Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 2129–2153
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

A unified perspective and new results on RHT computing, mixture based learning, and multi-learner based problem solving☆

Lei Xu*

Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong

Abstract

On one hand, multiple object detection approaches of Hough transform (HT) type and randomized HT type have been extended into an evidence accumulation featured general framework for problem solving, with five key mechanisms elaborated and several extensions of HT and RHT presented. On the other hand, another framework is proposed to integrate typical multi-learner based approaches for problem solving, particularly on Gaussian mixture based data clustering and local subspace learning, multi-sets mixture based object detection and motion estimation, and multi-agent coordinated problem solving. Typical learning algorithms, especially those that base on rival penalized competitive learning (RPCL) and Bayesian Ying–Yang (BYY) learning, are summarized from a unified perspective with new extensions. Furthermore, the two different frameworks are not only examined with one viewed crossly from a perspective of the other, with new insights and extensions, but also further unified into a general problem solving paradigm that consists of five basic mechanisms in terms of *acquisition*, *allocation*, *amalgamation*, *admission*, *and affirmation*, or shortly *A5 paradigm*.

Keywords: Object detection; Hough transform; Rival penalized competitive learning (RPCL); Elliptic RPCL; Local subspaces; Bayesian Ying–Yang learning; Automatic model selection; Multi-sets modelling; Mixture of experts; RBF nets; Evidence combination

1. Introduction

An intelligent system, which could be an individual or a collection of men, animals, robots, agents, and other intelligent bodies, survives in its world with needs of two types of intelligent abilities.

As illustrated by the right path in Fig. 1(a), Type I consists of abilities of knowing 'what it is' or discovering regularities or dependencies among data as its knowledge about the world. As illustrated in Fig. 1(b), the knowledge is obtained either from pieces of uncertain evidences (or called samples) about the world or from existing authorized sources (e.g., textbooks) that were also obtained from samples in past. Therefore, Type I abilities are obtained via processes what we usually call *learning*, during which an intelligent system gradually senses its world from samples and modifies itself to adapt the world.

As illustrated by the left path in Fig. 1(a), Type II consists of skills of problem solving, i.e., skills of appropriately responding upon what are currently encountering. The reaction can be either just perceiving (e.g., identify, detect, decision, recognize, etc.) or also reacting (e.g., reasoning, control, manage, etc.). As illustrated in Fig. 1(b), the skills of problem solving can be roughly classified into two categories. One is made via evidence combination, inference, optimization, based on a priori knowledge of Type I. The other is developing a fast implementing device (or called problem solver) for the dependence of either the input-perceiving type or the input-reacting type for those often encountered issues that usually need a rapid response. Specifically, the problem solver is developed via learning from samples in help of either a teacher who provides a set of input–response pairs (e.g., in supervised pattern recognition, function approximation, control system, etc.) or the Type I knowledge from learning dependence structures under observations (e.g., in data clustering, object detecting, etc.).

☆ The work described in this paper was fully supported by a grant from the Research Grant Council of the Hong Kong SAR (project No: CUHK4225/04E).
* Tel.: +852 2609 8423; fax: +852 2603 5024.
E-mail address: lxu@cse.cuhk.edu.hk.

0031-3203/$30.00
doi:10.1016/j.patcog.2006.12.016

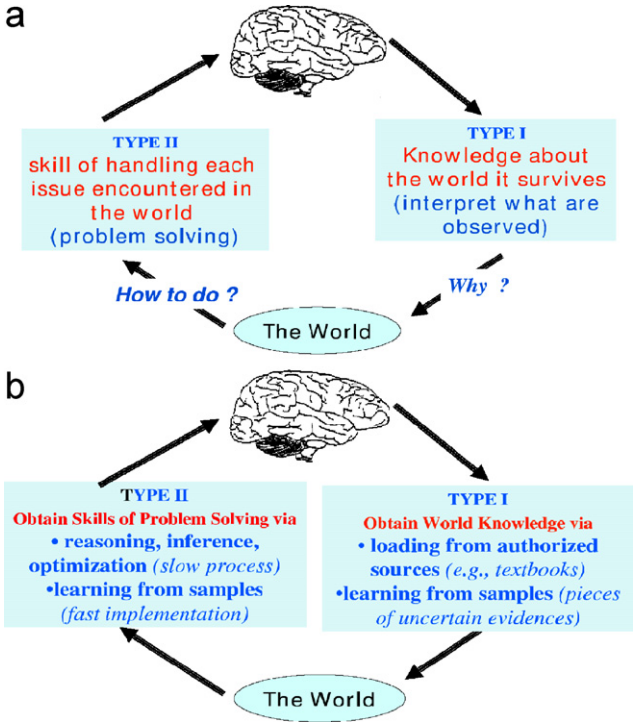

Fig. 1. (a) Two types of intelligent ability. (b) How to get the abilities?

