in N. Zhong and J. Liu (eds), Intelligent Technologies for Information Analysis, Springer, pp615-659, 2004

22. Bayesian Ying Yang Learning (I): A Unified Perspective for Statistical Modeling

Lei Xu

Chinese University of Hong Kong, Hong Kong

Abstract

Major dependence structure mining tasks are overviewed from a general statistical learning perspective. Bayesian Ying Yang (BYY) harmony learning has been introduced as a unified framework for mining these dependence structures, with new mechanisms for model selection and regularization on a finite size of samples. Main results are summarized and bibliographic remarks are made. Two typical approaches for implementing learning, namely optimization search and accumulation consensus, are also introduced.

22.1 Introduction: Basic Issues of Statistical Learning

Statistical learning is a process by which an intelligent system $\mathcal{M}$ estimates the underlying distribution and dependence structures among what is to be learned (i.e., among the world $\mathbf{X}$ that $\mathcal{M}$ observes). Specifically, learning is done via a certain medium that is usually a set of samples from the world $\mathbf{X}$. Statistical learning plays an essential role in the literature of neural networks. It also acts as a major part in the field of intelligent data processing, or called *data mining*, with an ever increasing popularity.

According to how a learning medium is used, studies on learning can be further classified into two quite different domains. One is the so-called active learning, with a feature that the strategy of getting samples varies according to the current status of $\mathcal{M}$. The other is the conventional statistical learning, during which samples come from the world according to its underlying distribution in help of a given sampling strategy that is independent of the status of $\mathcal{M}$. This paper focuses on the latter. In this case, learning is characterized by the following basic issues, or ingredients:

– A world $\mathbf{X}$, in the form of an underlying distribution with certain dependence structures.
– A machine $\mathcal{M}$ in an appropriate architecture that is able to accommodate or describe the underlying distribution.
– A finite size set $\mathcal{X} = \{x_t\}_{t=1}^N$ of random samples that comes from the underlying distribution.
– A learning principle or theory that coordinates the above three issues in order to get the best estimate by $\mathcal{M}$ of the underlying distribution based on $\mathcal{X}$.
– An efficient algorithm that implements the above learning theory

Though the entire world faced by $\mathcal{M}$ is usually very complicated, we can conduct studies by decomposing a complicated task into a number of learning tasks on much simplified small worlds, as discussed in the next two subsequent sections.

22.2 Dependence Among Samples from One-Object World

We start by considering a simple world $\mathbf{X}$ of only a single object that is observed via a set $\mathcal{X}$ of samples, with each sample $x = [x^{(1)}, \cdots, x^{(d)}]^T$ from the same underlying probability distribution or density $p(x)$. Typically, we can use one or more of the following five ways to describe the dependence structures among $\mathbf{X}$.

(a) Nonparametric joint density In general, dependence among variables $x^{(1)}, \cdots, x^{(d)}$ can always be described by a nonstructural or nonparametric joint density. Typical example are the empirical density:

$$p_0(x) = \tfrac{1}{N}\textstyle\sum_{t=1}^{N}\delta(x - x_t), \ \ \delta(x) = \begin{cases} \lim_{\delta\to 0}\frac{1}{\delta^d}, & x = 0, \\ 0, & x \neq 0, \end{cases} \tag{22.1}$$

where d is the dimension of x and $\delta > 0$ is a small number. This $p_0(x)$ is equivalent to memorizing and using a given data set $\mathcal{X}$ directly. Another typical example is a non-parametric Parzen window density estimate [22.10]:

$$p_h(x) = \tfrac{1}{N}\textstyle\sum_{t=1}^{N}K_h(x, x_t), K_h(x, x_t) \text{ is a kernel located at } x_t, \tag{22.2}$$

which means to use the data set $\mathcal{X}$ after certain smoothing by $K_h(x, x_t)$. In the simplest case, $K_h(x, x_t)$ is a hyper cubic of volume h^d with its center located at x_t, and $p_h(x)$ becomes the widely used histogram estimate that is a smoothed version of Eq. (22.1). The smoothness is controlled by a given parameter $h > 0$ that is called *smoothing parameter*. The other case is $K_h(x, x_t) = G(x|x_t, h^2 I)$ and

$$p_h(x) = \tfrac{1}{N}\textstyle\sum_{t=1}^{N}G(x|x_t, h^2 I), \tag{22.3}$$

where, and hereafter in this paper, $G(x|m, \Sigma)$ denotes a Gaussian density with mean vector m and covariance matrix Σ.

Such a nonparametric and non-structural joint density estimate, though conceptually implying all dependence relations among variables, has three major weak points. One is that it is usually a bad estimate when the size N of samples is finite, especially when the dimension d of x is high. The second is that dependence relations are not given directly. The third is the expensive computing cost for each calculation on $p(x)$ at any point $x = x'$.

To improve the weak points, efforts are made along two directions.

(b) Sample statistics The first direction is to describe dependencies among variables $x^{(1)}, \cdots, x^{(d)}$ collectively by sample statistics up to certain orders, instead of a joint density that conceptually implies statistics of all possible orders. The most widely encountered case is statistics up to the second order only, represented in the covariance matrix $\Sigma_x = E(x - m_x)(x - m_x)^T, m_x = Ex$, capturing the linear dependence structures among all the variables subject to the mean square error, where (and throughout this chapter) the notations $E(u) = Eu = E[u]$ denotes the expectation of the random variable u. Equivalently, this case actually assumes that the underlying distribution is a parametric Gaussian density with parameters m_x and Σ_x. Of course, we can also similarly consider higher order statistics. However, the number of such statistics increases exponentially, e.g., in the order of d^m for the m-th order statistics, which brings us to the weak points, similar to those for estimating a nonparametric density.

(c) Co-occurrence and associations The second direction is to focus on the pair-wise dependencies between u and v that denote two or more subsets of variables $x^{(1)}, \cdots, x^{(d)}$. The simplest one is co-occurrence or association. That is, we are interested in finding those events $u = U$ and $v = V$ that will co-occur with high probability, e.g., with the corresponding probability $P(u = U, v = V)$ higher than a pre-specified threshold. We are also interested in an association rule $(u = U) => (v = V)$ that describes how we can infer the occurrence of the event $v = V$ from the occurrence of the event $u = U$, which is measured by its `support` $P(u = U, v = V)$ and `confidence` $P(v = V|u = U)$, as defined in *data mining* literature for the market basket analysis (Chap. 7 in [22.17]).

We can see that the task of association rule mining is actually a task simplified from estimating the joint density $p(x)$. It is not necessary, and also expensive, to estimate $p(x)$. What we are interested in are only those events $u = U$ and $v = V$ with $P(u = U, v = V) > s$ and $P(v = V|u = U) > c$ for the pre-specified thresholds $s > 0$ and $c > 0$. In addition to those approaches given in (Chap. 7 in [22.17]), we suggest a stochastic approach with its key idea as follows:

– Find a way such that all the possible values of U and V can be sampled with an equal probability, which is referred as random sampling;
– Make n_s/s such random samples; if a specific pair, $\bar{U}$ and $\bar{V}$, appears more than $s \times n_s/s = n_s$ times, we get $(u = \bar{U}) => (v = \bar{V})$ as a candidate association rule;
– Conditioning on $u = \bar{U}$, make n_c/c random samples on V; if the specific value $\bar{V}$ appears more than $c \times n_c/c = n_c$ times, we take $(u = \bar{U}) => (v = \bar{V})$ as an association rule with its support larger than s and its confidence larger than c.

(d) Regressions or fittings Instead of considering the conditional probability $P(v = V|u = U)$, we may also directly consider the average or expected relation $E(v|u) = \int vP(v|u)dv$, which is non-probabilistic function $v = f(u)$ and usually called regression. The regression can be either linear or nonlinear. Particularly, when v is Gaussian from $G(v|f(u), \sigma_v^2 I)$, the function $f(u)$ fits a set of pairs $\{u_t, v_t\}_{t=1}^N$ in a sense that the average square fitting error $\frac{1}{N}\sum_{t=1}^N \|v_t - f(u_t)\|^2 \approx \sigma_v^2$ is minimized.

(e) Linear and nonlinear generative structures We can also explore dependencies among all the variables $x^{(1)}, \cdots, x^{(d)}$ in help of inner representations via certain hidden or latent structures. One typical structure is the following explicit stochastic linear function:

$$x = Ay + e, \; E(e) = 0, \; e \text{ is independent of } y. \tag{22.4}$$

The earliest effort on this linear model can be traced back to the beginning of the 20th century by Spearman [22.41], and has been followed by various studies in the literature of statistics. In this model, a random sample x of observations is generated via a linear mapping matrix A, from k inner representations or hidden factors in the form $y = [y^{(1)}, \cdots, y^{(k)}]^T$ from a parametric density $q(y)$, disturbed by a noise e. Generally, samples of e are independently and identically distributed (i.i.d.) from a parametric density $q(e)$, and the function forms of $q(y)$ and $q(e)$ are given. The matrix A and the statistics of y_t and e_t are to be learned.

The problem is usually not well defined because there are an infinite number of solutions. To reduce the indeterminacy, we assume that samples of x and e are i.i.d., and, correspondingly, samples of y are also i.i.d. such that dependence among variables of x is equivalently modeled by

$$q(x) = \int q(x - Ay) q(y) dy, \tag{22.5}$$

which implies a parametric density of x, described via the matrix A, the statistics of y and e, and the density forms of $q(y)$ and $q(e)$. Also, Eq. (22.4) implies that all the statistics of x are subject to the constraint. For example, with $\Sigma_x, \Sigma_y, \Sigma_e$ being covariance matrices of x, y, and e, respectively, we have

$$\Sigma_x = A^T \Sigma_y A + \Sigma_e. \tag{22.6}$$

Particularly, when y is Gaussian and uncorrelated with e, e also being uncorrelated with its components, the model Eq. (22.4) is called factor analysis [22.32, 22.39], which was first formulated in [22.3]. In this case, the integral $q(x)$ is analytically solvable, and becomes simply Gaussian.

As discussed in [22.57], this constraint, Eq. (22.6), is not enough to uniquely specify A and Σ_e. If we further impose the constraint

$$A = \phi D, \phi^T \phi = I, \; \Sigma_y = I, \; \Sigma_e = \sigma_e^2 I, \tag{22.7}$$

it follows [22.3, 22.66] that the maximum likelihood learning on $q(x)$ results in ϕ consisting of the k principal eigenvectors of Σ_x and D^2 consisting of the corresponding principal eigenvalues. That is, it becomes equivalent to the so-called principal component analysis (PCA). In recent years, such a special case of factor analysis has also been reiterated in the literature of neural networks under a new name, probabilistic PCA [22.45].

When y is non-Gaussian, the Eq. (22.4) implies not only the constraint Eq. (22.6), but also constraints on higher order statistics such that the indeterminacy on A and Σ_e may be removed. However, the integral $p(x)$ by Eq. (22.5) becomes analytically unsolvable when y is real and non-Gaussian.

Generally, Eq. (22.4) can also be either a nonlinear structure

$$x = g(y, A) + e, \quad q(x) = \int q(x - g(y, A))q(y)dy, \tag{22.8}$$

or a general probabilistic structure

$$q(x) = \int q(x|y)q(y)dy. \tag{22.9}$$

Actually, the above two cases are regarded as being equivalent via the link

$$g(y, A) = E(x|y) - Ee, \quad e = x - g(y, A), \tag{22.10}$$

where e generally relates to y, instead of being independent of y.

(f) Linear and nonlinear transform structures As a dual to generative structure, another typical structure for describing dependence among variables is a forward transform or mapping, $y = f(x, W)$ to inner representations of y.

The simplest case is $y = Wx$ when $Ex = 0$ (otherwise the mean can be subtracted in preprocessing). It specifies a relation $Eyy^T = \Sigma_y = W^T \Sigma_x W$ between the second order statistics of y and x, such that dependence among variables of x are implicitly specified via W and the statistics of y. When Σ_y becomes diagonal, the mapping $y = Wx$ is said to de-correlate the components of x. Moreover, $\max_{W^TW=I} E\|y\|^2, \ E\|y\|^2 = Tr[\Sigma_y]$, leads to the well known principal component analysis (PCA) that extracts the principal second order statistics. The study on PCA can be backtracked to as early as in 1936 [22.19], and has been also widely studied in the literature of statistics, pattern recognition, and neural networks [22.34]. Equivalently, PCA is also reached under the constraint $W^TW = I$ by maximizing the following entropy:

$$\max J(W), \ J(W) = -\int p(y) \ln p(y)dy, \tag{22.11}$$

when x comes from Gaussian and $y = Wx$ is still Gaussian.

Moreover, when $y = Wx$ satisfies

$$q(y) = \prod_{j=1}^{k} q(y^{(j)}) \tag{22.12}$$

with at most one component being Gaussian, the mapping $y = Wx$ is said to implement independent component analysis (ICA) [22.15, 22.46, 22.7] that extracts statistics of the second order and those higher orders. Also, when each component of y_t is interpreted as a sample of a time series at time t, the ICA solution $\hat{y}_t = Wx_t$ recovers y_t from $x_t = Ay_t$, up to a scaling indeterminacy. That is, the waveform of each component series can be recovered by $\hat{y}_t = Wx_t$, by a process also called blind source separation (BSS) that blindly separates the mixed signal $x_t = Ay_t$, with scaling indeterminacy [22.46].

However, an ICA algorithm with $q(y^{(j)})$ pre-fixed works well only on the cases where either all components of y are super-Gaussians or all components of y are sub-Gaussians [22.2, 22.4], but not on the cases where components of y are partly super-Gaussians and sub-Gaussians. This problem can be solved by learning $q(y^{(j)})$ via a mixture of parametric densities, or equivalently learning the cumulative distribution function (cdf) of $q(y^{(j)})$ by a mixture of parametric cdfs [22.87, 22.72, 22.73]. Efforts have also been made on using a nonlinear mapping $y = f(x, \theta)$ to implement a nonlinear ICA [22.42, 22.66]. However, the satisfaction of Eq. (22.12) will remain unchanged after any component-wise nonlinear transformation on y. Thus, a nonlinear ICA mapping $y = f(x, \theta)$ usually does not retain the feature of performing BBS, since it is no longer able to recover the original y up to scaling indeterminacy, unless in a specific situation with extra constraints imposed [22.42].

Instead of considering nonlinear ICA, we consider in [22.53] a nonlinear mapping $y = f(x, \theta)$ from the perspective of modeling the cumulative distribution function (cdf) of input data x.

22.3 Dependence Among Samples from a Multi-Object World

22.3.1 Dependence Among Samples from a Multi-Object World

We observe a world $\mathbf{X}$ with multiple objects that are either visible or invisible. We start by considering the cases where all the objects are visible, with each label ℓ in a set L denoting a specific object observed via a sample vector $x_\ell = [x_\ell^{(1)}, \cdots, x_\ell^{(d)}]^T$.

We can discover dependence structures within each object $\ell \in L$ individually, in the same ways introduced in the previous subsection. Moreover, dependence structures also exist across different objects. They are described both qualitatively by the topology of L, and quantitatively by dependence structures among variables of x_ℓ across objects.

In those most complicated cases, where there are dependence structures between any pair of objects, the topology of L is a complete graph with every pair of nodes connected. Specifically, if there is no dependence between a pair of objects, the connection between the corresponding two nodes is broken, and can, thus, be removed. Usually, three types of simplified topology are often encountered, described as follows:

(a) *A linear or serial chain* The simplest case is that the topology is a simple chain, $1, 2, \cdots, \ell - 1, \ell, \ell + 1, \cdots$, with the object ℓ directly connecting to only the object $\ell - 1$ and the object $\ell + 1$. In this case, $x_1, x_2, \cdots, x_{\ell-1}, x_\ell, x_{\ell+1}, \cdots$ form a sequence called a time series, where ℓ

denotes the time. The task of learning is to estimate quantitatively the dependence structures by the joint distribution of the whole sequence in a certain serial/temporal dependence structure. The task is often encountered in signal processing and stochastic process modeling, such as time series prediction and speech, audio, and text processing, via AR and ARMA models and, especially, via the Hidden Markov model [22.36].

(b) *An image and a d-dimensional lattice topology* We further consider the cases where labels in L are organized in a regular lattice topology, with each object ℓ denoted by a coordinate $(\ell_1, \ell_2, \ldots, \ell_d)$. It reduces to the above chain and time series model when $d = 1$. Moreover, $\{x_\ell : \ell \in L\}$ denotes an image when $d = 2$ and a 3D array or video when $d = 3$, etc., x_ℓ being a pixel. Similarly, the task of learning is to estimate the dependence structures by the joint distribution of all the pixels (usually with the help of certain dependence structures between a pixel and others in its neighborhood), which actually leads to a stochastic field. Such tasks are often encountered in image processing and computer vision.

