ying Yang harmony learning, in: M.A. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, 2nd Edition, The MIT Press, Cambridge, MA, 2002, in press.
[40] L. Xu, C.C. Cheung, S.I. Amari, Learned parametric mixture based ICA algorithm, Neurocomput. 22(1–3) (1998) 69–80. A part of its preliminary version on Proceedings of ESANN97, Bruges, April 16–18, 1997, pp. 291–296.
[41] L. Xu, M.I. Jordan, G.E. Hinton, An alternative model for mixtures of experts, Adv. Neural Inform. Process. Systems, 7 (1995) 633–640. Its preliminary version on Proc. WCNN'94 2 (1994) 405–410.

Lei Xu (IEEE Fellow) is a professor of Computer Science and Engineering at Chinese Univ Hong Kong (CUHK). He is also an adjunct Professor at Peking University and three other universities in China and UK. After receiving his Ph.D. from Tsinghua Univ. in early 1987, he joined Peking Univ. where he became one of ten university-level exceptionally promoted young associate professors in 1988 and further been exceptionally promoted to a full professor in 1992. During 1989–1993, he worked at several universities in Finland, Canada and USA, including Harvard and MIT. He joined CUHK in 1993 as a senior lecturer, then became promoted to professor in 1996 and took the current chair professor position in 1996. Prof. Xu has published over 240 academic papers, with a number of them well cited in literature. He has given a number of keynote/plenary/invited/tutorial talks in international major Neural Networks (NN) conferences, such as WCNN, IEEE-ICNN, IJCNN, ICONIP, etc. He is on the Governor Board of International NN Society, a past president of Asia-Pacific NN Assembly, the chair of the Computational Finance Technical Committee of IEEE Neural Networks Society, and an associate editor for six International journals on NN, including Neural Networks, IEEE Trans. Neural Networks. He is the general chair of IEEE CIFER'03 and a program committee co-chair of Joint ICANN'03-iCONiP'03, and was a ICONIP'96 program committee chair and a general chair of IDEAL'98, IDEAL'00. Also, he has served as program committee members in international major NN conferences in the past decade, including IJCNN (97,99,00,01), WCNN(95,96), IEEE-ICNN (96), etc. He has received several Chinese national prestigious academic awards (including the National Nature Science Prize) and also some international awards (including an 1995 INNS Leadership Award). Prof. Xu is a Fellow of IEEE and a Fellow of the International Association for Pattern Recognition.