Typical examples of the first category of problem solving skills include those Hough transform (HT) like approaches for detecting objects in an image, started from the early 1960s [1,2]. With a priori Type I knowledge (i.e., each object to be detected is described by a parametric equation), each point in the parameter space describes a potential assumption of one object. The evidence carried by each pixel on an image is accumulated in the parametric space, with those points receiving enough number of votes as the corresponding parametric expressions of the detected objects. However, HT has several shortcomings, and there have been many efforts targeting at solving this or that part of the problems, while the key idea remains more or less the same [1]. Randomized Hough transform (RHT) [3,4] overcomes the problems in help of an innovation on fundamental mechanisms. In the past decade, many studies and applications have been made along this RHT branch with a number of new results [5–13].

Typical examples of the second problem solving category include those learning approaches for estimating a multi-modes featured distribution from samples, mining multiple dependence structures among data, as well as detecting multiple objects and their motions in an image. These approaches are featured by a coordinated learning process by a number of learners or intelligent systems/agents (shortly agents). We start at introducing rival penalized competitive learning (RPCL) for clustering analysis and line detection in an image [14,15]. Again, each point in a parametric space describes a potential assumption of one object. A number of learners search among the parameter space, with each describing a candidate of a detected object. As one sample comes, it is allocated to one candidate via competition, and the winner moves a little bit to adapt the information carried by this sample. Moreover, the rival (i.e., the second winner) is repelled a little bit from the sample to reduce a duplicated information allocation. Such a rival penalized mechanism can detect an appropriate number of candidates. Not only RPCL has been further extended from spheral clusters to elliptic clusters via Gaussian mixture [16–18], but also multi-sets mixture learning (MML) has been developed [19,20] for detecting objects ranging from lines and subspaces to ellipses, and even any shapes via given templates. Readers are referred to Ref. [21] for a recent elaboration and to [22,23] for successful applications. Also, many studies and applications have been made on RPCL learning in the past decade [24–33].

This paper aims at jointly investigating both the above two categories. Basic mechanisms behind each category have been elaborated and a general framework is proposed for each category to summarize variants and to explore extensions. Not only each framework is viewed crossly from a perspective of the other, but also both are unified into a general problem solving paradigm that consists of five mechanisms, namely *acquisition*, *allocation*, *amalgamation*, *admission*, *and affirmation*. In Section 2, after introducing HT and RHT, a general evidence accumulation framework for problem solving is presented for object detection, with new variants and extensions of RHT discussed. In Section 3, after introducing the fundamentals of RPCL learning, another general framework is proposed to integrate multi-learner based approaches for problem solving, with typical learning algorithms (especially RPCL and Bayesian Ying–Yang (BYY) learning based ones), summarized from a unified perspective with new extensions. In Section 4, each framework is examined crossly from a perspective of the other, and both are unified as a specific implementation of a general problem solving paradigm. Concluding remarks will be made in Section 5.

2. Evidence accumulation approaches: RHT and extensions

2.1. A brief introduction of HT and RHT

As shown in Fig. 2(a), we take straight line detection as an example to introduce the key idea of HT. A pixel (x, y) is mapped into a line $b = xk + y$ passing a point (k, b) in a parameter space where each point represents a potential assumption of a detected line. That is, one pixel from the image allocates its partial evidence to a set of potential assumptions located on the line $b = xk + y$. In other words, all these assumptions are activated by this piece of evidence. Moreover, a set of points on the line $y = kx + b$ in the image are mapped into a set of lines across a point (k, b) in the parameter space. As a result, the activated assumption at the point (k, b) receives an amount of evidences much higher than other assumptions and thus detects a line $y = kx + b$ in the image. To implement such an idea, a grid with a uniform quantization is located on a window in the (k, b) space, and a score accumulator $a(k, b)$ is located at each bin that represents a potential assumption. As each point (x, y) on the image is mapped into a line in the (k, b) space, every

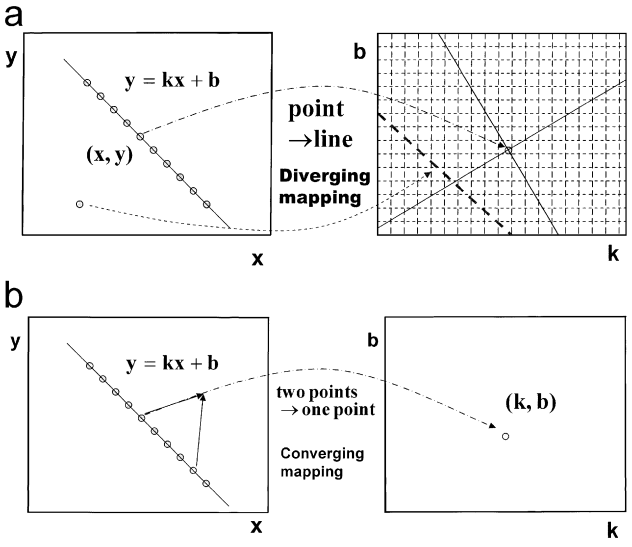

Fig. 2. From Hough transform to randomized Hough transform.