(c) *A tree topology* Another typical case is where L has a tree topology, with each object ℓ being a node on the tree, i.e., each node has direct dependence only on its father or children. The task of learning is to estimate the dependence structures by the joint distribution $\{x_\ell : \ell \in L\}$ subject to this tree dependence structure. This is a typical task that has been studied as the popular topic of probabilistic graphical model or Belief networks in the recent literature of AI, statistics, and neural networks; readers are referred to the books [22.35, 22.22].

(d) *A joint temporal topology and spatial topology* A topology that combines the above case (a) with either of the case (b) and case (c) is also widely studied in the literature. Here, a topology of either of image, d-lattice, and tree describes spatial relations among different objects, while a linear topology describes temporal relations of each object that changes with time, such as encountered in video processing.

22.3.2 Mining Dependence Structure Across Invisible Multi-Object

Next, we further observe a world of invisible multiple objects, with each sample vector x coming from one object $\ell \in L$, but with its label ℓ missing.

Given a set of such samples, we want to discover the dependence structures among the objects, which are again described both qualitatively by the topology of L and quantitatively by the dependence structures among variables of x within and across objects. Still, the tasks of learning can be classified according to what types of topology of L are taken into consideration.

(a) *Finite Mixture* The simplest case is that we ignore the topology of L, and consider only dependence structures among the variables of x, which depends on the dependence structure between each sample x and each label ℓ. The general form of the dependence is described by a joint distribution $q(x, \ell)$

that describes the joint occurrence of the events that a sample is valued by x and that the sample comes from the object ℓ.

Specifically, $q(x,\ell)$ can be represented via the decomposition $q(x|\ell)q(\ell)$, which describes $q(x,\ell)$ as a casual view on where x comes from. Each of $q(x|\ell), \ell = 1, \cdots, k$, describes the dependence structures within the ℓ-th object, and

$$q(\ell) = \sum_{j=1}^{k} \alpha_j \delta(\ell - j), \text{ with the constraint } \alpha_\ell \geq 0, \sum_{\ell=1}^{k} \alpha_\ell = 1, \quad (22.13)$$

where α_ℓ denotes a priori probability that x comes from the ℓ-th object.

The task of learning is to estimate α_ℓ, and to discover dependence structures within each $q(x|\ell)$, which is equivalent to learning dependence structures in the format

$$q(x) = \sum_{\ell \in L} \alpha_\ell q(x|\ell). \quad (22.14)$$

It is usually called finite mixture in the literature [22.9, 22.38, 22.33]. It is also said to have modular structure in a sense that there are a number of individual modules in an assembly.

As a by-product, we also have the Bayesian rule

$$q(\ell|x) = \alpha_\ell q(x|\ell)/q(x), \quad (22.15)$$

and, thus, another decomposition, $p(x,\ell) = p(\ell|x)p(x)$ with $p(\ell|x)$ describing an inference view on which object the observation x may come from. It results in a partition of a set of samples into different objects, and, thus, is usually called pattern recognition or clustering analysis [22.10]. Particularly, when the dependence structure of $q(x|\ell)$ is simply a Gaussian $q(x|\ell) = G(x|\mu_\ell, \Sigma_\ell)$, Eq. (22.14) becomes the widely used Gaussian mixture [22.9, 22.38, 22.33], and Eq. (22.15) is called Bayesian classifier [22.11]. Moreover, when $\Sigma_\ell = \sigma^2 I$ and $\alpha_\ell = 1/k$, a hard-cut version of Eq. (22.15) leads to the conventional least square clustering [22.77].

(b) *Self-organizing map* We further consider the cases where L has a given regular d-dimensional lattice topology. Since the label ℓ associated with x is invisible, we are not able to recover the dependence structures among the variables of x across different objects. Alternatively, we re-establish dependence structures according to *a general belief that objects locating topologically in a small neighborhood N_ℓ should be same as or similar to each other*, where a small neighborhood N_ℓ of a knot ℓ usually consists of 2^d knots that are directly connected to ℓ.

The direct placement of all the objects on such a lattice, according to a criterion or measure to judge whether two objects are same or similar, is computationally an NP-hard problem. Instead, this placement can be implemented approximately. Interestingly, a good approximation for this problem is provided by biological brain dynamics of self-organization [22.31], featured by a Mexican hat-type interaction, namely, neurons in near neighborhood excite each other with learning, while neurons far away inhibit each other with de-learning. Computationally, such a dynamic process can be simplified by certain heuristic strategies. Here, we consider two typical ones.

– *One member wins, a family gains* That is, as long as one member wins in the winner-take-all competition, all the members of a family gain, regardless of whether other members are strong or not. This direction is initialized by a simple and clever technique, i.e., the well known Kohonen self-organizing map [22.23]. In the literature, a great number of studies have been made on extending the Kohonen map. Recently, a general formulation of this strategy has also been proposed [22.54].
– *Stronger members gain and then team together* That is, a number of stronger members in competition will be picked as winners, who not only gain learning but are also teamed together to become neighbors [22.54]. It can speed up self-organization, especially in the early stages of learning. Also, we can combine the first and the second strategies by using the second in the early stage, and subsequently switching to the first, which is experimentally demonstrated in [22.8]

(c) *Self-organizing graphical topology* Following a similar line, we can also extend the above studies to a more complicated but given topology of either a tree or a general graph. The only difference is that a small neighborhood N_ℓ of a specific ℓ consists of all the knots that are directly connected to ℓ. and that the number of elements in N_ℓ is not usually 2^d, but equal to the degree of the node ℓ.

(d) *Dynamic self-organizing map* Another useful but even more complicated case is a combination of a temporal topology (i.e., a line topology) with either of the above cases (a), (b) and (c), which describes not only relations across objects but also how the relations change with time.

22.4 A Systemic View on Various Dependence Structures

Those dependence structures discussed in Sect. 1.2 and Sect. 1.3 can be systematically summarized according to which inner representation is adopted and what kind of inner architectures is used.

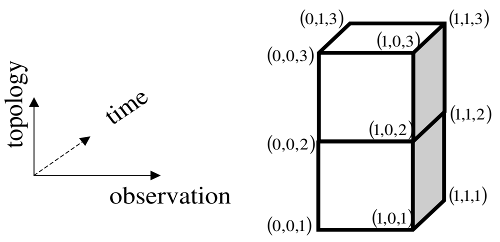
Figure 22.1 shows the simplest family F_0, by considering an object as a whole without considering inner representation and inner architecture. The types of dependence here can be classified according to differences in three features. First, focus is put whether on dependencies among all the components of observation as a whole or on the relation between two particular parts of observation. Second, whether temporal dependencies among samples are ignored or considered. Third, whether multiple objects are jointly

The F_0 Family

This F_0 is featured by (0,0) with the 1st '0' denoting a 0-architecture, i.e., an object or distribution is considered as a whole without considering architecture, with the 2nd '0' denoting that no inner representation y is considered.

Different types of dependences in this family come from three features
(*observation, time, topology*)

0 for *observation:*	**dependence among components of x are considered as a whole**
1 for *observation:*	**dependence between two parts (ξ_t, ς_t) of $x_t = (\xi_t, \varsigma_t)$ is focused**
0 for *time:*	**no dependence between any two samples of x from different times**
1 for *time:*	**dependences among samples from different times are considered**
1 for *topology:*	**only a single object is considered**
2 for *topology:*	**multiple objects without topology are considered**
3 for *topology:*	**multiple objects are considered to be mapped onto a topology**



(observation, time, topology)

T_1	T_2	T_3
(0,0,1)	(0,0,2)	(0,0,3)
(0,1,1)	(0,1,2)	(0,1,3)
(1,0,1)	(1,0,2)	(1,0,3)
(1,1,1)	(1,1,2)	(1,1,3)

Fig. 22.1. The family F_0

considered without or with a topology. As shown in Fig. 22.1, there are 12 combinatorial types in three groups. T_1 consists of

- $(0, 0, 1)$ (e.g., either a nonparametric density or a simple Gaussian density),
- $(1, 0, 1)$ (e.g., co-occurrence, association, regression and fitting etc.),
- $(0, 1, 1)$ (e.g., with time added, a Gaussian density is extended to a conventional AR, ARMA models),
- $(1, 1, 1)$ (e.g., multi-channel AR, ARMA models).

Moreover, T_2 extends those of T_1 to jointly consider multiple models but ignoring topology, consisting of

- $(0, 0, 2)$ (e.g., finite mixture or Gaussian mixture),
- $(1, 0, 2)$ (e.g., mixture-of-experts [22.12, 22.13, 22.14]),
- $(0, 1, 2)$ (e.g., finite mixture of AR, ARMA models [22.25, 22.48, 22.49, 22.43]),
- $(1, 1, 2)$ (e.g., mixture-of-experts of AR, ARMA models [22.44]).

Furthermore, T_3 extends those of T_1 to jointly consider multiple models and also map them onto a regular topology, with $(0, 0, 3)$ as a self-organizing map (SOM) and other three are further extensions of SOM.

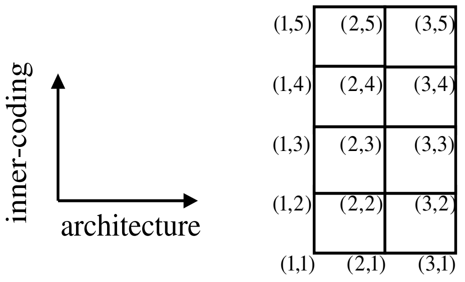
Figure 22.2 shows more sophisticated alternatives of F_0, featured by using what type of five typical inner representations and which of three major architectures, indicated by a code $(i, j), i = 1, 2, 3, j = 1, 2, 3, 4, 5$. With this notation, F_0 can be regarded as a degenerated case with the code $(0, 0)$. Different types of inner representations encode different types of dependencies and, thus, perform different pattern recognition and information processing tasks. A same task can be implemented by either of three architectures. As discussed in Sect. 2.1, a forward structure directly implements the mapping $x \to y$ per sample and indirectly describes data distribution in a differential manner, while a backward structure directly describes data distribution in an integral manner and indirectly implements the mapping $x \to y$ with an expensive computing cost. A bi-directional architecture trades off the features of the two such that not only modeling data distribution and implementing $x \to y$ can be both directly made but also each of the two provides a structural constraint to the other as a role of regularization (see Item 3.4 in [22.74]). The least mean square error reconstruction learning [22.94] is a simple example of a BI-architecture that combines a sigmoid post-linear forward structure and a linear backward structure by Eq. (22.4). In fact, this is the key sprit of Bayesian Ying Yang system. The details will be given the subsequent sections.

Types of dependences, under the perspective of BYY system, can be understood by five families featured by a combination of the following two features:

(*architecture, inner-coding*)

architecture: **1** Forward **2** Backward **3** Bi-directional

inner-coding: **1** Gaussian $[y_1, \cdots, y_m]$ **2** nonGuassian $[y_1, \cdots, y_m]$ **3** binary $[y_1, \cdots, y_m]$ **4** $y=1, \cdots, k$ **5** known



F_1 :	(1,1)	(1,2)	(1,3)
F_2 :	(2,1)	(2,2)	(2,3)
F_3 :	(3,1)	(3,2)	(3,3)
F_4 :	(4,1)	(4,2)	(4,3)
F_5 :	(5,1)	(5,2)	(5,3)

Each family F_i consists of three sub-families with each in a format (*,*)

Each subset (i,j) consists of types of dependence with at most 15 choices, featured by

(observation, time, topology)

In a total, ***(architecture, inner-coding) (observation, time, topology)*** has at most

$12 + 15 \times 12$ choices

Fig. 22.2. Types of Dependence from BYY System Perspective

As shown in Fig. 22.2, we can further group the 3×5 combinations into five families according to five types of inner representations, with each family consisting of three sub-families featured by each of three architectures. From

F_1 to F_5, constraints on the inner representation are gradually enhanced from de-correlated to becoming independence and then to uniform or known such that indeterminacy of the structures is gradually reduced. The details are briefed as follows:

(a) $F_1 = \{(1,1),(2,1),(3,1)\}$ is featured by a Gaussian vector $y = [y_1, \cdots, y_k]$ with its components de-correlated from each other. Typical examples of $(1,1)$ includes PCA, MCA, and their degenerated case called de-correlated component analysis (DCA) [22.53]. Typical example of $(2,1)$ is factor analysis (FA) by Eq. (22.4), and that of $(3,1)$ is $\min E\|x - AWy\|^2$ that leads to a singular value decomposition (SVD) analysis [22.6]. In this special case, both the linear mapping $x \to y$ and $y \to x$ can be directly obtained. Interestingly, performing PCA is a common special case of three architecture. Similar to Fig. 22.1, they can be further extended to not only temporal PCA (see Eq. (87) in [22.57]; Eq. (163) in [22.53]), temporal FA (see Eqs. (78) and (79) in [22.57]), and temporal SVD by adding in time, as well as hidden layer aided temporal regression by focusing on two part relation, but also to local PCA, local MCA, and local FA [22.54], etc. by jointly considering multiple models but ignoring topology, as well as further to their self-organizing map versions by mapping them on a regular topology.

(b) $F_2 = \{(1,2),(2,2),(3,2)\}$ is featured by a non-Gaussian vector $y = [y_1, \cdots, y_k]$ with components independent from each other such that statistics higher than the second order are also in consideration. A typical example of $(1,2)$ is the ICA discussed at the end of Sect. 2.1. A typical example of $(2,2)$ is non-Gaussian factor analysis (NFA) (see Sect. III in [22.57]), which is superior to ICA via taking observation noise in consideration. An example of $(3,2)$ is an extension of LMSER learning (Sect. 5.4 in [22.53]). Again, it follows from Fig. 22.1 that they can be further extended to temporal versions by adding in time, to local ICA, local NFA, and local LMSER [22.54] by jointly considering multiple models but ignoring topology, as well as further to mapping them on a regular topology.

(c) $F_3 = \{(1,3),(2,3),(3,3)\}$ is featured by a binary vector $y = [y_1, \cdots, y_k]$ with bits independent from each other, which is suitable for problems of encoding data into binary bits. The ICA algorithm made by [22.4] can be regarded as a typical example of $(1,3)$. An example of $(2,3)$ is binary factor analysis (BFA), and that of $(3,3)$ is the LMSER learning [22.94]. Similar to Fig. 22.1, they can be extended to temporal ICA, independent HMM, and its Bi-directional version [22.53, 22.59, 22.57] by adding in time, to local ICA, local BFA, and local LMSER [22.54, 22.55] as well as further to their SOM extensions.

(d) $F_4 = \{(1,4),(2,4),(3,4)\}$ is featured by a discrete label $y = 1, \cdots, k$ that is suitable for making pattern classification via the mapping $x \to y$. The classic perceptron [22.11] and the recent SVM based perceptron [22.47] are both examples of $(1,4)$ for a two classes problem (i.e., $k = 2$). The maximum balanced mapping certainty principle proposed in [22.84] is another example of $(1,4)$ for $k > 2$. A typical example of $(2,4)$ is the finite mixture given by Eq. (22.14), and that of $(3,4)$ is given by BYY learning with a Bi-directional architecture [22.30].

(e) $F_5 = \{(1,5),(2,5),(3,5)\}$ is featured by a uniform distribution for y. It can be regarded as special cases of the above four families with the distribution of y fixed uniformly in a sense that every value of y is taken with an equal probability. As discussed at the end of Sect. 22.2, the mapping $x \to y$ completely describes data distribution in a CDF form [22.53]. Moreover, we may also fix y on a known distribution for specific tasks.

The above discussed perspective of jointly considering forward structure and backward structure, together with typical types of inner representation, provides a systematic view to model various dependence structures. What discussed above is exactly the fundamental sprit of Bayesian Ying Yang system that acts as a unified statistical learning framework. Actually, a quite number of dependence structures mentioned above were initially brought into studies under the guidance of this BYY system. In the rest of this chapter, advances on BYY system and harmony learning will be further overviewed.