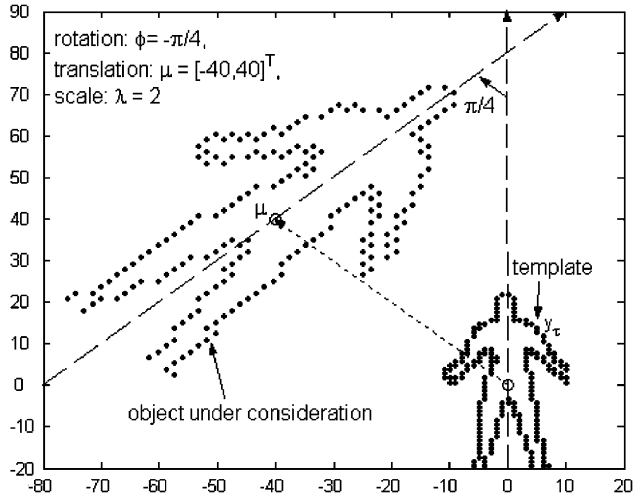

Fig. 3. Template matching.

accumulator $a(k, b)$ with the line passing through is added by 1. We can detect lines by finding out each peak score of $a(k, b)$. The same idea also applies to the detection of circles, ellipses, as well as a more general shape that can be represented by a parametric equation [1].

However, HT has several critical drawbacks as follows:

(a) All pixels are mapped, and every bin in the grid needs an accumulator. If there are d parameters with each represented by M quantizing units, there need accumulators of M^d.
(b) To reduce the computing cost, quantization resolution cannot be high, which blurs the peaks and leads to a low detection accuracy.
(c) Each pixel activates every accumulator located on a line, with only one representing the correct one while all the others are disturbances.
(d) If the grid window is set inappropriately, some objects may locate outside the window and thus is not able to be detected.
(e) Noisy pixels cause many disturbing accumulations.

As shown in Fig. 2(a), HT maps one pixel (x, y) into all the points on a line passing (k, b). It is this diverging mapping mechanism that actually incurs the above drawbacks (a)–(e). RHT [3,4] replaces this mechanism with a converging mapping mechanism as shown in Fig. 2(b). That is, two or more pixels are picked to jointly determine a line, i.e., mapped into one point (k, b). By this mechanism, different points on a same line $y = kx + b$ will hit the same point (k, b), without creating a great number of false accumulations. Also, the feature of being mapped into one point per time makes it is possible to lay accumulators dynamically without the need of laying a grid on a pre-specified window. We only need to accumulate scores $a(k, b)$ at those locations activated by the converging mappings. Also, not only quantization resolution may vary for locations, and each quantization bin can be replaced by a kernel. As a result, the drawbacks (b)–(d) no longer exist.

Furthermore, there is no need to enumerate all the pixels. For each converging mapping, we can randomly pick two or more pixels with each in an equal probability. When there are many pixels on a line $y = kx + b$, the probability that a point (k, b) is hit by converging mappings from samples on the line will be much higher than the probability that other points in the (k, b) space are hit by the converging mappings from samples not on lines. Therefore, a line can be detected via a point in the (k, b) space if it is hit by certain amount of times. In other words, the drawbacks (a), (e) have also been avoided. In the past decade, studies on RHT has become an important branch of HT studies [5–13].

RHT is directly applicable to the detection of circles, ellipses, as well as other more general shapes in a parametric equation $f(x, y, \theta) = 0$ with a number κ of free parameters. Solving the following joint equations yields a converging mapping into a unique point θ in the parameter space [4]:

$$f(x_i, y_i, \theta) = 0, \quad i = 1, \ldots, \kappa. \tag{1}$$

This mechanism can also be extended to the cases that objects are represented by templates as shown in Fig. 3. A template can be used to match a shape via a translation μ, a rotation ϕ and a scaling λ. That is, each pixel $u = (x, y)$ on the image relates to a corresponding point v on the template via

$$u = \lambda R(\phi)(v - \mu), \tag{2}$$

where $R(\phi)$ is a rotation matrix by an angle ϕ. If we are able to get $m \geqslant 2$ pairs of the (u, v) correspondence, the parameters λ, ϕ, c can be solved jointly by m matrix equations obtained with Eq. (2) for each pair. In other words, randomly sampling m pairs of (u, v) with u from the image and v from the template, we get a converging mapping to one point in the parameter space of λ, ϕ, μ. From an alternative view, we can also detect a motion of an object by the displacement μ and the rotation ϕ between two times by setting $\lambda = 1$.

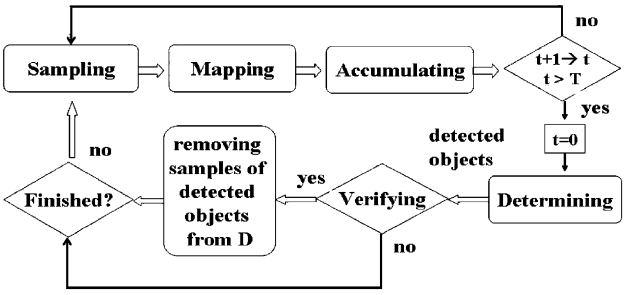

Fig. 4. Evidence accumulation framework.