22.5 Bayesian Ying Yang System

The world that we consider above can be denoted by $\mathbf{X} = \{X, L\}$ that consists of a number of objects to observe, with L denoting a set of labels with each $\ell \in L$ denoting one object. X is a set of samples with each sample x coming from one of these objects. We have several special cases that correspond to dependence structures previously discussed. The simplest case is that there is only one object in L and X consists of a size N of samples that all come from this object. Thus, L can be ignored. For the cases in Sect. 22.3.1, L has a regular lattice topology and each object ℓ locates at each node n_ℓ. Also, it is already known that each x comes from which node and, thus, can be labeled explicitly, i.e., x_ℓ. Particularly, the topology L for the case (d) can be decomposed into a direct product of a line L_t for time and a regular lattice L_r, i.e., $L = L_t \times L_r$. For the cases in Sect. 22.3.2, the label that indicates where x comes from is actually missing. Through learning, either each sample x is classified into one ℓ of a finite mixture or each sample x is mapped to a node n_ℓ on L such that nodes located nearby describes similar samples. For the case (d) with a topology $L = L_t \times L_r$, x comes at which time is known, and, thus, there is no need to allocate x to a time. Further noticing that the cases of $x = [\xi, \eta]$ are covered as special cases of the above discussed, all the cases in Fig. 22.1 can be summarized by the notation $\mathbf{X} = \{X, L\}$.

Types of dependencies in Fig. 22.2 are further considered via a corresponding representation domain $\mathbf{Y} = \{Y, L\}$ of a BYY system. For simplicity, we start at considering L that consists of a number of isolated objects without any topology. In this case, each $\mathbf{x} = \{x, \ell\}$ denotes a joint event of an observation $x = [x^{(1)}, \cdots, x^{(d)}]^T$ and an object ℓ, subject to a joint underlying distribution $p(\mathbf{x}) = p(x, \ell)$. Corresponding to each $\mathbf{x}$, there is an inner representation $\mathbf{y} = \{y, \ell\}$ in the representation domain $\mathbf{Y} = \{Y, L\}$, subject to a parametric structure of $q(\mathbf{y}) = q(y, \ell)$. Except of the family F_5, representation types of y are rather simple and correspond to the following distributions:

$$q(\mathbf{y}) = q(y, \ell) = q(y|\ell)q(\ell), \; q(\ell) = \sum_{j=1}^{k} \alpha_j \delta(\ell - j), \; \alpha_\ell \geq 0, \; \sum_{\ell=1}^{k} \alpha_\ell = 1,$$

$$q(y|\ell) = \textstyle\prod_{j=1}^{m_\ell} q(y^{(j)}|\ell), \; y = [y^{(1)}, \cdots, y^{(m_\ell)}]^T, \tag{22.16}$$

$$q(y^{(j)}|\ell) = \begin{cases} G(y^{(j)}|\mu_\ell^{(j)}, \lambda^{(j)\,2}), & \text{for } F_1, \\ \sum_i \beta_{ji} G(y^{(j)}|\mu_\ell^{(ji)}, & \\ \lambda^{(ji)\,2}), \sum_i \beta_{ji} = 1, 0 \leq \beta_{ji} \leq 1, & \text{for } F_2, \\ (q_\ell^{(j)})^{y^{(j)}} (1 - q_\ell^{(j)})^{1 - y^{(j)}}, & \text{for } F_3, \\ \delta(y^{(j)} - \mu_\ell^{(j)}), & \text{for } F_4. \end{cases}$$

where k is the number of labels in L, and m_ℓ is the dimension of either a binary or real vector y.

As shown in Fig. 22.3, we consider the joint distribution of $\mathbf{x}$ and $\mathbf{y}$, which can be understood from two complementary perspectives. On the one hand, we can interpret each $\mathbf{x}$ generated from an invisible inner representation $\mathbf{y}$ via a backward path distribution $q(\mathbf{x}|\mathbf{y})$, or a generative model

$$q(\mathbf{x}) = \textstyle\int q(\mathbf{x}|\mathbf{y}) q(\mathbf{y}) d\mathbf{y} \tag{22.17}$$

that maps from an inner distribution $q(\mathbf{y})$. On the other hand, we can interpret each $\mathbf{x}$ as being mapped into an invisible inner representation $\mathbf{y}$ via a forward path distribution $p(\mathbf{y}|\mathbf{x})$, or a representative model

$$p(\mathbf{y}) = \textstyle\int p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \tag{22.18}$$

that matches the inner density $q(\mathbf{y})$.

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 constraints, the two decompositions should be theoretically identical. Considering real situations, however, the four components $p(\mathbf{y}|\mathbf{x}), p(\mathbf{x}),\; q(\mathbf{x}|\mathbf{y})$, and $q(\mathbf{y})$ should be subject to certain structural constraints. Thus, we usually have two different but complementary Bayesian representations:

$$p(u) = p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}), \; q(u) = q(\mathbf{x}, \mathbf{y}) = q(\mathbf{x}|\mathbf{y})q(\mathbf{y}), \tag{22.19}$$

As discussed in the original paper [22.88], thanks to the famous Chinese ancient Ying-Yang philosophy, $p(\mathbf{x}, \mathbf{y})$ is called Yang machine, and consists of the observation space (or Yang space) of $p(\mathbf{x})$ and the forward pathway (or Yang pathway) of $p(\mathbf{y}|\mathbf{x})$, and $q(\mathbf{x}, \mathbf{y})$ is called the Ying machine, and consists of the invisible state space (or Ying space) of $q(\mathbf{y})$ and the Ying (or

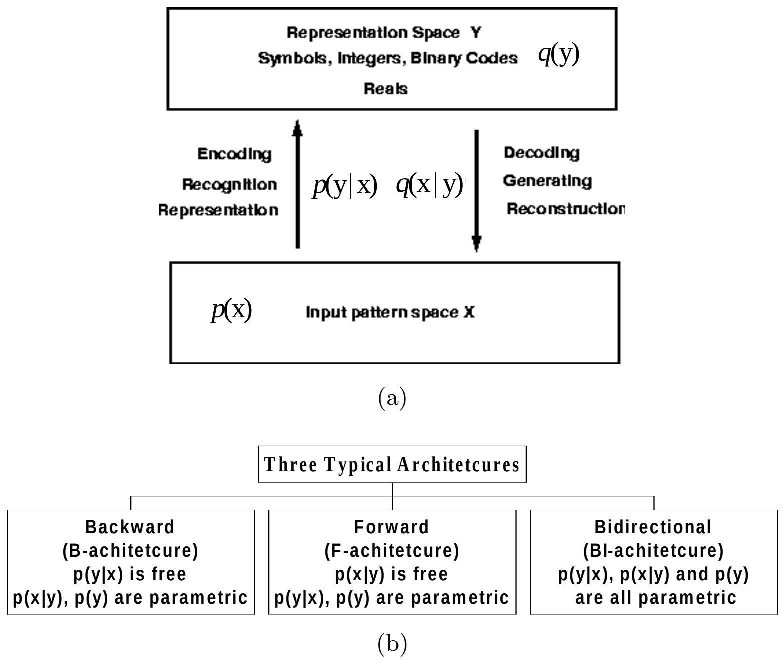


Fig. 22.3. Bayesian Ying Yang System and Three Architectures

backward) pathway of $q(\mathbf{x}|\mathbf{y})$. Such a pair of Ying-Yang models is called *the Bayesian Ying Yang (BYY) system.*

This BYY system provides a unified framework for describing various dependence structures in Sect. 22.2 and Sect. 22.3, with details as follows:

- The distribution $p(\mathbf{x})$ is obtained on a set $\mathcal{X}$ of samples from the observed world $\mathbf{X}$, either by the empirical density of Eq. (22.1) or by the nonparametric estimate of Eq. (22.3) with an unknown smoothing parameter h.
- The Ying path, at the special case of only one object in L, covers Eq. (22.9) that includes Eq. (22.5) and Eq. (22.8). Generally, the cases with $k \geq 2$ extend Eq. (22.9) to modular generative structures.
- The Yang path provides the general form of Eq. (22.18) for describing various modular linear and nonlinear transform structures, including those in Sect. 22.2 at a special case of only one object in L.
- All the dependence structures in Sect. 22.3.1 are all covered when L has the topology of a line, a lattice, and a tree, respectively, when the dependence relation between X and L pre-specified. Moreover, they can be further extended to their corresponding self-organizing maps when the dependence relation between X and L is missing.

The task of learning on a BYY system consists of specifying all the aspects of $p(\mathbf{y}|\mathbf{x})$, $q(\mathbf{x}|\mathbf{y})$, and $q(\mathbf{y})$, as well as h (if applicable).

First, it follows from Eq. (22.16) that $q(\mathbf{y})$ is specified by both $\mathbf{k}$ that consists of the scales k and $\{m_\ell\}$ of the representation domain $\mathbf{Y}$ and a set θ_y of all parameters in $q(\mathbf{y})$.

Second, we need to design the structures of $p(\mathbf{y}|\mathbf{x})$ and $q(\mathbf{x}|\mathbf{y})$. We say $p(u|v)$ is structurally free if $p(u|v) \in \mathcal{P}^0_{u|v}$, with $\mathcal{P}^0_{u|v}$ consisting of all functions in the form of $p(u|v)$ that satisfy $\int p(u|v)du = 1$ and $p(u|v) \geq 0$. One of $p(\mathbf{y}|\mathbf{x}), q(\mathbf{x}|\mathbf{y})$ is designed as being structure-free, which means that there is no priori constraint to impose on it, and it will be specified during learning by other components in the BYY system. In contrast, a parametric $p(u|v) \in \mathcal{P}^S_{u|v}$ means that it comes from a family $\mathcal{P}^S_{u|v}$ with a pre-specified structure based on certain priori requirements or knowledge, and a particular density is specified by a set $\theta_{u|v}$ of parameters. That is, the function form of $p(u|v)$ is pre-specified, while the value of $\theta_{u|v}$ remains unknown. As suggested in [22.66, 22.62, 22.60], when u is a binary vector, a typical example is

$$\begin{aligned} &p(u|v) = \prod_{j=1}^{m} \pi_j(v)^{u^{(j)}} (1-\pi_j(v))^{1-u^{(j)}}, \\ &\pi(v) = [\pi_1(v), \cdots, \pi_m(v)]^T = S(Wv + c), \\ &S(y) = [s(y^{(1)}), \cdots, s(y^{(m)})]^T, \ 0 \leq s(r) \leq 1 \text{ is a sigmoid function.} \end{aligned} \tag{22.20}$$

When u is a real vector, a typical example is

$$\begin{aligned} &p(u|v) = \sum_{j=1}^{n} \beta_j(v) p_j(u|v), \ \sum_{j=1}^{n} \beta_j(v) = 1, \ \beta_j(v) \geq 0, \\ &p_j(u|v) = G(u|f_j(v|\theta_{u|v,j}), \Sigma_{u|v,j}). \end{aligned} \tag{22.21}$$

Correspondingly, the function of a BYY system is featured by the representation types of y and the implementing architecture of a BYY system is featured by a combination of the specific structures of $p(\mathbf{y}|\mathbf{x})$ and $q(\mathbf{x}|\mathbf{y})$. As shown in Fig. 22.3(b), we have

– A B-architecture when focusing on only dependence structures of a generative type, as discussed in Sect. 22.2, with a structure-free $p(\mathbf{y}|\mathbf{x})$,
– An F-architecture when focusing on only the linear and nonlinear transform structures in Sect. 22.2, with a structure-free $q(\mathbf{x}|\mathbf{y})$,
– A BI-architecture when both types of dependence structures are explored together.

The architecture in which both $p(\mathbf{y}|\mathbf{x})$ and $q(\mathbf{x}|\mathbf{y})$ are structure-free is useless and, thus, ignored.

In a summary, our learning task includes two subtasks. One is parameter learning for determining the value of θ that consists of all the unknown parameters in $p(\mathbf{y}|\mathbf{x})$, $q(\mathbf{x}|\mathbf{y})$, and $q(\mathbf{y})$, as well as h (if applicable). The other is selecting the representation scales $\mathbf{k} = \{k, \{m_\ell\}\}$, called *model selection*, since a collection of specific BYY systems with different scales corresponds to a family of specific models that share the same system configuration but in different scales.

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

– The difference between the two Bayesian representations in Eq. (22.19) should be minimized.
– The resulting BYY system should be of the least complexity.

22.6 BYY Harmony Learning

22.6.1 Kullback Divergence, Harmony Measure, and *Z*-Regularization

To implement the above harmony learning principle, we need to formalize it mathematically. One possible measure is the well known Kullback divergence

$$KL(p\|q) = \int p(u) \ln \frac{p(u)}{q(u)} du \geq 0, \ KL(p\|q) = 0, iff \ p(u) = q(u) \tag{22.22}$$

which is applicable to the cases where both p and q are discrete and continuous densities. The minimization of the Kullback divergence implements the first point well, and this is why it has been used in early stages of the BYY system [22.89, 22.82, 22.83, 22.85, 22.74, 22.75, 22.76, 22.77]. However, it is not able to implement the least complexity nature. In other words, the Kullback divergence can only be used for partial implementation. We need a measure that implements both the points.

We consider the following cross entropy

$$H(p\|q) = \textstyle\sum_{t=1}^{N} p_t \ln q_t, \tag{22.23}$$

where both $p(u)$ and $q(u)$ are discrete densities of the form

$$q(u) = \textstyle\sum_{t=1}^{N} q_t \delta(u - u_t), \ \sum_{t=1}^{N} q_t = 1. \tag{22.24}$$

The maximization of $H(p\|q)$ has two interesting natures:

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

$$q_t = p_t, \text{ for all } t. \tag{22.25}$$

• *Least complexity nature* with q fixed, $\max_p H(p\|q)$ pushes p toward its simplest form

$$p(u) = \delta(u - u_\tau), \text{or } p_t = \bar{\delta}_{t,\tau}, \text{ with } \tau = arg \max_t \ q_t, \tag{22.26}$$

where, and throughout this paper, $\bar{\delta}_{j,j^*} = 1$ when $j = j^*$, $\bar{\delta}_{j,j^*} = 0$ otherwise.

As discussed in [22.57, 22.58, 22.54, 22.55], Eq. (22.26) is a kind of the least complexity form from a statistical perspective. In other words, the maximization of the functional $H(p\|q)$ indeed implements the above harmony learning principle mathematically.

Moreover, as shown in [22.57, 22.58, 22.54, 22.55], either a discrete or a continuous density $q(u)$ can be represented in the form of Eq. (22.24) via the normalization

$$q_t = q(u_t)/z_q, \; z_q = \textstyle\sum_{t=1}^{N} q(u_t), \tag{22.27}$$

based on a given set $\{u_t\}_{t=1}^{N}$ of samples.

Putting Eq. (22.27) into Eq. (22.23), it follows that we can further get a general form of the harmony measure [22.57]

$$H(p\|q) = \textstyle\int p(u) \ln q(u) du - \ln z_q, \tag{22.28}$$

which degenerates to Eq. (22.23) when $p(u)$ and $q(u)$ are discrete, as in Eq. (22.24).

To get a better insight, we further examine the maximization of $H(p\|q)$ from the aspects of its two natures discussed above.

Still, $\max_p H(p\|q)$, with q fixed, leads to the least complexity nature of the form in Eq. (22.26). It is not directly observable that this nature has any use, since $\delta(u-u_\tau)$ has a very limited representation ability. This may be the reason why the least complexity nature of Eq. (22.26) of the cross entropy in Eq. (22.23) has been rarely studied in the literature. However, as will be introduced in the next subsection, this nature makes a big difference on a BYY system because it enables model selection.

In contrast, the matching nature by $\max_q H(p\|q)$, with p fixed, behaves similarly in both the direct case of Eq. (22.28) and its use on a BYY system. The details are discussed as follows:

– When $p(u)$ is given by its empirical density in the form of Eq. (22.1), considering a crude approximation $z_q = 1$ will make $H(p\|q)$ become the likelihood

$$L(\theta) = \textstyle\sum_{t=1}^{N} \ln q(u_t). \tag{22.29}$$

That is, it becomes equivalent to the maximum likelihood (ML) learning.

– Considering Eq. (22.28) with the normalization term z_q in Eq. (22.27), it follows from $p(u)$, given by Eq. (22.1), and $q(u)$, given by Eq. (22.27), with u in the place of x, that

$$H(p\|q) = L(\theta) - \ln z_q, \; z_q = \textstyle\sum_{t=1}^{N} q(u_t). \tag{22.30}$$

By comparing the gradients

$$\begin{aligned} &\nabla_\theta L(\theta) = Gd(\gamma_t)|_{\gamma_t=\frac{1}{N}}, \; \nabla_\theta H(p\|q) = Gd(\gamma_t)|_{\gamma_t=\frac{1}{N}-\tilde{q}(u_t|\theta)}, \\ &Gd(\gamma_t) = \textstyle\sum_t \gamma_t \nabla_\theta \ln q(u_t|\theta), \; \tilde{q}(u_t|\theta) = q(u_t|\theta)/\textstyle\sum_\tau q(u_\tau|\theta), \end{aligned} \tag{22.31}$$

we see that the log-normalization term $\ln z_q$ causes a *conscience* de-learning on the ML learning, which is, thus, referred as *normalization learning*, in a sense that the degree of de-learning on learning u_t is proportional to the likelihood that $q(u|\theta)$ fits u_t. That is, the better it is fitted, the more conscience it makes during learning, which actually provides a regularization that prevents $q(u|\theta)$ from overfitting a data set of a finite size.

– Considering $p(u)$ given by the Parzen window estimate of Eq. (22.3), with u in the place of x, we can also approximate z_q under a weak constraint $\sum_{t=1}^{N} p(u_t) \approx \sum_{t=1}^{N} q(u_t)$, which leads to a regularized learning as shown by Eqs. (28) and (33) in [22.57]. That is, Eq. (22.28) becomes

$$\begin{aligned} & H(p\|q) = \tilde{L}_S(\theta_h) + 0.5d \ln(2\pi h^2) + \ln N - J_H(h,k), \\ & J_H(h,k) = \ln[\sum_{\tau=1}^{N}\sum_{t=1}^{N} e^{-0.5\frac{\|u_t - u_\tau\|^2}{h^2}}], \\ & \tilde{L}_S(\theta_h) = \int p_h(u) \ln q(u) du \approx L(\theta) + 0.5h^2\pi_q, \\ & \pi_q = \frac{1}{N}\sum_t Tr[\frac{\partial^2 \ln q(u|\theta)}{\partial u \partial u^T}]_{u=u_t}. \end{aligned} \qquad (22.32)$$

$\tilde{L}_S(\theta_h)$ regularizes the ML learning by smoothing each likelihood $\ln q(u_t)$ in the near-neighborhood of u_t, something referred to as *data smoothing.* It can also be observed that the role of h^2 is equivalent to that of the hyper-parameter in Tikhonov-type regularization [22.5]. What is new here is that the other terms in $H(p\|q)$ balance $\tilde{L}_S(\theta_h)$ such that an appropriate h can be learned together with θ by Eq. (34) [22.54].

The fact that $\max_\theta \int p_0(u) \ln q(u|\theta) du$ leads to the ML learning of Eq. (22.29) is well known in the literature. Moreover, $\max_\theta \int p^*(u) \ln q(u|\theta) du$, with $p^*(u)$ being the true distribution of samples, has also been studied in developing the AIC criterion [22.1]. What is new here is that the term z_q introduces a regularization to the ML learning, which is shortly called the z-regularization. This regularization is implemented by two techniques, i.e., either a conscience de-learning type or a Tikhonov-type term, with the hyper-parameter h learned in an easily implemented way.

A further insight can be obtained by returning to the Kullback divergence Eq. (22.22). When both $p(u)$ and $q(u)$ are discrete densities in the form of Eq. (22.24), from Eq. (22.22) we can directly get

$$KL(p\|q) = \sum_{t=1}^{N} p_t \ln \frac{p_t}{q_t} = -E_p - H(p\|q), E_p = -\sum_{t=1}^{N} p_t \ln p_t. \qquad (22.33)$$

Helped by the form of Eq. (22.27) for both p and q, similar to Eq. (22.28) we can get

$$\begin{aligned} & KL(p\|q) = \sum_{t=1}^{N} \frac{p(u_t)}{z_p} \ln \frac{p(u_t)/z_p}{q(u_t)/z_q} \approx \int p(u) \ln \frac{p(u)/z_p}{q(u)/z_q} du, \\ \text{or } \ & KL(p\|q) = -H(p\|q) - E_p, E_p = -\int p(u) \ln p(u) du + \ln z_p. \end{aligned} \qquad (22.34)$$

Obviously, $KL(p\|q)$ becomes as in Eq. (22.22) when we have $z_q = z_p$, e.g., both $p(u)$ and $q(u)$ are discrete densities or approximately have $z_q = z_p$, e.g., when both $p(u), q(u)$ are continuous densities with $p(u)$ given by the Parzen window estimate of Eq. (22.3) and $q(u)$ given by a continuous parametric model. That is, Eq. (22.22) is directly applicable to the cases where both

$p(u)$ and $q(u)$ are discrete densities and that both $p(u), q(u)$ are continuous densities. However, Eq. (22.22) is not directly applicable when $p(u)$ is discrete and $q(u)$ is continuous, with z_p being infinite and z_q remaining bounded. In this case, $p(u)$ should be replaced by its normalized version $\hat{p}(u) = p(u)/z_p$.

Also, it follows from Eq. (22.33) and Eq. (22.34) that $\min KL(p\|q)$ is different from $\max H(p\|q)$ in that it also maximizes the entropy E_p of $p(u)$, which prevents $p(u)$ from approaching the form of Eq. (22.26). This explains why Eq. (22.33) does not have the least complexity nature. Particularly, when $E_p = c$ is a constant that does not relate to learning, $\min KL(p\|q)$ become equivalent $\max_{s.t.\ E_p=c} H(p\|q)$. Moreover, when $p(u)$ is given by a Parzen window, we get a regularized learning via data smoothing in a way similar to Eq. (22.32) but with the team $J_H(h,k)$ replaced by a different term $J_{KL}(h,k)$ that takes a similar role [22.54]. Furthermore, when $p(u) = p_0(u)$ given by Eq. (22.1), we have $E_p = 0$ and, thus, $\min KL(p\|q)$ and $\max H(p\|q)$ become equivalent completely. A detailed relation of the two types of learning is referred to the next chapter.

22.6.2 BYY Harmony Learning

By putting Eq. (22.19) into Eq. (22.28), we have

$$H(p\|q) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \ln [q(\mathbf{x}|\mathbf{y})q(\mathbf{y})]d\mathbf{x}d\mathbf{y} - \ln z_q \tag{22.35}$$

for the harmony learning on a BYY system. Here, we get a salient feature that is not shared by Eq. (22.28). Now, only $p(\mathbf{x})$ is fixed at a non-parametric estimate, and $p(\mathbf{y}|\mathbf{x})$ is either free in a B-architecture or a parametric form in a BI-architecture, and will be pushed into its least complexity form due to the nature Eq. (22.26). For example, in a B-architecture $p(\mathbf{y}|\mathbf{x})$ will be determined by $\max_{p(\mathbf{y}|\mathbf{x})} H(p\|q)$, resulting in the following least complexity form:

$$p(\mathbf{y}|\mathbf{x}) = \delta(\mathbf{y} - \mathbf{y}(\mathbf{x})), \quad \mathbf{y}(\mathbf{x}) = arg \max_{\mathbf{y}} [q(\mathbf{x}|\mathbf{y})q(\mathbf{y})]. \tag{22.36}$$

On the other hand, the matching nature of harmony learning will further push $q(\mathbf{x}|\mathbf{y})$ and $q(\mathbf{y})$ toward their corresponding least complexity forms. In other words, the least complexity nature and the matching nature collaborate to make model selection possible such that $\mathbf{k}$ is appropriately determined.

Mathematically, the harmony learning is implemented by

$$\max_{\theta,\mathbf{k}} \ H(\theta,\mathbf{k}), \ H(\theta,\mathbf{k}) = H(p\|q), \tag{22.37}$$

which is a combined task of continuous optimization for parameter learning and discrete optimization for model selection, both under the same cost function $H(\theta,\mathbf{k})$. This feature makes it possible to simultaneously implement parameter learning and model selection together. Actually, the least complexity nature of Eq. (22.26) makes it possible for us to implement parameter learning with automatic model selection.

To get a further insight, it follows from Eq. (22.16) that $\alpha_\ell = 0$ for some ℓ implies that k is reduced by one. Also, we observe that a form

$$q(y|\ell) = \delta(y^{(j)} - \mu_0) q(y^-|\ell), \; y = [y^-, y^{(j)}]^T, \; \mu_0 \text{ is a constant}, \quad (22.38)$$

implies that the dimension of $y^{(j)}$ is not actually in action, and, thus, the dimension m_ℓ is effectively reduced by one. Therefore, a value of θ with these two types of settings is equivalent to forcing k and $\{m_\ell\}$ to be effectively reduced to appropriate scales.

With Eq. (22.36) put into Eq. (22.35), we can observe that $\max H(p\|q)$ is equivalent to maximizing $\ln q(x_t|y_t, \ell)$, $\ln q(y_t|\ell)$, and $\ln \alpha_\ell$ for each sample. Specifically, in maximizing $\ln \alpha_\ell$,

each extra α_ℓ is pushed toward zero, (22.39)

and in maximizing $\ln q(y|\ell)$,

$q(y^{(j)}|\ell)$ on each extra dimension is pushed toward $\delta(y^{(j)} - \mu_0)$.(22.40)

Therefore, fixing the scales of $\mathbf{k}$ large enough, we can implement the harmony learning by

$$\max_\theta \; H(\theta), \quad H(\theta) = H(\theta, \mathbf{k}), \quad (22.41)$$

which will let θ take a specific value such that $\mathbf{k} = \{k, m\}$ is effectively reduced to an appropriate scale. In other words, the least complexity nature of Eq. (22.26) automatically implies model selection during learning. As demonstrated in Fig. 22.4(b), the learning by Eq. (22.41) will push a set of extra parameters $\theta_2 = \theta_2^*$ such that a large k becomes effectively k^*, that is, a smallest value of k at which the maximum of $H(\theta)$ is reached.

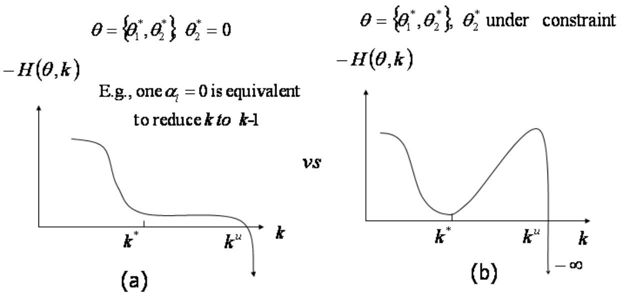


Fig. 22.4. (a) Model selection made after parameter learning on every k in a given interval $[k_d, \; k_u]$, (b) Automatic model selection with parameter learning on a value k of large enough.

The above feature is not shared by existing approaches in the literature. By the conventional approaches, parameter learning and model selection are made in a two-phase style. First, parameter learning is made usually under the maximum likelihood principle. Then, model selection is made by a different criterion, e.g., AIC, MDL, etc. These model selection criteria are usually not good for parameter learning, while the maximum likelihood criterion is not good for model selection, especially on a small size of training samples.

If one wants, the problem Eq. (22.37) can also be implemented in such a two-phase style. In the first phase, we enumerate $\mathbf{k} = \{k, \{m_\ell\}\}$ from small values incrementally. At each k and $\{m_\ell\}$, we perform parameter learning for seeking a best value of θ^*. In the second phase, we select a best $k^*, \{m^*_\ell\}$ by

$$\min_{1 \le k \le k^u,\ 1 \le m_\ell \le m^u_\ell,\ \forall \ell \in L} J(k, \{m_\ell\}), \quad J(k, \{m_\ell\}) = -H(\theta^*, \mathbf{k}), \qquad (22.42)$$

where $k^u, \{m^u_\ell\}$ is a set of upper bounds. To get an insight, we consider a Gaussian mixture by Eq. (22.14) with $q(x|\ell) = G(x|\mu_\ell, \Sigma_\ell)$. In this case, we have [22.77]:

$$J(k) = \frac{0.5}{k} \textstyle\sum_{\ell=1}^{k} \ln |\Sigma_\ell| + \ln k. \qquad (22.43)$$

As k increases the number of samples that are allocated to each Gaussian decreases. Thus, each $|\Sigma_\ell|$ decreases with k and the first term of $J(k)$ decreases with k. However, $\ln k$ increases with k. The two terms trade off such that $J(k)$ first decreases with k and reaches a minimum, and then increases due to $\ln k$. As k further increases, there will be few samples available to be allocated to a Gaussian such that Σ_ℓ becomes singular which brings $J(k)$ drops rapidly toward $-\infty$. That is, $J(k)$ generally has an inverse N shape as shown in Fig. 22.4(a). We use k^u to denote the smallest value of k that makes $J(k)$ rapidly tend to ∞. The best value k^* is selected by Eq. (22.42) within $1 \le k \le k^u$.

As above discussed, parameter learning by Eq. (22.41) usually (e.g., for F_1, F_2, F_3 in Fig. 22.2) leads to an automatic model selection and, thus, there is no need to implement the selection by Eq. (22.42). However, for certain learning tasks (e.g., for F_4, F_5 in Fig. 22.2), the inner representation is prespecified to be uniform across both different objects and different dimensions [22.54]. That is,

$$\alpha_\ell = \frac{1}{k}, \textstyle\int (y - \mu_{y,\ell})(y - \mu_{y,\ell})^T q(y|\ell) dy = b_0 I, \mu_{y,\ell} = \textstyle\int y q(y|\ell) dy (22.44)$$

For example, $b_0 = 1$ for a real $y^{(j)}$ and $b_0 = 0.25$ for a binary $y^{(j)}$. Due to the constraint, automatic model selection will not happen during learning by Eq. (22.41) in the first phase, we need to implement Eq. (22.42) in the second phase.

Alternatively, we can also replace Eq. (22.41) by minimizing the Kullback divergence Eq. (22.22), i.e.,

$$\min_\theta \ KL(\theta) = \textstyle\int p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}) \ln \frac{p(\mathbf{y}|\mathbf{x}) p(\mathbf{x})}{q(\mathbf{x}|\mathbf{y}) q(\mathbf{y})} d\mathbf{x} d\mathbf{y}, \qquad (22.45)$$

without the constraint by Eq. (22.44). Particularly, on a B-architecture, the minimization of the above $KL(\theta)$ with respect to a free $p(\mathbf{y}|\mathbf{x})$ will result in

$$p(\mathbf{y}|\mathbf{x}) = \frac{q(\mathbf{x}|\mathbf{y})q(\mathbf{y})}{q(\mathbf{x})},\ q(\mathbf{x}) = \int q(\mathbf{x}|\mathbf{y})q(y)d\mathbf{y},$$
$$KL(\theta) = \int p(\mathbf{x}) \ln \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}, \tag{22.46}$$

which becomes equivalent to ML learning on $q(\mathbf{x})$ when $p(\mathbf{x}) = p_0(x)$ is given by Eq. (22.2) [22.88]. In this case, we actually implement the ML learning in the first phase and then model selection by Eq. (22.42) in the second phase.

No longer holding the least complexity nature by Eq. (22.26), the implementation of Eq. (22.45) will not lead to a case of Fig. 22.4(b), and, thus, there is no need to impose the assumption that $q(y)$ comes from a family with equal variances among components.

22.6.3 A Further Extension: From ln(r) to Convex Function

Extensions of the BYY harmony learning have been made either into temporal BYY system by taking temporal relation in consideration or into structural BYY system by considering the representation space $\mathbf{y}$ with certain structure [22.90, 22.74, 22.65, 22.61, 22.64, 22.59, 22.57, 22.53, 22.50]. Extension has also been made with the log function $\ln(r)$ replaced by the following convex function family [22.75, 22.77, 22.80, 22.81]

$$F_c = \{f(r) : \frac{d^2 f(r)}{d^2} < 0,\ f(r) \text{ monotonically increases with } r\} \tag{22.47}$$

It is interesting to investigate what will happen when $\ln(r)$ is replaced with any $f(r) \in F_c$.

Let us return back to consider such an extension from Eq. (22.23). That is,

$$H_f(p\|q) = \textstyle\sum_{t=1}^{N} p_t f(q_t), \tag{22.48}$$

It can be observed that the maximization of this $H_f(p\|q)$ with respect to p_t still have the same least complexity nature as Eq. (22.26) while the match nature by Eq. (22.25) is modified into

$$q_t = f'(\frac{1}{p_t}) / \textstyle\sum_{t=1}^{N} f'(\frac{1}{p_t}),\ f'(r) = df(r)/dr, \tag{22.49}$$

which returns back to Eq. (22.25) when $f'(r) = 1/r$. We can further classify other $f(r) \in F_c$ according to whether its $f'(r)$ decreases with r in a rate slower than $1/r$. If yes, it is said to be super-ln; otherwise it is said to be sub-ln.