2.2. Evidence accumulation framework and five basic mechanisms

2.2.1. Evidence accumulation framework

Both the processes of HT and RHT are featured by gathering evidences from a data set **D** from either an image after certain pre-processing or an other information source, and accumulating evidences piece by piece in a parametric space, such that certain points in the parametric space are detected as objects if each of the points receives an enough number of votes.

In the implementation of HT as shown in Fig. 2(a), not the entire parameter space but the points within each bin are actually considered. Also, all the points within each bin is treated as a same potential assumption of an object to detect. A potential assumption is activated when a line $b = xk + y$ passes through the corresponding bin. An assumption becomes active once it is activated. Each active assumption is attached with an accumulator that updates its supporting evidence every time it is activated. For a pixel (x, y), the assumptions with their corresponding bins on the line $b = xk + y$ are all activated. In the implementation of RHT as shown in Fig. 2(b), every point in the entire parameter space is implicitly regarded as a potential assumption that becomes active once it is hit by a converging mapping. A converging mapping activates merely one assumption.

Actually, both a HT process and a RHT process can be regarded as specific implementations of a general evidence accumulation framework as shown in Fig. 4, featured by five basic mechanisms, namely *sampling* for acquiring information, *mapping* for activating assumptions, *accumulating* evidences to intensify the activated assumptions, then *determining* and *verifying* candidates. A problem solving process consists of a series of epochs, with each epoch featured by one run of consecutively implementing the first three mechanisms. The last two mechanisms may be implemented per $T \geqslant 1$ epochs. The entire process stops either after all the samples have been enumerated (e.g., for HT) or when terminating signal is received externally or via a monitoring measure that says "there is no more object to detect among the remaining part of **D**".

In the sequel, we further elaborate the five mechanisms in Table 1 with more details. Each of the five basic mechanisms has several choices to implement. The differences between HT and RHT arise from an integrated implementation of the first three mechanisms.

2.2.2. Sampling and mapping

As shown in Table 1, each epoch starts at *Sampling* that picks one or several samples from **D**, in one of the following ways:

(a) As used in HT and variants, one is simply picking one sample per epoch by scanning all the samples among **D**, with each sample picked only once. Moreover, one may also pick k pixels per epoch, but the number of possible k-pixel-combinations to be scanned increase explosively as k increases.
(b) As used in RHT [3,4], at each epoch, one or several samples are randomly picked with each in an equal probability. There is no need on exhaustively enumerating all the samples or their combinations.
(c) The third way is picking one or several samples in help of currently available knowledge, e.g., locally around certain region of an image by connectivity or along certain directions, some examples are discussed in Ref. [34].

The choice on a way for picking samples is not completely independent from the choices of the subsequent mechanisms. E.g., when we choose to scan all the samples, we have to let the threshold T being the total number of samples in **D**. Though such a coupling can be removed if we randomly pick samples, the number m of picked samples still affect the choices of the subsequent mechanisms, as listed in the second column of Table 1. For detecting an object that has to be determined by at least κ points (e.g., $\kappa = 2$ for detecting a line), we encounter the following three scenarios:

(a) When $m < \kappa$, we are unable to uniquely activate an assumption. Instead, we can only explicitly or implicitly specify a manifold in the parameter space. For detecting a line by HT, we have $m = 1 < \kappa = 2$ and the manifold is a line. For the cases with $\kappa > 2$, there will be $\kappa - 2$ different manifolds that lead to different variants of HT.
(b) When $m = \kappa$, except those degenerated cases that become actually the above situation (a), the joint equations by Eq. (1) or (2) will activate a unique candidate. E.g., in the implementation of RHT, we have $2 = m = \kappa = 2$ that yields a converging mapping.
(c) When $m > \kappa$, the situation would have no difference from the above (b) if there is no noise on samples. However, there are always noises and quantization errors such that the joint equations have no solution. As suggested in Ref. [4] for detecting curves on an image, we may use the median axis of scatted pixels as an estimation, as shown in Fig. 5. Also, we may even simply check whether the average error δ is smaller than a given threshold. This approach works well for a parametric expression with a small κ, e.g., line, circle, ellipse, etc. In general, we consider the following double minimization approach.

Considering that samples from each object include one deterministic part plus noises. The deterministic part is described by a finite or continuous set $S(\theta)$ of real points in R^d, subject to a parametric set θ with a number of unknown parameters.

Table 1
Different choices for five essential mechanisms

Sampling	Mapping	Accumulating	Determining	Verifying
(a) Enumerate	(a) and (b) $m \leqslant \kappa$:	(a) Uniform	(a) Large ones	(a) Length
(b) Random	solving equations	(b) Nonuniform	(b) Peaks	(b) Error
sampling	$f(x, y, \theta) = 0$	Over bins	(c) Clusters	L_2
(c) Local	(c) $m > \kappa$	Over epochs	*Spherical*	L_1
searching	median or	(c) Kernel based	*Others*	log-L
	L_p *fitting*	(d) Discard dead		(c) Test

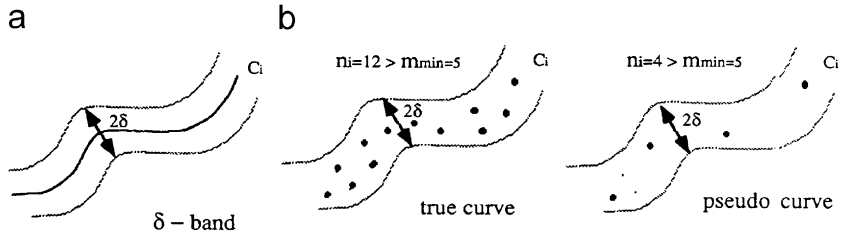

Fig. 5. Curves under noises and disturbances. (a) A δ band of curve c_i is a band of width 2δ with c_i as the median axis. (b) A true curve c_i has at $m_{\min}$ pixels falling in a given δ band; otherwise c_i is a pseudo-curve.

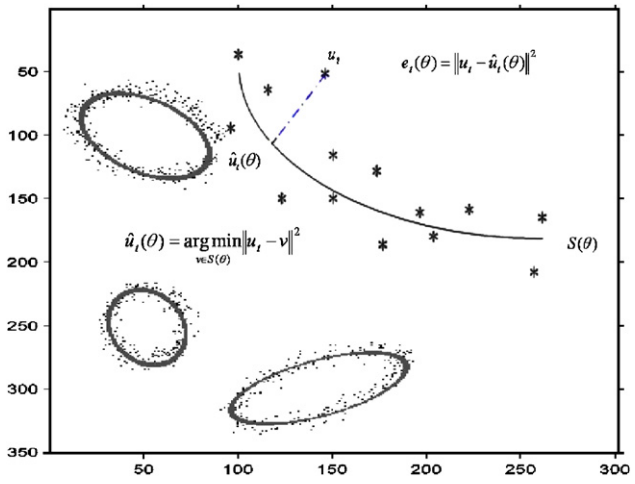

Fig. 6. Least-square error fitting.

$S(\theta)$ represents a shape such as line, curve, and ellipsis, as well as a pre-specified shape. For an expression $f(u, \theta) = 0, u = (x, y)$, we have

$$S(\theta) = \{u : u \text{ satisfies } f(u, \theta) = 0\}. \tag{3}$$

For an object in a shape without a parametric expression, as shown in Fig. 6, a template is given via a set of samples $Y = \{y_\tau\}_{\tau=1}^N$ that represents a contour of a specific shape, the following $S(\theta_j)$ is used to denote a shape resulted from a displacement μ, a rotation matrix $R(\phi_j)$ for an angle ϕ_j, and a scaling λ_j

$$S(\theta_j) = \{\lambda;\, R(\phi_j)(y_\tau + \mu_j) : \forall y_\tau \in Y\}$$
$$\text{where } \theta_j = \{\mu_j, \phi_j, \lambda_j\}. \tag{4}$$

As shown in Fig. 6, we define the error that a point u_t deviates from an object represented by $S(\theta)$ via the following minimization:

$$e(u_t, \theta) = u_t - u_t(\theta), \quad u_t(\theta) = \min_{v \in S(\theta)} \|u_t - v\|_p, \tag{5}$$

where $u_t(\theta)$ is a reconstruction or representation of u_t by $S(\theta)$. Then, we solve the second minimization $\min_\theta \sum_{t=1}^m \|e(u_t, \theta)\|_p$, $m \geqslant \kappa$ via finding one or several local minimum values of θ for activating assumptions in the parameter space, where $\|\xi\|_p$ denote the L_p norm of the vector ξ, e.g., L_1 norm for $p = 1$ and L_2 norm for $p = 2$. Though a larger m may produce a better performance in removing the noises if the samples all come from a same object, we need to check the samples in a way similar to that in Fig. 5.

2.2.3. Accumulating and determining

For each assumption, its corresponding supporting evidence is accumulated per activating, according to one of the following rules (summarized in the third column of Table 1):

(a) The simplest one is voting, i.e., each assumption is added with 1 per activation, uniformly over all the activated assumptions and all the epochs.
(b) As shown in Fig. 2(b), each bin is placed with one accumulator. Due to quantization, each bin is voted by an amount l_b proportional to the segment length of a line $b = xk + y$ falling within each bin.
(c) With $m > \kappa$ samples, our confidence on activated assumptions vary as epoch t varies, e.g., in Fig. 5 it can be measured by either or both of the average error δ_t and the fraction f_t of samples falling within the δ band. Correspondingly, the activated assumptions should be accumulated with a piece of evidence proportional to $f_t l_b/\delta_t$.