A typical family of super-ln is the so-called α-function [22.69, 22.77, 22.80, 22.81] as follows:

$$f(r) = r^{\alpha}, 0 \le \alpha \le 1,\ \textit{and, thus, }\ f'(r) = \alpha / r^{1-\alpha}. \tag{22.50}$$

In this case, Eq. (22.49) can be rewritten as

$$q_t = p_t^{1-\alpha} / \textstyle\sum_{t=1}^{N} p_t^{1-\alpha}, \tag{22.51}$$

which returns to Eq. (22.25) with $q_t = p_t$ when $\alpha = 0$ and becomes $q_t = 1/N$ when $\alpha = 1$. In other cases, p_t attempts to take q_t with lower probabilities increased but higher probabilities decreased in a certain degree that is controlled by α. That is, p_t becomes more spreading than q_t as α increases from 0 to 1. It becomes uniform when $\alpha = 1$. In other words, a super-ln function leads to a conscience de-learning regularization.

An example family of sub-ln is the following inverted and negated α-function as follows:

$$f(r) = -r^{\alpha}, \alpha < 0, \ and, \ thus, \ f'(r) = |\alpha| / r^{1+|\alpha|}. \tag{22.52}$$

which leads to

$$q_t = p_t^{1+|\alpha|} / \textstyle\sum_{t=1}^{N} p_t^{1+|\alpha|}. \tag{22.53}$$

Now, p_t adopts q_t with higher probabilities increased but lower probabilities further decreased in a certain degree. That is, p_t becomes more concentrated than q_t as $|\alpha|$ increases. In other words, a sub-ln function leads to a competition effectively similar to the least complexity nature.

The above discussions also apply to the continuous case by Eq. (22.28), which becomes:

$$H(p\|q) = \textstyle\int p(u) f(q(u)/z_q) du. \tag{22.54}$$

Now, not only the term z_q provides a regularization, which is referred to the next chapter in this book for a detailed discussion, but also z_q and a super-ln $f(r)$ jointly introduce the above discussed regularization.

Even without z_q (e.g., let $z_q = 1$), a super-ln $f(r)$ will also perform a regularization role. With $p(u)$ given by Eq. (22.1), Eq. (22.54) becomes a f-likelihood function $L_f = \frac{1}{N}\sum_{t=1}^{N} f(q(u_t))$ which differs from the log-likelihood $L = \frac{1}{N}\sum_{t=1}^{N} \ln q(u_t)$ in that $f(q(u_t))$ takes the position of $\ln q(u_t)$. Those unlikely samples with small $q(u_t)$ (e.g., outliers) locate within a drastic varying range of $\ln q(u_t)$ and, thus, contribute a big portion to affect L. That is, the ML learning is vulnerable to the disturbance by outliers. In contrast, for a super-ln $f(r)$ such as given by Eq. (22.50), the varying range of $f(q(u_t))$ with a small $q(u_t)$ is much smaller than that of $\ln q(u_t)$. Thus, L_f will be much less affected by those outliers with a small $q(u_t)$. In other words, we get a type of robust ML learning [22.77, 22.80].

The Kullback divergence can also be extended. There are two ways. One is simply replace $\ln r$ with $f(r)$, resulting in

$$KL_f(p\|q) = \textstyle\int p(u) f(\frac{p(u)/z_p}{q(u)/z_q}) du. \tag{22.55}$$

The other is considering its equivalent form of maximizing $H_{KL}(p\|q) = \int p(u) \ln \frac{q(u)/z_q}{p(u)/z_p} du$ and then replace $\ln r$ with $f(r)$, which results in the maximization of

$$H_{KL_f}(p\|q) = \int p(u)f(\frac{q(u)/z_q}{p(u)/z_p})du. \tag{22.56}$$

Being different from Eq. (22.55), it will return to exactly Eq. (22.54) when $p(u)$ is given by empirical density Eq. (22.1).

In the special case that $z_p = z_q$, we have

$$KL_f(p\|q) = \int p(u)f(\frac{p(u)}{q(u)})du, \; H_{KL_f}(p\|q) = \int p(u)f(\frac{q(u)}{p(u)})du. \tag{22.57}$$

Detailed studies can be further made by considering certain specific features of $f(r)$. For example, for those $f(r) \in F_c$ that satisfy

$$(a) \; f(ab) = f(a)f(b), \quad (b) \; f(a^{-1}) = f^{-1}(a), \tag{22.58}$$

e.g., for the case of Eq. (22.50) we can rewrite Eq. (22.54) into

$$H(p\|q) = f^{-1}(z_q)\int p(u)f(q(u))du, \tag{22.59}$$

and Eq. (22.56) into

$$H_{KL_f}(p\|q) = \frac{f(z_p)}{f(z_q)}\int p(u)\frac{f(q(u))}{f(p(u))}du. \tag{22.60}$$

All the above results can be applied to a BYY system, what needs to do is simply put Eq. (22.19) into Eq. (22.54), Eq. (22.55), Eq. (22.56), Eq. (22.59), and Eq. (22.60). First, maximizing $H(p\|q)$ by Eq. (22.54) with respect to a free $p(\mathbf{y}|\mathbf{x})$ will still lead to Eq. (22.36). Also, both minimizing $KL_f(p\|q)$ by Eq. (22.55) and maximizing $H_{KL_f}(p\|q)$ by Eq. (22.56) with respect to a free $p(\mathbf{y}|\mathbf{x})$ will lead to Eq. (22.46) too. Second, we are also lead to what will be discussed in the next chapter on trading off the strength of regularization and the ability of model selection in help of designing a parametric model for $p(\mathbf{y}|\mathbf{x})$. Third, a new perspective we get here is that such a trading off may also be achieved via choosing $f(r)$ to be super-ln or sub-ln, in the matching process of a Ying machine $q(\mathbf{x}|\mathbf{y})q(\mathbf{y})$ to a Yang machine $p(\mathbf{x}|\mathbf{y})p(\mathbf{y})$ with $p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x}$ and $p(\mathbf{x}|\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})/p(\mathbf{y})$.

Learning can still be implemented by the Ying-Yang alternative procedure in Sect. 22.7. For the Ying step, we no longer have the separated integral format as in Eq. (22.63). However, we can still get the gradient format similar to those in Eq. (22.65). The only difference is that $\eta_t(y)$ is replaced by $\eta_t^f(y)\eta_t(y)$ with

$$\eta_t^f(y) = \begin{cases} f'(\frac{q(x_t|y)q(y)}{z_q})\frac{q(x_t|y)q(y)}{z_q}, & \text{Harmony learning by Eq. (22.54)}, \\ f'(\frac{p(u)/z_p}{q(u)/z_q})\frac{p(u)/z_p}{q(u)/z_q}, & KL_f \text{ learning by Eq. (22.55)}, \\ f'(\frac{q(u)/z_q}{p(u)/z_p})\frac{q(u)/z_q}{p(u)/z_p}, & KL_f \text{ learning by Eq. (22.56)}, \end{cases} \tag{22.61}$$

where $f'(r) = df(r)/dr$.

For the Yang step, the discussions on a B-architecture in Sect. 22.7 remains the same. While for a BI-architecture, $\eta_t(y)$ in Eq. (22.71) is also replaced by $\eta_t^f(y)\eta_t(y)$ with $\eta_t^f(y)$ given by Eq. (22.61).

22.7 Ying-Yang Alternative Procedure for Parameter Learning

As discussed in [22.88], learning on the Yang machine $p(\mathbf{x},\mathbf{y})$ and the Ying machine $q(\mathbf{x},\mathbf{y})$ can be implemented alternatively by

$$\begin{array}{l}\text{Ying step: fixing } p(\mathbf{x},\mathbf{y}), \text{update unknowns in } q(\mathbf{x},\mathbf{y}),\\ \text{Yang step: fixing } q(\mathbf{x},\mathbf{y}), \text{update unknowns in } p(\mathbf{x},\mathbf{y}),\end{array} \qquad (22.62)$$

such that $-H(\theta)$ from Eq. (22.41), or $KL(\theta)$ from Eq. (22.45), gradually decreases until it converges.

This Ying-Yang alternative procedure also applies to the extension in the previous section, i.e., every appearance of $\ln r$ is replaced by a convex function $f(r) \in F_c$ by Eq. (22.47). For clarity, we focus on the function $\ln r$ in this section. However, all the discussions here directly applying to the cases with $\ln r$ replaced by a convex function $f(r)$.

(1) Ying step With $p(\mathbf{y}|\mathbf{x})$ fixed and the regularization term z_q ignored, both Eq. (22.41) and Eq. (22.45) share the same updating format, as follows:

$$\begin{array}{l}\max_{\theta_{x|y}} \ L(\theta_{x|y}), \ L(\theta_{x|y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \ln q(\mathbf{x}|\mathbf{y}) d\mathbf{x} d\mathbf{y},\\ \max_{\theta_y} \ L(\theta_y), \ L(\theta_y) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \ln q(\mathbf{y}) d\mathbf{x} d\mathbf{y},\end{array} \qquad (22.63)$$

where $\theta_{x|y}$ and θ_y consist of unknown parameters in $q(\mathbf{x}|\mathbf{y})$ and $q(\mathbf{y})$, respectively. We make the updates $\delta\theta_{x|y}$ and $\delta\theta_y$ such that $L(\theta_y + \delta\theta_y)$ and $L(\theta_{x|y} + \delta\theta_{x|y})$ either reach a local maximum or increase to certain extent. Typically, an updating $\delta\theta$ that increases an index $J(\theta)$ is a step size along a gradient-based direction $g_\theta = \nabla_\theta J(\theta)$, i.e.,

$$\delta\theta = \eta T g_\theta, \qquad (22.64)$$

where $\eta > 0$ is a small positive number that defines a step size. This $\delta\theta$ is along either the gradient direction when $T = I$ is a unit matrix or a direction that has a positive projection on the gradient direction when T is a positive definite matrix. Particularly, $\delta\theta$ is the well known natural gradient direction when T is the inverse of the metric tensor of $J(\theta)$.

When $p(\mathbf{x})$ is given by Eq. (22.1), the integral over $\mathbf{x}$ in both Eq. (22.35) and Eq. (22.45) will disappear. When $p(\mathbf{x})$ is given by Eq. (22.3), this integral over $\mathbf{x}$ can also be removed with the help of certain approximations [22.58, 22.54, 22.51, 22.52].

The integrals over y is either analytically solved when $q(x|y)$ and $q(y)$ are both Gaussian or becomes a computable summation when y takes one of discrete values $1, \cdots, k$ or is a binary vector $y = [y^{(1)}, \cdots, y^{(m)}]$. Otherwise, these integrals are difficult to compute. Still, even when it becomes a computable summation for a binary vector $y = [y^{(1)}, \cdots, y^{(m)}]$, the computing cost will increase exponentially with m.

The problem is tackled via letting the integral over the entire domain of y to be approximated by a summation on a set Y_t of a finite number of samples of y, which are obtained according to $p(y|x)$ in the Yang step. One typical case is that Y_t consists of only one sample $y_t = y(x_t)$. In this case, we can further make updating $\delta\theta_{x|y}, \delta\theta_y$ in the form of Eq. (22.64) with $\ln q(\mathbf{x}_t|\mathbf{y}_t)$ and $\ln q(\mathbf{y}_t)$ in the place of $J(\theta)$, respectively.

Without considering regularization, we have that both the harmony learning by Eq. (22.37) and the KL learning by Eq. (22.45) share a same format in the following gradients:

$$g_{\theta_{x|y}} = \frac{1}{N}\sum_{t=1}^{N}\int \eta_t(y)\nabla_{\theta_{x|y}} \ln q(x_t|y)dy, \tag{22.65}$$
$$g_{\theta_y} = \frac{1}{N}\sum_{t=1}^{N}\int \eta_t(y)\nabla_{\theta_y} \ln q(y)dy,$$

$$\eta_t(y) = \begin{cases} p(y|x_t) \text{ by Eq. (22.46)}, & \text{the KL learning by Eq. (22.45)}, \\ \delta(y - y(x_t)), & \text{the harmony learning by} \\ & \text{Eq. (22.37)}, \end{cases}$$

which can be further put in Eq. (22.64) for updating.

(2) Yang step This could be implemented in one of four typical ways, according not only to whether Eq. (22.41) or Eq. (22.45) is used for learning, but also whether a B-architecture or a BI-architecture is in consideration. The details are given as follows:

– In implementing the KL learning on a B-architecture with Eq. (22.45) for parameter learning or, equivalently, the ML learning on $q(\mathbf{x})$, the Yang step is to get Eq. (22.46). This is exactly the E-step by the well known EM algorithm [22.9, 22.38], with Eq. (22.63) being exactly the M-step. In other words, the EM algorithm is a specific case of the Ying-Yang alternative procedure given by Eq. (22.62). An integral over y has to be encountered for getting $q(x)$, which is either analytically solvable when $p(y|x)$, $p(x|y)$, and $p(y)$ are all Gaussian densities or becomes a computable summation when y takes one of discrete values $1, \cdots, k$ or is a binary vector $y = [y^{(1)}, \cdots, y^{(m)}]$. Moreover, even when it is computable for a binary vector $y = [y^{(1)}, \cdots, y^{(m)}]$, the computing cost will increase exponentially with m. In other cases, the integral is difficult to compute. It was as previously suggested in 1997 to be implemented approximately via a computationally expensive Monte Carlo simulation. That is, Y_t is obtained via randomly picking a set of samples of y according to $p(y|x_t)$, (see the choice (1) of Table 2(C) in [22.76] and the choice (a) in Eq. (24) and Sect. 3.1 in [22.66]) or even in a rough approximation according to $q(y)$ (see Step 1 in TableII(B) in [22.76]).
– In implementing the KL learning on a BI-architecture with Eq. (22.45) for parameter learning, the Yang step is to update $\theta_{y|x}$, which consists of all the parameters in $p(\mathbf{y}|\mathbf{x})$, by either Eq. (22.21) or Eq. (22.20). The update $\delta\theta_{y|x}$ is made such that $KL(\theta_{y|x} + \delta\theta_{y|x})$ either reaches a local minimum or reduces to a certain extent. Here, the integral over $\mathbf{y}$ for getting $q(\mathbf{x})$

by Eq. (22.46) is avoided. However, we encounter not only the integrals in Eq. (22.63) but also the following one

$$H(\theta_{y|x}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \ln p(\mathbf{y}|\mathbf{x}) d\mathbf{x} d\mathbf{y}. \tag{22.66}$$

It was also firstly suggested in 1997 under the name of the mean field approximation (see the choice (3) of Table 2(C) in [22.76] and the choice (c) in Eq. (24) of [22.66]) that we get Y_t consisting of only one mean point

$$y_t = \int y p(y|x_t) dy \tag{22.67}$$

which is computable for a $p(y|x)$ given by either of Eq. (22.20) and Eq. (22.21), but not applicable to a B-architecture with $p(y|x)$ given by Eq. (22.46) since another integral has to be encountered to get $q(x)$.

– In implementing the harmony learning with Eq. (22.41) for parameter learning, the Yang step is simply getting Y_t that consists of only one peak point

$$y_t = arm \max_y p(y|x_t). \tag{22.68}$$

which was firstly suggested again in 1997 (see the choice (2) of Table 2(C) in [22.76] and the choice (b) in Eq. (24) of [22.66]) and then further encountered in Eq. (22.36) on a B-architecture. This nonlinear optimization is implemented in help of an iterative procedure:

$$\mathbf{y}^{new}(\mathbf{x}_t) = ITER(\mathbf{y}^{old}(\mathbf{x}_t)). \tag{22.69}$$

Specific algorithms of this type are proposed in [22.57, 22.54] to suit specific structures of $q(\mathbf{x}|\mathbf{y})$ and $q(\mathbf{y})$. The complexity of making a nonlinear optimization is considerably less than that of making the integrals over $\mathbf{y}$. Moreover, we usually need only a few iterations by Eq. (22.69) instead of waiting it to converge. This is another salient advantage that the least complexity nature of Eq. (22.36) provides us, in addition to making model selection possible.