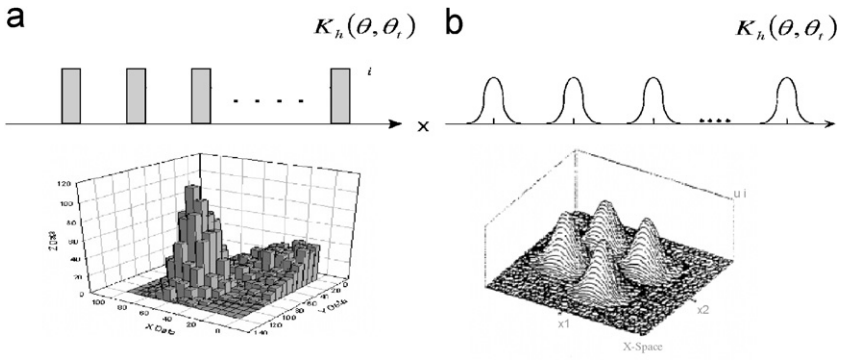

Fig. 7. Kernel supported RHT.

(d) Instead of voting to a cubic bin as in Fig. 7(a), we can place a kernel function with its center located at each activated assumption, as shown in Fig. 7(b). That is, assumptions in its neighbor area are accumulated by certain degree too, which motivates a new direction—kernel supported RHT.
(e) When an active assumption has not been activated for a long period τ, we may regard it as dead and remove its corresponding accumulator, such that the cost for space can be saved and the performance may be improved. This τ may relate to its accumulated amount of evidences.

After implementing accumulation, as shown in Fig. 4, it returns to the sampling mechanism unless it has finished T epochs. As in the fourth column of Table 1, the assumptions accumulated after T epochs are then assessed to determine on which of them can be admitted as candidate solutions, according to

- whether the amount of its accumulated evidence is larger than a threshold;
- whether the amount of its accumulated evidence becomes stably a peak;
- whether some active assumptions have steadily formed a cluster either in spherical shapes or in other configurations, such that the cluster centers or modes are picked as candidates.

The obtained candidates can be either directly taken as solutions or further verified to check whether each of them can be determined as a detected object. This verifying procedure consists of two steps. First, we search within a neighbor region of the candidate to identify samples that belong to the object. The simplest way is to take the samples within a δ band in Fig. 5. We can also make a maximum likelihood (ML) detection based on a model. Second, we test whether it can be determined according to one of the following criteria:

- Enough number of samples, e.g., enough length of a line segment.
- An average error of these samples in fitting the object, or a log-likelihood of the samples from the object under a probabilistic model.
- A statistical testing based on some testing-ratio, e.g., Bayes factor [35].

3. Multi-learner based problem solving approaches

The previous category of problem solving approaches is featured by the first choice on the left path in Fig. 1(b). Given the objects' structures, e.g., a line, an ellipse, a parametric equation by Eq. (1), and a template as shown in Fig. 3, the Type I knowledge (i.e., the locations, orientations, and scales of the structures) are obtained via evidence accumulation and searching optimums in the parameter space, together with pixels classified into different objects.

Another category of problem solving approaches is featured by the second choice on the left path in Fig. 1(b), i.e., the Type I knowledge are learned via adapting information among samples by multiple learning agents (or shortly learners). There are two coupled tasks. First, each learner fights to be allocated to an environment via competition and communication mutually among learners. Second, each learner learns the knowledge about its environment, and perhaps also make certain action according to its knowledge. Both the tasks are also affected by interactions among learners. In the sequel, a general framework is proposed to integrate typical multi-learner based problem solving approaches from a unified perspective with several new extensions.

3.1. RPCL for clustering analysis

We start at the simplest scenario that each learner θ_j is featured by a point $\theta_j = \mu_j$ that moves among the observation samples and seeks to be located at the center of a cluster of samples. This center may represent an object to be detected among noisy observations. Jointly, several learners collectively perform data clustering or detecting multiple objects. One earliest effort is the classical competitive learning (CCL) [36] which is implemented adaptively per sample. As a sample x_t comes, each learner μ_j has an individual value or criterion $\varepsilon_t(\theta_j) = \|x_t - m_j\|^2$ as a degree of dissatisfaction on its representation of x_t, and competes to represent x_t such that $\varepsilon_t(\theta_j)$ is reduced or minimized. The competition is guided globally by