– In implementing the harmony learning on a BI-architecture, the Yang step consists of two parts. One is simply implementing Eq. (22.68). For a $p(y|x)$ given by Eq. (22.21), we can get $y_t = y(x_t)$ via a simple comparison that reduces significantly the computational complexity for making a nonlinear optimization by Eq. (22.36). For a $p(y|x)$ given by Eq. (22.20), we can get either

$$y^{(j)} = \pi_j(x_t), \ or \ \ y^{(j)} = \begin{cases} 1, & \pi_j(x_t) \geq 0.5, \\ 0, & \text{otherwise.} \end{cases} \tag{22.70}$$

The second part attempts to increase $\ln\left[q(\mathbf{x}|\mathbf{y})q(\mathbf{y})\right]_{\mathbf{y}=f_{j^*(x)}(\mathbf{x}|\theta_{y|x,j^*(x)})}$ or $\ln\left[q(\mathbf{x}|\mathbf{y})q(\mathbf{y})\right]_{\mathbf{y}=\pi(x_t,\theta)}$. In help of the chain rule for gradient, it can be implemented via an update $\delta\theta_{y|x,j^*(x)}$ in the form of Eq. (22.64) as follows

$$\delta\theta_{y|x,j^*(x)} = \eta_t(y) g^T_{\theta_{y|x,j^*(x)}}[\psi(y) + \phi(y)],$$

$$\psi(y) = \frac{\partial \ln q(x|y)}{\partial y}, \ \phi(y) = \frac{\partial \ln p(y)}{\partial y}, \tag{22.71}$$
$$g_{\theta_{y|x,j^*(x)}} = \nabla_{\theta_{y|x,j^*(x)}} f_{j^*(x)}(x|\theta_{y|x,j^*(x)}), \ \mathit{or} \ \ g_\theta = \nabla_\theta \pi(x|\theta).$$

22.8 Learning Implementation: From Optimization Search to Accumulation Consensus

By Eq. (22.62), the process of implementing parameter learning can be regarded as an optimization process. Actually, many learning tasks are implemented in a two-stage way with its first stage to form an objective function $J(\theta)$ and with its second stage to maximize $J(\theta)$. At the second stage, any optimization techniques developed in the nonlinear programming literature are possible to be adopted for solving the problems. One typical class of techniques is the gradient-based search, that is,

$$\theta^{new} = \theta^{old} + \eta \nabla_\theta J(\theta), \tag{22.72}$$

where $\eta > 0$ is a small number as a pre-specified step size. The updating is iterated until it is converged. Such a process is featured by one candidate solution θ^{old} that is updated locally and iteratively, and also terminated according to certain local property (i.e., it is a local maximum). However, the local feature makes the process be easy to be trapped at a local maximum.

This problem may be remedied by implementing multiple search processes of the type by Eq. (22.72) either in parallel or subsequently. In parallel, it means that a set Θ_c of candidate solutions are considered with each being iterated via the same way as in Eq. (22.72). Subsequently, it means that once a convergence is reached, certain perturbation is made on the converged solution to get a new initial candidate of the next iterative process, and selecting the best among all the converged local maxima as the final solution. The chance of finding the global solution increases as the number of such search processes increases. However, the computational costs also increase significantly. In the worst case, an exhaustive search of the entire space of θ will guaranteed to find the global solution.

However, optimization should not be understood being same as learning. The key challenge of learning is to find a principle for designing an objective function such that the learning model can both fit a given set of training samples and generalize as good as possible on new samples that are different from the given training sample set but come from the same underlying distribution. Even assuming such an objective function already available, learning is also not simply an optimization.

Being different from an objective function $J(\theta)$ for optimization, an objective function $J(\theta|\mathbf{X})$ is not only a function of θ but also bases on a set $\mathbf{X}$ of samples. Of course, we can treat $J(\theta|\mathbf{X})$ as $J(\theta)$ with learning made as an optimization problem. However, we are not bounded to only this way. We can explore the specific structure of $J(\theta|\mathbf{X})$ for a different implementation. It can be observed, from not only Eq. (22.29) or Eq. (22.45) but also Eq. (22.30) and Eq. (22.35) with $z_q = 1$ as well as from many existing learning approaches, that an objective function formed under a learning principle usually has an additive decomposition form

$$J(\theta|\mathbf{X}) = \int p(\mathbf{x}|\mathbf{X})J(\theta|x)dx = \tfrac{1}{N}\textstyle\sum_{t=1}^{N} J(\theta|x_t), \tag{22.73}$$

with $p(\mathbf{x})$ estimated by the empirical density Eq. (22.1).

First, this additive decomposition makes learning able to be implemented adaptively. Instead of forming the function $J(\theta|\mathbf{X})$ based on the entire set $\mathbf{X}$ and then making an optimization, a candidate solution is updated to adapt each current coming sample x_t such that the objective function $J(\theta|x_t)$ per each sample x_t increases by certain extent, e.g., via Eq. (22.72) but with $J(\theta)$ replaced by $J(\theta|x_t)$. This is usually called adaptive learning that is different from directly optimizing $J(\theta)$ in that the objective function $J(\theta|x_t)$ per each sample x_t actually varies. Since each sample x_t comes via random sampling, this learning process is also called stochastic approximation in the related literature [22.37, 22.24]. Extension can also be made by each time considering a set X_t that consists of several samples. Then, an adaptation is made by Eq. (22.72) but with $J(\theta)$ replaced by $\sum_{x\in X_t} J(\theta|x)$. With the learning step size η reduces as learning proceeds, such an iteration will be terminated. However, whether the resulted solution is same as a local or global maximum solution of $J(\theta|\mathbf{X})$ usually turned out a difficult problem, except some simple cases.

Second, the additive decomposition in Eq. (22.73) also provides a possibility to develop a completely different solving strategy called *accumulation consensus* that consists of the following ingredients:

(a) *Quantization* Considering a bounded space of θ in which the solution is located, we quantize this bounded space into a set of discrete points with a pre-specified resolution $\delta > 0$. Each of these discrete points is considered as a candidate solution with an accumulator attached. All the accumulators forms an accumulation array.

(b) *Sampling* Getting a sample x_t from the sample set $\mathbf{X}$.

(c) *Voting and accumulation* At each time t, we let every accumulator in the accumulation array to be increased by a score of $J(\theta|x_t)$ with θ being the discrete point that the accumulator locates at.

(d) *Selection and Testing* After a certain period, the accumulator with the largest score is selected among the accumulation array and the corresponding discrete point is tested by certain local statistical properties. If passed, it is taken as the solution.

The main problem of this accumulation procedure is that the size of the accumulation array increases rapidly with a fine resolution $\delta > 0$ and exponentially with the number of unknown parameters. Also, voting and accumulating should be made on all the accumulators.

However, to implement the BYY harmony learning by Eq. (22.35) on those finite mixture related learning tasks in Sect. 22.3.2, we are lead to a very different situation. With $q(\mathbf{y}) = q(y = \ell) = \alpha_\ell$ and $q(\mathbf{x}|\mathbf{y}) = q(x|\phi_\ell)$, the marginal density by Eq. (22.17) becomes equivalent to the finite mixture model by Eq. (22.14), which includes not only various Gaussian mixture based tasks for density estimation and clustering analysis [22.54, 22.58, 22.77] but also various multi-sets mixture based tasks for curve and object detection [22.52]. Moreover, we have that Eq. (22.36) becomes

$$p(\mathbf{y}|\mathbf{x} = x_t) = \delta(y - \ell_t), \ \ell_t = arg \max_\ell [q(x_t|\phi_\ell)\alpha_\ell], \tag{22.74}$$

and we further have that Eq. (22.73) becomes

$$J(\theta|\mathbf{X}) = \tfrac{1}{N} \textstyle\sum_{t=1}^N \ln [q(x_t|\phi_{\ell_t})\alpha_{\ell_t}], \ \theta = \{\phi_\ell, \alpha_\ell\}_{\ell=1}^k. \tag{22.75}$$

As a result, the above voting and accumulating mechanism can be considerably simplified. Only one of k parameter sets $\{\phi_\ell, \alpha_\ell\}_{\ell=1}^k$ will be voted at each time t. Thus, it only needs to consider accumulating the votes on $\phi_{\ell_t}, \alpha_{\ell_t}$. Moreover, considering that the k sets of parameters share a same format ϕ, α and thus the dimension of accumulation array can be reduced by k times, we can simplify the above accumulation procedure as follows:

(a) *Quantization* Considering a bounded space of ϕ, α that is quantized.

(b) *Sampling* the same as above.

(c) *Voting and accumulation* At each time t, we let every accumulator, that locates at $\theta = [\phi, \alpha]$, to be increased by a score of $\ln [q(x_t|\phi)\alpha]$.

(d) *Selection and Testing* the same as above.

Specifically, the test can base on a known statistical property of $q(x_t|\phi)$. For simplicity, the test can also be ignored.

In the particular case that $\alpha_\ell = 1/k$ and each $q(x_t|\phi_\ell) = \delta(g(x_t, \phi_\ell))$ (i.e., $g(x_t, \phi_\ell) = 0$ is a deterministic equation), only the accumulators locate at those values of ϕ_ℓ that satisfing $g(x_t, \phi_\ell) = 0$ are voted and accumulated. This case leads us to the well known existing techniques developed in the literature of pattern recognition. Specifically, at each time t if every sample in the sample set $\mathbf{X}$ is enumerated, the above accumulation procedure will become equivalent to the well known Hough Transform (HT) for curve detection [22.20, 22.21]. For example, when x_t is of a 3-dimension and $g(x_t, \phi_\ell) = 0$ is a line, we are lead to a HT for detection lines on an image. Furthermore, if each x_t is sampled randomly and the period in (d) covers an enough number of samples, we are lead to a probabilistic variant of the HT [22.16].

The above accumulation procedure extends the HT and probabilistic HT. Not only α_ℓ is no longer constrained to be equally $1/k$ such that the number of pixels or samples on an object is considered together with their fitting errors. But also with a probabilistic $q(x_t|\phi_\ell)$ in place of the deterministic model $g(x_t, \phi_\ell) = 0$ such that random noises and variations are taken in consideration. For example, objects such lines, circles, ellipses, and even complicated shapes can be detected under strong noises. Moreover, we can further reduce the complexity for voting and accumulating because only those accumulators that locate on a smaller subset θ_t are increased by a score of $J(\theta|x_t)$, where θ_t is given as follows:

$$\theta_t = \{\phi, \alpha : q(x_t|\phi)\alpha \geq b, \quad b > 0 \text{ is a pre-specified threshold,}\} \tag{22.76}$$

which degenerates back to the entire space when $b = 0$.

One variant of the above accumulation procedure is obtained by modifying its sampling mechanism. At each t, a set X_t of several samples are sampled randomly from $\mathbf{X}$ instead of only one sample x_t is picked. In voting and accumulating, if the above hard-cutting is not considered, we simply let every accumulator in the accumulation array to be increased by a score $\sum_{x \in X_t} J(\theta|x)$ in place of $J(\theta|x_t)$. However, situation will become quite different by considering accumulators only on

$$\theta_t = \{\phi, \alpha : \; q(x|\phi)\alpha \geq b, \; \forall x \in X_t\}. \tag{22.77}$$

As $b > 0$ increases and the number of samples in X_t increase, the cardinality of θ_t will decrease, even possibly toward 1 or 0. Consequently, the voting mechanism changes from one-to-many diverging mapping to many-to-one or many-to-few converging mapping. As a result, the high space complexity of the above quantization based accumulation array can be replaced by a dynamic accumulator structure A. At each time t, we increase the accumulator at θ by a score $\sum_{x \in X_t} J(\theta|x)$. If such an accumulator has not been included in A yet, we add it in. Thus, the number of accumulators in A grows dynamically as the voting proceeds. Finally, solutions are selected among A either via those accumulators with largest votes or via the cluster centers of accumulators.

This modified accumulation procedure becomes equivalent to the randomized Hough Transform (RHT) [22.96, 22.95] in the particular case that each $q(x_t|\phi_\ell) = \delta(g(x_t, \phi_\ell))$ and $\alpha_\ell = 1/k$. The RHT was developed as an important advance along the research direction of the conventional HT. Moreover, the accumulation procedure here extends the RHT approach such that not only the number of pixels or samples on an object is considered together with their fitting errors but also random noises and variations are taken in consideration.

Further improvements are possible along the following directions:

(1) Instead of deciding solution only based on those accumulators with largest votes or the cluster centers of accumulators, a statistical test can be imposed by jointly considering the local fitting property via $q(x_t|\phi_\ell)$ and the global property via α_ℓ.

(2) Instead of implementing learning only based on either optimization search or accumulation consensus, possible combinations of two can be investigated. For example, the final results of the above accumulation procedure can also be used as initial points of a local adaptation based optimization search to further refine these solutions. Also, the accumulation consensus may

be implemented together with a parallel local adaptation based optimization search.

(3) Instead of pre-specifying a threshold $b > 0$, b may also be dynamically controlled during the above accumulation procedure.

22.9 Main Results and Bibliographic Remarks

22.9.1 Main Results on Typical Learning Problems

Bayesian Ying Yang (BYY) harmony learning was firstly proposed in 1995 [22.88] and then developed in past several years. New results can be summarized on two aspects. One aspect is on those advances of the BYY harmony learning as a general statistical learning framework with a new mechanism for model selection and regularization, the details will be introduced in next chapter of this book. The other aspect is on various specific cases of the BYY harmony learning, which lead to various specific learning algorithms as well as the detailed forms for implementing model selection and regularization, which covers three main statistical learning paradigms, namely, unsupervised learning, supervised learning, and temporal modeling. The major results are summarized into the following list.

(a) Gaussian mixture and multi-sets-mixture based structures

– Smoothed EM (see Eq. (18) in [22.75]), Robust EM (see Eq. (33) in [22.77]), Hard-cut EM (see [22.88, 22.82]), RPCL learning, Normalization RPCL-type EM [22.58, 22.54, 22.51].
– Elliptic clustering [22.58, 22.54] and multi-sets mixture based multiple object detection (see Sect. 3 in [22.82] and also see Sect. 3.3 in [22.52]).
– Criteria for the number k of clusters and of Gaussians [22.88, 22.82, 22.85, 22.77].
– Adaptive learning with automatic selection on k [22.88, 22.82, 22.77, 22.58, 22.54].
– Support vector based Parzen window estimate [22.58, 22.54].

(b) Independent structures

– Adaptive EM-like algorithms for independent binary or Bernoulli FA (BFA) with hidden factors selected in either of two ways, that is, selected either automatically during learning or alternatively via a criterion obtained from Eq. (22.42)(see Sect. 3.2(B) in [22.71], Sect. 4.2.2 & 4.2.4 and Fig. 2 in [22.66]).
– New insight on LMSER learning [22.94] for ICA and new adaptive algorithm with hidden factors selected in either of two ways (see Sect. 8 in [22.74], Sect. 4.3.4 in [22.66], and also the recent [22.54, 22.51]).
– Extensions of the LMSER learning (see Sect. 3 in [22.66] and Sect. 3 in [22.68]).

- Factor analysis and PCA with subspace dimension determined in either of two ways (see Sect. 3.2(B) in [22.71], also see Sect. 4.2.4 in [22.66]).
- Adaptive algorithms that are able to effectively implement the non-Gaussian factor analysis (NFA), with hidden factors selected in either of two ways [22.57, 22.54, 22.51, 22.53].
- A learned parametric model (LPM) ICA algorithm that it works on any combinations of super-Gaussian sources and sub-Gaussian sources [22.76, 22.78, 22.79, 22.72].
- The one-bit conjecture and its proof [22.26].

(c) Mixtures of independent structures

- Smoothed EM, Robust EM, Hard-cut EM, RPCL learning, Normalization RPCL-type EM, with the number of local models and the subspace dimension of each model determined in either of two ways [22.54, 22.51].
- Extensions of BFA, NFA, LMM-ICA, LMSER to their local versions, with both the number of local models and the number hidden factors selected in either of two ways (see Item 7.6 and Item 7.7 in [22.74], and the recent [22.54, 22.51, 22.52]).
- Extensions to self-organizing maps in collaboration with the BYY harmony learning [22.54].

(d) Mixture-of-experts (ME), RBF nets, and kernel regression

- Cooperative competitive learning (CCL) for ME learning, with the number of experts determined in either of two ways (see Sect. 4 in [22.82] and Table 7 in [22.76], also see Sect. 2.2 in [22.67]).
- An alternative ME model [22.92] with its ML learning exactly implemented by the EM algorithm. Criterion for the number of experts, as well as CCL learning algorithms and normalization RPCL-type EM, with the number of experts determined in either of two ways (see Sect. 4.3 in [22.83] and Sect. 3.2 in [22.67], also see [22.58, 22.54]).
- Variants on both the ME model and the alternative ME model via introducing regularization of either data smoothing or normalization.
- RBF nets are linked to the alternative ME model as special cases. Instead of the conventional two step learning, i.e., making clustering and then least square learning, learning is implemented not only by EM in exactly a maximum likelihood sense with the number of basis functions selected by a simple criterion from Eq. (22.42), but also by adaptive algorithms in a BYY harmony learning sense with the number of basis functions determined automatically during learning. Moreover, extensions are made by introducing regularization of either data smoothing or normalization [22.67, 22.58, 22.54].
- As the special cases of the normalized radial basis function (RBF), we can get a support vector based kernel regression [22.58, 22.54].