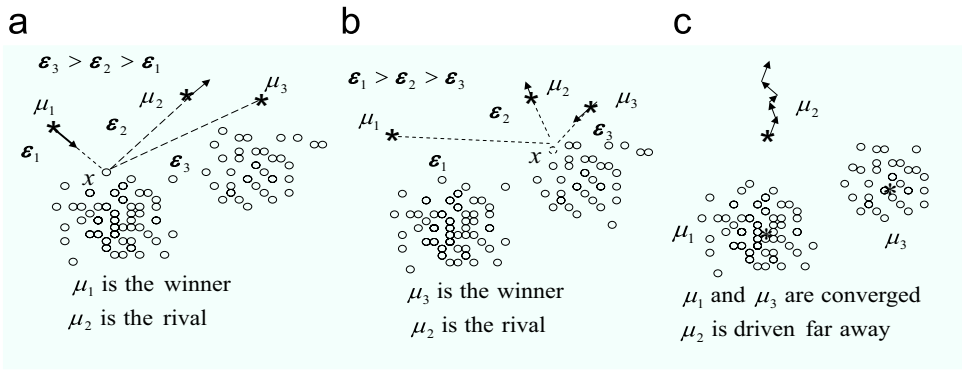

Fig. 8. Rival penalized competitive learning.

the winner-take-all (WTA) policy

$$p_{j,t} = \begin{cases} 1 & \text{if } j = c, \\ 0 & \text{otherwise,} \end{cases} \qquad c = \arg\min_j \varepsilon_t(\theta_j). \tag{6}$$

Each learner updates to reduce $\varepsilon_t(\theta_j)$ by

$$\mu_j^{new} = \mu_j^{old} + \eta p_{j,t}(x_t - \mu_j^{old}). \tag{7}$$

It is well known that the CCL has a so-called *under-utilized* or *dead unit* problem, i.e., some learners will become 'dead', while each of other learners may take samples crossly from more than one clusters, which makes the task of clustering analysis badly performed. This problem comes from that the WTA or hogging mechanism is too strong. There needs a sharing mechanism to balancing such a monopolizing tendency. One sharing mechanism is the so-called frequency sensitive competitive learning (FSCL) [37], by simply modifying $\varepsilon_t(\theta_j) = \|x_t - \mu_j\|^2$ into

$$\varepsilon_t(\theta_j) = \alpha_j \|x_t - \mu_j\|^2, \tag{8}$$

where α_j is the frequency that the jth learner won in past. Such a spirit of reducing the winning rates of frequent winners can also be implemented in alternative ways, e.g., *conscience* [38]. Moreover, a sharing mechanism can be introduced in different ways too, e.g., *leaky learning* [36,39], *convex bridge* [40], and neighbor sharing in Kohonen Map [41].

The sharing mechanism by α_j in Eq. (8) works well when k is pre-assigned to an appropriate one. However, when k is pre-assigned to a value larger than an appropriate one, FSCL fails due to its sharing mechanism is excessive, which makes the extra units also move into data to disturb the correct locations of other units. This is another critical difficulty that we need to overcome. One solution is given under the name of RPCL, proposed in the early 1990s for clustering analysis and detecting lines in an image [14,15].

The key idea is shown in Fig. 8, i.e., not only the winning learner moves a little bit to adapt x_t, but also the rival (i.e., the second winner) is repelled a little bit from x_t to reduce a duplicated information allocation. For example, as a sample x_t comes, μ_1 is the winner and μ_2 is its rival in Fig. 8(a), μ_1 is moved towards x_t by a small step size, while μ_2 is repelled away from x_t by a much smaller step size. Similarly, in Fig. 8(b), μ_3 is the winner and moved towards x_t, while μ_2 is its rival and repelled away from x_t. Formally, this is implemented by Eq. (7) with Eq. (6) replaced by

$$p_{j,t} = \begin{cases} 1 & \text{if } j = c, \\ -\gamma & \text{if } j = r, \\ 0 & \text{otherwise,} \end{cases} \qquad \begin{cases} c = \arg\min_j \varepsilon_t(\theta_j), \\ r = \arg\min_{j \neq c} \varepsilon_t(\theta_j), \end{cases} \tag{9}$$

where $\varepsilon_t(\theta_j)$ is given by Eq. (8), and γ approximately takes a number between 0.05 and 0.1 for controlling the penalizing strength versus learning strength.

As illustrated in Fig. 8(c), μ_1 and μ_3 finally converge to clusters' centers, while μ_2 is driven far away from samples. That is, the rival penalized mechanism has balanced the excessive sharing mechanism by α_j in Eq. (8) and thus driven extra learners far away from data. In other words, RPCL determines the number of clusters or objects automatically during learning. This is a favorable feature that the conventional competitive learning or clustering algorithm (e.g., k-means) does not have. In the past decade, a number of applications of RPCL have been made [24–29,31–33].

In 1998 [16,42], RPCL based clustering analysis is extended to elliptic clusters with each cluster by a Gaussian in general, with certain simplification on updating the rival. The updating equations without simplification are given in Ref. [17, Tables 1(B) and 2(A); 43, Eq. (33)]. Again, the number of Gaussians can be determined automatically during RPCL learning. Moreover, the MML has been developed [19,20] for detecting objects ranging from lines and subspaces to ellipses, and even any shapes via given templates. Also, it has been demonstrated that RPCL can be regarded as one fast approximate implementation of the BYY harmony learning that was firstly proposed in 1995 [44] and developed in the past decade [45–48], and thus the RPCL's ability of automatic model selection can be understood in a top-down way from the BYY harmony learning theory. Different implementations of the BYY harmony learning lead to different versions of RPCL-like mechanism. A historic remark on this topic is referred to Section 23.7 in Ref. [48].