(e) Three layer nets and hidden units

- Regularized ML learning obtained via either normalization or data smoothing [22.51].
- Three layer nets with hidden units are linked to independent factor structures as extended cases [22.51, 22.53].
- Easy implementing criterion obtained from Eq. (22.42) for determining hidden units.
- An EM-like algorithms for a three layer net of stochastic hidden unit in multivariate Bernoulli, with the number of hidden units determined (see Table 4 and 5 in [22.76]).

(f) Temporal modeling

- Extensions of BFA to independent HMM and higher order independent HMM (see Item 4.1 in [22.75], Sect. 2.4 in [22.65], Eq. (26) in [22.61], and pages 839-849 of [22.57]).
- Extension of ICA to temporal ICA for blind separation of temporal sources (see Eq. (29a) and (29b) in [22.65], Sect. 3 in [22.65], Sect. IV(A) & (B) in [22.59], and Sect. III(B) in [22.57]).
- Extensions of FA to not only Kalman filter, but also temporal factor analysis (TFA) that has no rotation indeterminacy and, thus, can be used for both not only implementing a real BSS with noise and but also making model identification (see Sect. 2.3.1 in [22.65]), Sect. IV(C) in [22.59], and Sect. III(A) in [22.57]).
- Extensions of TFA to temporal NFA, with the driving noises of the autoregressive state space being non-Gaussian and modeled via Gaussian mixtures (see Sect. 4 in [22.61] and Sect. IV(B) in [22.59]).
- Easy implementing criterion obtained from Eq. (22.42) for determining hidden states.
- Adaptive algorithms with the number of states determined automatically during learning (see [22.57, 22.53]).
- Temporal extensions of LMSER, competitive ICA, local FA and local NFA to [22.57, 22.53].

These results have been obtained as the progress of studies on BYY system and harmony learning. Bibliographic remarks on the progress are provided in the following two subsections from both the aspect of BYY system with the KL learning and the aspect of computing techniques for implementing BYY learning. In the next chapter of the present book, bibliographic remarks will be also provided on the progress from the model selection and regularization aspects of BYY harmony learning.

22.9.2 Bibliographic Remarks on BYY System with KL Learning

As discussed in Sect. 22.5, acting as a unified framework for understanding several existing major learning models, the BYY learning was firstly published in 1995 with the following main issues [22.88]:

- The basic system as shown in Fig. 22.1 and three typical architectures as shown in Fig. 22.2 were firstly proposed and studied under the KL learning by Eq. (22.45).
- The equivalence of the KL learning on a B-architecture to the ML learning on $q(x)$ was found with Eq. (22.46), with the EM algorithm [22.9] revisited and an alternative but much simpler mathematical proof on its convergence (see Sect. 3.1 in [22.88], especially the part after Eq. (9) in [22.88]).
- The relationship of this KL learning on a BI-architecture to the Helmholtz machine learning and variational approximation [22.40, 22.18] was established (see Sect. 3.4 in [22.88]).
- The relationship of this KL learning on a F-architecture to the maximum information preservation were established (see Sect. 3.3 in [22.88]).

In the same year, the key ideas were also proposed for extending the BYY system under the KL learning by Eq. (22.45) to supervised learning (see Sect. 5 in [22.90]) and temporal modeling (see [22.89] and Sect. 4 in [22.90]). Specifically, advances of BYY system under the KL learning can be summarized along the following directions:

(1) The equivalence between the KL learning on a B-architecture and the ML learning on the marginal density $q(x)$ was further elaborated as Theorem 1 in [22.77]. The Ying-Yang procedure discussed in Sect. 22.7 provides the general form of the EM algorithm [22.9] from a new perspective, which not only leads us to revisit the detailed forms of the EM algorithm on Gaussian mixture [22.38], mixture of experts [22.13, 22.14, 22.92], factor analysis [22.39], etc., but also brings us new results on several typical learning models in two aspects:

(a) On those models that have not taken the advantages of the EM algorithm yet, the detailed forms of the EM algorithm are developed for implementing the ML learning. The following are two typical examples.
  - The so-called multi-sets modeling is proposed in 1994 for modeling these objects in typical shapes such as lines, circles, and ellipses, as well as pre-specified templates [22.93] and [22.91]. At the beginning, its learning is formulized as an extension of the conventional mean square error (MSE) clustering analysis and then implemented via a generalized version of the well known KMEAN algorithm [22.10]. In help of a link, that was firstly built in [22.88], between the ML learning with the EM algorithm on Gaussian mixture and the MSE clustering with the KMEAN algorithm, this type of multi-sets is represented into

a finite mixture that is also recently called multi-sets-mixture [22.52], the detailed forms of the EM algorithm are proposed for implementing its ML learning (see Sect. 3.2 in [22.88] and Sect. 3 in [22.82], also see the recent elaboration in [22.52]).
- The alternative model [22.92] of the mixture-of-experts (ME) [22.12, 22.13] with the EM algorithm for its ML learning has been revisited. As special cases of this alternative ME model, the detailed forms of the EM algorithm is developed for implementing the ML learning on the normalized RBF nets, the extended normalized RBF nets, as well as their elliptic extensions (see Sect. 4 in [22.82] and Table 7 in [22.76], also see Sect. 2.2 in [22.67]), in place of the conventional sub-optimal two stage training algorithm (i.e., making clustering to locate basis functions and then making linear regression for the output layer).

(b) The above EM algorithms as well as those existing EM algorithms on several typical learning models are further extended to adaptive versions (i.e., adaptation is made per a sample). Typical examples include those adaptive algorithms on:
- Gaussian mixture (see Sect. 6.1 in [22.88]),
- factor analysis (see Sect. 3.2(B) in [22.71], also see Sect. 4.2.4 in [22.66]),
- binary factor analysis (see Sect. 4.2.2 and Fig. 2 in [22.66]),
- local PCA (see Item 7.6 and Item 7.7 in [22.74]),
- multi-sets mixture (see Sect. 3 in [22.82] and also see Sect. 3.3 in [22.52]),
- mixture-of-experts model and alternative mixture-of-experts model (see Sect. 4.3 in [22.83] and Sect. 3.2 in [22.67]).

(2) Following a link initially made in [22.88] on the relationship between the KL learning with a BI-architecture and the Helmholtz machine learning, the detailed relationship between BYY system and Helmholtz machine is further explored via a special BI-architecture for factorial encoding that leads both special cases of the Helmholtz machine with one hidden layer and a regularized version of LMSER [22.94] and auto-association [22.6]. This regularized LMSER actually performs jointly a noisy-ICA and a LMSER learning (see Sect. 8 in [22.74] and Sect. 4.3.4 in [22.66]). Its learning is implemented either via a Monte-Carlo technique (see Table 2 in [22.76]) or in help of a mean-field approximation based EM like algorithm (see Sect. 4.3.2 in [22.66]). With a binary inner representation replaced with a real non-Gaussian, two other advances have been achieved. One is a Monte-Carlo based EM algorithm for implementing independent factor analysis with non-Gaussian factors (see Sect. 3 in [22.66] and Sect. 3 in [22.68]) and the other is another extension of nonlinear LMSER learning with a real non-Gaussian inner representation (see Sect. 3.4 in [22.66]).

(3) Following a link initially made also in [22.88] on the relationship the relationship between the KL learning with a F-architecture and the maximum information preservation, subsequently it is further obtained that

- The KL learning on a joint special case of both BYY F-architecture and BYY BI-architecture leads the widely studied information theoretic based ICA [22.86].
- With $q(y)$ being a product of independent factors in a parametric model that is learned together with the F-architecture, we were motivated on getting an important improvement on the information theoretic based ICA, under the name Learned Parametric Mixture based ICA (shortly LMP-ICA) algorithm [22.87] that works well on any combination of super-Gaussian or sub-Gaussian sources. Each independent factor can be modeled via either a mixture of CDF [22.78, 22.79] or a mixture of Gaussian mixture (see Table 3 in [22.76] and [22.72]).
- This LMP-ICA has also been extended to the cases with observation noises via implementing jointly a noisy-ICA and either a non-Gaussian independent factor analysis or a LMSER learning (see Eq. (10.8) and Eq. (10.9) in [22.74]).

(4) The KL learning on a specific B-architecture also leads to the ML learning and particularly the mean square error learning on a three layer network (see Table 5(1) in [22.76]). With two types of stochastic hidden units, the ML learning on three layer networks can be made by an EM like algorithm (see Table 4 in [22.76]), which is implemented via either a Monte-Carlo technique (see Table 5(4) in [22.76]) or a mean-field based technique (see Table 5(3) in [22.76]).

(5) Studies on BYY systems for temporal modeling started from 1995 too (see [22.89] and Sect. 4 in [22.90]), in which a preliminary framework was proposed and the KL learning on temporal BYY system with a B-architecture was shown to be equivalent to the ML learning on a temporal process. To implement the learning, the following studies have been made:

- The KL measure on the entire temporal process has been turned into a summation of the KL measure at every time instance t weighted by a transfer probability from its past to the moment t.
- The integrals over each inner representation y_t has been approximated via a Monte-Carlo technique (see Sect. 5 in [22.75]).
- A fast implementation is further developed with the integrals over each y_t being approximated in help of a so-called CRP approximation that includes the mean-field approximation as one of three typical examples (see Sect. 2.1 in [22.65]). Then, this CRP approximation is mathematically justified by considering $\int p(u)T(u)du$ via the first order Taylor expansion of $T(u)$ around $\hat{u} = \int up(u)du$ (see Sect. 2.3 in [22.64]).
- A better approximation of the integrals over each y_t has also be considered via the above Taylor expansion of $T(u)$ up to the second order (see Sect. 2.3 in [22.64] and Sect. II(D) in [22.59]).

(6) Furthermore, two types of particular temporal models have been studied.

(a) One is that y is real and described by a state space model of an auto-regressive (AR) process (e.g., a first order AR process) even the first order auto-regressive process, with the following results:
   - The well known Kalman filter on the state space is revisited as a special case with extensions obtained (see Item 4.2 in [22.75], Sect. 2.2 in [22.65] and [22.64], and Sect. III(B) in [22.57]).
   - ICA is extended to temporal ICA (see Eq. (29a) and (29b) in [22.65]), and Sect. 3 in [22.65], also see Sect. IV(A) and (B) in [22.59] and Sect. III(B) in [22.57]).
   - The temporal factor analysis (TFA) has been proposed not only as an extension of factor analysis with its rotation indeterminacy solved by temporal relation and but also as an extension of Kalman filter with the state space parameters determined via learning instead of requiring them known in advance (see Sect. 2.3.1 in [22.65]), Sect. IV(C) in [22.59], and Sect. III(A) in [22.57]).
   - Temporal noisy-ICA that implements jointly a temporal ICA and a temporal extension of non-Gaussian independent factor analysis (see Sect. 4 in [22.61] and Sect. IV(B) in [22.59]).

(b) The other type is that y is discrete or binary and described by a Hidden Markov process, with the following results:
   - The well known Hidden Markov model (HMM) is revisited when y takes a finite number of discrete labels, and the Ying-Yang procedure leads to the well known Baum-Welch algorithm (see Item 4.1 in [22.75] and Sect. 2.4 in [22.65]). Moreover, with the CRP approximation, a fast algorithm was suggested for approximately implementing the ML learning (see Sect. 2.4 in [22.65]).
   - Temporal extension of binary FA is made into a type of independent HMM, with its learning implemented by an EM-like algorithm (see Eq. (26) in [22.61] and page 839–849 of [22.57]).
   - When a B-architecture is replaced with a BI-architecture for directly mapping $x_t \to y_t$, temporal extension of the previously discussed regularized LMSER is made into another type of independent HMM, with its learning implemented by an EM-like algorithm (see Sect. 4 in [22.64], Sect. IV(D) in [22.59], and Sect. III(B) in [22.57]).

(7) Extensions of the KL learning by Eq. (22.45) has also been made with the KL-divergence replaced by f-divergence and the so-called weighted EM algorithm was firstly proposed (see Item 1.4 in [22.75] and Sect. 4 in [22.75]), supported by experiments on Gaussian mixture [22.77] and ICA problems [22.80] with advantage of being robust to outliers. Further studies have also been made in [22.69, 22.70, 22.62, 22.63], especially with a systematic summary in [22.60].

22.9.3 Bibliographic Remarks on Computing Techniques

During the studies on BYY learning, the following specific computing techniques have also been developed to support implementation:

– **Ying-Yang Alternative implementation**, as given by Eq. (22.62), provides a unified implementation procedure that facilitates the learning on such a paired system. It not only includes the EM algorithm [22.9] as a special case but also applies to various cases of learning on a BYY system under either KL-divergence or harmony measure as well as other learning costs.
– The technique of approximating $\int p(u)T(u)du$ via the first order Talyor expansion of $T(u)$ around $\hat{u} = \int up(u)du$ (see Sect. 2.3 in [22.64]) provides a very useful tool for tackling the integrals of $\int p(u)T(u)du$ type, which takes an important role in implementation of temporal BYY learning and data smoothing regularization.
– The peak finding problem in Eq. (22.36) takes an important part in implementation of BYY harmony learning, which is solved by techniques given in Table 1 of [22.57].
– An iterative updating on a covariance matrix Σ has to ensure its positive definite nature. It is usually not guaranteed by a updating rule with both learning and de-learning. Two techniques was firstly proposed in Table 1 of [22.58]. Namely, one bases on the decomposition $\Sigma = BB^T$, which have been adopted in [22.54, 22.56, 22.51, 22.52] and the other bases on the eigen-decomposition, which have been adopted in [22.27, 22.28, 22.29].
– The accumulation consensus technique discussed in Sect. 22.8 can be traced back to the RHT [22.96, 22.95] for detecting line and curves. The RHT opened one new direction on studies of the conventional HT, which has been widely studied and applied to various problems of curve detection and object detection.
– The multi-set mixture based learning was developed in [22.93, 22.91]. The link between RHT and multi-sets mixture based learning was built in [22.52] where the key ideas discussed in Sect. 22.8 were firstly initialized.

22.10 Conclusions

Various dependence structures among data are important to many tasks of statistical learning and data mining. Undertaking both a survey on major tasks of dependence structure mining and a summary on fundamentals and main results of BYY harmony learning, we observe that the BYY harmony learning provides a unified framework for various dependence structures, with new mechanisms for model selection and regularization on a finite size of samples.

References

22.1 H. Akaike: A new look at the statistical model identification, IEEE Tr. Automatic Control, 19, 714-723 (1974)

22.2 SI. Amari, A. Cichocki, HH. Yang: A new learning algorithm for blind separation of sources. In: DS Touretzky et al. (eds.) *Advances in Neural Information Processing 8,* MIT Press, 757-763 (1996)

22.3 TW. Anderson, H. Rubin: Statistical inference in factor analysis, *Proc. Berkeley Symp. Math. Statist. Prob. 3rd 5*, UC Berkeley, 111-150 (1956)

22.4 A. Bell, T. Sejnowski: An information maximization approach to blind separation and blind deconvolution, Neural Computation, 17, 1129-1159 (1995)

22.5 CM. Bishop: Training with noise is equivalent to Tikhonov regularization, Neural Computation 7, 108-116 (1995)

22.6 H. Bourlard, Y. Kamp: Auto-association by multilayer Perceptrons and singular value decomposition, Biol. Cyb. 59, 291-294 (1988)

22.7 P. Comon: Independent component analysis - a new concept? Signal Processing, 36, 287-314 (1994)

22.8 KY. Chan, WS. Chu, L. Xu: Experimental Comparison between two computational strategies for topological self-organization, *Proc. of IDEAL03*, Lecture Notes in Computer Science, LNCS 2690, Springer-Verlag, 410-414 (2003)

22.9 AP. Dempster, NM. Laird, DB. Rubin: Maximum-likelihood from incomplete data via the EM algorithm, J. Royal Statistical Society, B39, 1-38 (1977)

22.10 PA. Devijver, J. Kittler: Pattern Recognition: A Statistical Approach, Prentice-Hall (1982)

22.11 RO. Duda, PE. Hart: *Pattern classification and Scene analysis* (Wiley, 1973)

22.12 RA. Jacobs et al.: Adaptive mixtures of local experts, Neural Computation, 3, 79-87 (1991)

22.13 MI. Jordan, RA. Jacobs: Hierarchical mixtures of experts and the EM algorithm, Neural Computation, 6, 181-214 (1994)

22.14 MI. Jordan, L. Xu: Convergence results for the EM approach to mixtures of experts, Neural Networks, 8, 1409-1431 (1995)

22.15 C. Jutten, J. Herault: Independent Component Analysis versus Principal Component Analysis, *Proc. EUSIPCO88*, 643-646 (1988)

22.16 H. Kälviäinen, P. Hirvonen, L. Xu, E. Oja: Probabilistic and Non-probabilistic Hough Transforms: Overview and Comparisons, *Image and Vision Computing*, Vol. 5, No. 4, pp. 239-252 (1995)

22.17 J. Han, M. Kamber: *Data Mining: Concepts and Techniques* (Morgan Kaufmann, 2001)

22.18 GE. Hinton, P. Dayan, BJ. Frey, RN. Neal: The wake-sleep algorithm for unsupervised learning neural networks, Science, 268, 1158-1160 (1995)

22.19 H. Hotelling: Simplified calculation of principal components, Psychometrika, 1, 27-35 (1936)

22.20 P.V.C. Hough: Method and means for recognizing complex patterns, *U.S. Patent 3069654* (Dec.18, 1962)

22.21 J. Illingworth, J. Kittler: A survey of the Hough Transform, Comput. Vision Graphics and Image Process, 43, 221-238 (1988)

22.22 FV. Jensen: *An introduction to Bayesian networks* (University of Collage London Press) (1996)

22.23 T. Kohonen: *Self-Organizing Maps* (Springer-Verlag, Berlin, 1995)

22.24 H. Kushner, D. Clark: *Stochastic approximation methods for constrained and unconstrained systems* (New York: Springer) (1998)

22.25 HY. Kwok, CM. Chen, L. Xu: Comparison between Mixture of ARMA and Mixture of AR Model with Application to Time Series Forecasting, *Proc. ICONIP'98*, Oct.21-23, 1998, Kitakyushu, Japan, Vol. 2, 1049-1052

22.26 ZY. Liu, KC. Chiu, L. Xu: The One-bit-Matching Conjecture for Independent Component Analysis, Neural Computation, Vol. 16, No. 2, pp. 383-399 (2003)

22.27 ZY. Liu, KC. Chiu, L. Xu: Strip Line Detection and Thinning by RPCL-Based Local PCA, Pattern Recognition Letters, 24, pp. 2335-2344 (2003)

22.28 ZY. Liu, KC. Chiu, L. Xu: Improved system for object detection and star/galaxy classification via local subspace analysis, Neural Networks, 16, 437-451 (2003)

22.29 ZY. Liu, L. Xu: Smoothed Local PCA by BYY data smoothing learning, *Proc ICCAS 2001*, Jeju, Korea, Oct.17-21, 2001, pp. 924-927

22.30 J. Ma, T. Wang, L. Xu: A gradient BYY harmony learning rule on Gaussian mixture with automated model selection, Neurocomputing, 56, 481-487 (2004)

22.31 Ch. von der Malsburg: Self-organization of orientation sensitive cells in the striate cortex, Kybernetik 14, 85-100 (1973)

22.32 R. McDonald: *Factor Analysis and Related Techniques* (Lawrence Erlbaum)

22.33 GJ. McLachlan, T. Krishnan: *The EM Algorithm and Extensions*, John Wiley & Son, INC (1997)

22.34 E. Oja: *Subspace Methods of Pattern Recognition* (Research Studies Press, UK 1983)

22.35 J. Pearl: *Probabilistic reasoning in intelligent systems: networks of plausible inference* (San Francisco, CA: Morgan Kaufman 1988)

22.36 L. Rabiner, BH. Juang: *Fundamentals of Speech Recognition*, Prentice Hall, Inc. (1993)

22.37 H. Robbins, S. Monro: A stochastic approximation method, Ann. Math. Statist., 22, 400-407 (1950)

22.38 RA. Redner, HF. Walker: Mixture densities, maximum likelihood, and the EM algorithm, SIAM Review, 26, 195-239 (1984)

22.39 D. Rubi, D. Thayer: EM algorithm for ML factor analysis, Psychometrika, 57, 69-76 (1976)

22.40 L. Saul, MI. Jordan: Exploiting tractable structures in intractable Networks, Advances in neural information processing systems, 8, MIT Press, 486-492 (1995)

22.41 C. Spearman: General intelligence domainively determined and measured, Am. J. Psychol. 15, 201-293 (2004)

22.42 A. Taleb, C. Jutten: Nonlinear source separation: The post-nonlinear Mixtures, *Proc. ESANN97*, 279-284 (1997)

22.43 H. Tang, KC. Chiu, L. Xu: Finite Mixture of ARMA-GARCH Model For Stock Price Prediction, to appear on *Proc. CIEF'2003*, NC, USA (Sept.26-30, 2003)

22.44 H. Tang, L. Xu: Mixture-Of-Expert ARMA-GARCH Models For Stock Price Prediction, *Proc. of ICCAS 2003*, Oct.22-25, 2003 Gyeongju, KOREA, pp. 402-407 (2003)

22.45 ME. Tipping, CM. Bishop: Mixtures of probabilistic principal component analysis, Neural Computation, 11, 443-482 (1999)

22.46 L. Tong, Y. Inouye, R. Liu: Waveform-preserving blind estimation of multiple independent sources, IEEE Tr on Signal Processing, 41, 2461-2470 (1993)

22.47 VN. Vapnik: *The Nature Of Statistical Learning Theory* (Springer-Verlag) (1995)

22.48 CS. Wong, WK. Li: On a mixture autoregressive model, Journal of the Royal Statistical Society Series B, Vol. 62, No. 1, pp. 95-115 (2000)

22.49 W. Wong, F. Yip, L. Xu: Financial Prediction by Finite Mixture GARCH Model, *Proc. ICONIP'98*, Oct.21-23, 1998, Kitakyushu, Japan, Vol. 3, pp. 1351-1354 (1998)

22.50 L. Xu: Temporal BYY Learning, Identifiable State Spaces, and Space Dimension Determination, *IEEE Trans on Neural Networks*, *Special Issue on Temporal Coding for Neural Information Processing*, in press (2004)

22.51 L. Xu: BYY Learning, Regularized Implementation, and Model Selection on Modular Networks with One Hidden Layer of Binary Units", *Neurocomputing*, Vol. 51, pp. 227-301 (2003)

22.52 L. Xu: Data smoothing regularization, multi-sets-learning, and problem solving strategies, Neural Networks, Vol. 15, No. 56, 817-825 (2003)

22.53 L. Xu: Independent Component Analysis and Extensions with Noise and Time: A Bayesian Ying Yang Learning Perspective, Neural Information Processing - Letters and Reviews, Vol. 1, No. 1, pp. 1-52 (2003)

22.54 L. Xu: BYY Harmony Learning, Structural RPCL, and Topological Self-Organizing on Mixture Models, Neural Networks, Vol. 15, nos. 8-9, 1125-1151 (2002)

22.55 L. Xu: Bayesian Ying Yang Harmony Learning, *The Handbook of Brain Theory and Neural Networks*, Second edition, (MA Arbib, Ed.), Cambridge, MA: The MIT Press, pp. 1231-1237 (2002)

22.56 L. Xu: Mining Dependence Structures from Statistical Learning Perspective. In: H, Yin et al. (eds.), *Proc. IDEAL2002: Lecture Notes in Computer Science, 2412*, Springer-Verlag, 285-306 (2002)

22.57 L. Xu: BYY Harmony Learning, Independent State Space and Generalized APT Financial Analyses, IEEE Tr on Neural Networks, 12(4), 822-849 (2001)

22.58 L. Xu: Best Harmony, Unified RPCL and Automated Model Selection for Unsupervised and Supervised Learning on Gaussian Mixtures, Three-Layer Nets and ME-RBF-SVM Models, *Intl J of Neural Systems* 11 (1), 43-69 (2001)

22.59 L. Xu: Temporal BYY Learning for State Space Approach, Hidden Markov Model and Blind Source Separation", IEEE Tr on Signal Processing 48, 2132-2144 (2000)

22.60 L. Xu: BYY Learning System and Theory for Parameter Estimation, Data Smoothing Based Regularization and Model Selection, Neural, Parallel and Scientific Computations, Vol. 8, pp. 55-82 (2000)

22.61 L. Xu: Temporal Bayesian Ying Yang Dependence Reduction, Blind Source Separation and Principal Independent Components, *Proc. IJCNN'99*, July 10-16, 1999, DC, USA, Vol. 2, pp. 1071-1076 (1999)

22.62 L. Xu: Bayesian Ying Yang Unsupervised and Supervised Learning: Theory and Applications, *Proc. of 1999 Chinese Conf. on Neural Networks and Signal Processing*, pp. 12-29, Shantou, China (Nov. 1999)

22.63 L. Xu: Bayesian Ying Yang Theory for Empirical Learning, Regularization and Model Selection: General Formulation, *Proc. IJCNN'99*, DC, USA, July 10-16, 1999, Vol. 1 of 6, pp. 552-557

22.64 L. Xu: Temporal BYY Learning and Its Applications to Extended Kalman Filtering, Hidden Markov Model, and Sensor-Motor Integration, *Proc. IJCNN'99*, DC, USA, July 10-16, 1999, vol.2 of 6, pp. 949-954

22.65 L. Xu: Bayesian Ying Yang System and Theory as a Unified Statistical Learning Approach :(V) Temporal Modeling for Temporal Perception and Control, *Proc. ICONIP'98*, Kitakyushu, Japan, Vol. 2, pp. 877-884 (1998)

22.66 L. Xu: Bayesian Kullback Ying-Yang Dependence Reduction Theory, Neurocomputing, 22 (1-3), 81-112 (1998)

22.67 L. Xu: RBF Nets, Mixture Experts, and Bayesian Ying Yang Learning, Neurocomputing, Vol. 19, No. 1-3, 223-257 (1998)

22.68 L. Xu: Bayesian Ying Yang Dependence Reduction Theory and Blind Source Separation on Instantaneous Mixture, *Proc. Intl ICSC Workshop I&ANN'98*, Feb.9-10, 1998, Tenerife, Spain, pp. 45-51 (1998)

22.69 L. Xu: Bayesian Ying Yang System and Theory as A Unified Statistical Learning Approach :(VI) Convex Divergence, Convex Entropy and Convex Likelihood? *Proc. IDEAL98*, Oct.14-16, 1998, Hong Kong, pp. 1-12 (1998)

22.70 L. Xu: Bayesian Ying Yang System and Theory as A Unified Statistical Learning Approach: (IV) Further Advances, *Proc. IJCNN98*, May 5-9, 1998, Anchorage, Alaska, Vol. 2, pp. 1275-1270 (1998)

22.71 L. Xu: BKYY Dimension Reduction and Determination, *Proc. IJCNN98*, May 5-9, 1998, Anchorage, Alaska, Vol. 3, pp. 1822-1827 (1998)

22.72 L. Xu, CC. Cheung, SI. Amari: Learned Parametric Mixture Based ICA Algorithm, Neurocomputing, 22 (1-3), 69-80 (1998)

22.73 L. Xu, CC. Cheung, SI. Amari: Further Results on Nonlinearity and Separation Capability of A Linear Mixture ICA Method and Learned Parametric Mixture Algorithm, *Proc. I&ANN'98*, Feb.9-10, 1998, Tenerife, Spain, pp. 39-44 (1998)

22.74 L. Xu: Bayesian Ying Yang System and Theory as A Unified Statistical Learning Approach: (I) Unsupervised and Semi-Unsupervised Learning. In: S. Amari, N. Kassabov (eds.), *Brain-like Computing and Intelligent Information Systems*, Springer-Verlag, pp. 241-274 (1997)

22.75 L. Xu: Bayesian Ying Yang System and Theory as A Unified Statistical Learning Approach (II): From Unsupervised Learning to Supervised Learning and Temporal Modeling. In: KM. Wong et al. (eds.), *Theoretical Aspects of Neural Computation: A Multidisciplinary Perspective*, Springer-Verlag, pp. 25-42 (1997)

22.76 L. Xu: Bayesian Ying Yang System and Theory as A Unified Statistical Learning Approach (III): Models and Algorithms for Dependence Reduction, Data Dimension Reduction, ICA and Supervised Learning. In: KM. Wong et al. (eds.), *Theoretical Aspects of Neural Computation: A Multidisciplinary Perspective*, Springer-Verlag, pp. 43-60 (1997)

22.77 L. Xu: Bayesian Ying Yang Machine, Clustering and Number of Clusters, Pattern Recognition Letters 18, No. 11-13, 1167-1178 (1997)

22.78 L. Xu, CC. Cheung, HH. Yang, SI. Amari: Independent Component Analysis by The Information-Theoretic Approach with Mixture of Density, *Proc. IJCNN97*, Vol. 3, 1821-1826 (1997)

22.79 L. Xu, CC. Cheung, J. Ruan, SI. Amari: Nonlinearity and Separation Capability: Further Justification for the ICA Algorithm with A Learned Mixture of Parametric Densities, *Proc. ESANN97*, Bruges, April 16-18, 1997, pp. 291-296 (1997)

22.80 L. Xu: Bayesian Ying Yang Learning Based ICA Models, *Proc. 1997 IEEE NNSP VII*, Sept.24-26, 1997, Florida, pp. 476-485 (1997)

22.81 L. Xu: New Advances on Bayesian Ying Yang Learning System with Kullback and Non-Kullback Separation Functionals, *Proc. IJCNN97*, June 9-12, 1997, Houston, TX, USA, Vol. 3, pp. 1942-1947 (1997)

22.82 L. Xu: Bayesian-Kullback YING-YANG Learning Scheme: Reviews and New Results, *Proc. ICONIP96*, Vol. 1, 59-67 (1996)

22.83 L. Xu: Bayesian-Kullback YING-YANG Machines for Supervised Learning, *Proc. WCNN96*, Sept.15-18, 1996, San Diego, CA, pp. 193-200 (1996)

22.84 L. Xu: A Maximum Balanced Mapping Certainty Principle for Pattern Recognition and Associative Mapping, *Proc. WCNN96*, Sept. 15-18, 1996, San Diego, CA, pp. 946-949 (1996)

22.85 L. Xu: How Many Clusters?: A YING-YANG Machine Based Theory for A Classical Open Problem in Pattern Recognition, *Proc. IEEE ICNN96*, June 2-6, 1996, DC, Vol. 3, pp. 1546-1551 (1996)
22.86 L. Xu, SI. Amari: A general independent component analysis framework based on Bayesian Kullback Ying Yang Learning, *Proc. ICONIP96*, 1253-1240 (1996)
22.87 L. Xu, HH. Yang, SI. Amari: Signal Source Separation by Mixtures Accumulative Distribution Functions or Mixture of Bell-Shape Density Distribution Functions, Research Proposal, presented at FRONTIER FORUM, organized by S. Amari, S. Tanaka, A. Cichocki, RIKEN, Japan (April 10, 1996)
22.88 L. Xu: Bayesian-Kullback Coupled YING-YANG Machines: Unified Learnings and New Results on Vector Quantization, *Proc. ICONIP95*, Oct 30-Nov.3, 1995, Beijing, China, pp. 977-988 (1995)
22.89 L. Xu: YING-YANG Machine for Temporal Signals, Keynote Talk, *Proc. 1995 IEEE Intl Conf. on Neural Networks and Signal Processing*, Dec. 10-13, 1995, Nanjing, Vol. 1, pp. 644-651 (1995)
22.90 L. Xu: New Advances on The YING-YANG Machine, *Proc. Intl. Symp. on Artificial Neural Networks*, Dec.18-20, 1995, Taiwan, pp. 07-12 (1995)
22.91 L. Xu: A unified learning framework: multisets modeling learning, *Proceedings of 1995 World Congress on Neural Networks*, Vol. 1, pp. 35-42 (1995)
22.92 L. Xu, MI. Jordan, GE. Hinton: An Alternative Model for Mixtures of Experts. In: JD. Cowan et al. (eds.), *Advances in Neural Information Processing Systems 7*, MIT Press, 633-640 (1995)
22.93 L. Xu: Multisets Modeling Learning: An Unified Theory for Supervised and Unsupervised Learning, Invited Talk, *Proc. IEEE ICNN94*, June 26-July 2, 1994, Orlando, Florida, Vol. 1, pp. 315-320 (1994)
22.94 L. Xu: Least mean square error reconstruction for self-organizing neural-nets, Neural Networks 6, 627-648, 1993. Its early version on *Proc. IJCNN91'Singapore*, 2363-2373 (1991)
22.95 L. Xu, E. Oja: Randomized Hough Transform (RHT): Basic Mechanisms, Algorithms and Complexities, *Computer Vision, Graphics, and Image Processing: Image Understanding*, Vol. 57, no.2, pp. 131-154 (1993)
22.96 L. Xu, E. Oja, P. Kultanen: A New Curve Detection Method: Randomized Hough Transform (RHT), Pattern Recognition Letters, 11, 331-338 (1990)