3.2. A general multi-learner based problem solving framework

3.2.1. Three fundamental ingredients and automatic model selection

The RPCL learning can be regarded as a special case of a general multi-learner based problem solving framework that integrates several related learning algorithms from a unified perspective, featured by three ingredients:

3.2.1.1. Leaner structure. Instead of being represented simply by a point μ_j, each learner needs a hardware S_j that is able to handle typical problem solving tasks. This S_j can be one of two choices:

- *Without external action*: Each learner targets only on *representing* the knowledge about its environment, in help of a pre-specified structure S^r and a set θ^r_j of parameters within S^r_j. For object detection, S^r_j may be given either by Eq. (3) for a shape such as line, curve, and ellipses or by Eq. (4) for a shape in a template. Beyond object detection, S^r_j may also be an appropriate parametric model for representing a particular structure underlying samples. Moreover, some S^r further consists of a number m^r_j of components, e.g., a subspace of dimension m^r_j in Fig. 11(a) later in Section 3.3.2. Usually, we use θ^r_j, plus m^r_j to denote this structure.
- *With external action*: In certain tasks, upon an input x_t we also need an additional structure S^a_j for making an *action* to its environment, via a computable function or operator

$$\zeta_t = \zeta^a(x_t|\theta^\zeta_j), \tag{10}$$

where ζ_t is either a real number or a discrete label. Similarly, we can use θ^a_j, plus m^a_j (if any), to denote this structure S^a_j.

In the above formulation, the tasks of problem solving are embedded in the process of determining or learning all the unknowns of every learner. The unknowns always contain $\{\theta^r_j\}$ that implicitly includes the unknowns $\{m^r_j\}$ (if any). For the tasks that involves making actions, the unknowns also contain $\{\theta^a_j\}$ and $\{m^a_j\}$ (if any). There may also be some additional unknowns for combining the two parts S^r_j and S^a_j. We simply use θ_j to denote all these unknowns, which are determined jointly by the learners' individual criteria and the policy for a global coordination.

3.2.1.2. Individual criterion. As an extension of the previous $\varepsilon_t(\theta_j) = \|x_t - \mu_j\|^2$, each learner has an individual value $\varepsilon_t(\theta_j)$ to evaluate its performance per sample x_t observed. This performance may involve two types of contributions. One is how good x_t is represented by S^r_j. As discussed in Eqs. (3)–(5), we can get $x_t(\theta^r_j)$ as a reconstruction or a representation of x_t by S^r_j and thus an error

$$e^r_t(\theta^r_j) = x_t - x_t(\theta^r_j), \tag{11}$$

from which we get $\varepsilon_t(\theta^r_j)$ as this part of contribution, where $\varepsilon(e^r_t(\theta^r_j)) \geqslant 0$ and $\varepsilon_t(\theta^r_j) = 0$ iff (if and only if) $e^r_t(\theta^r_j) = 0$. The most common one is

$$\varepsilon(e^r_t(\theta^r_j)) = \|e^r_t(\theta^r_j)\|^2. \tag{12}$$

Other typical examples are also given by Eqs. (32)–(34) in Ref. [21].

In the situation that the learner needs to make an action by Eq. (10), we need another contribution $\varepsilon_t(\theta^a_j)$ to evaluate the performance of the action. There are two typical scenarios:

- *Supervised learning*: There is a teacher to give a set of training pair $\{x_t, \zeta_t\}$, e.g., for the tasks of classification, function regression, prediction, etc. It follows from Eq. (10) that we can simply have

$$e^a_t(\theta^a_j) = \zeta_t - \zeta^a(x_t|\theta^\zeta_j) \tag{13}$$

from which we get $\varepsilon_t(\theta^a_j) = \varepsilon(e^a_t(\theta^a_j)) \geqslant 0$ and $\varepsilon_t(\theta^a_j) = 0$ iff $e^a_t(\theta^a_j) = 0$. Again, one example is the square error

$$\varepsilon(e^a_t(\theta^a_j)) = \|e^a_t(\theta^a_j)\|^2. \tag{14}$$

There could be various specific forms for each individual criterion $\varepsilon_t(\theta_j)$. Every one expects a good performance but possibly in a different sense. The square error equations (12) and (14) target at a best performance on a currently available set of samples. There are a number of other criteria that seeks a best performance in this sense. However, due to noise and not an enough number of samples, a best performance on a currently available set from a underlying structure may not generalize well (i.e., it my not still be the best as more samples come from the same underlying structure). Thus, we may also consider $\varepsilon_t(\theta_j)$ that aims at a best generalization sense.

- *Reinforcement learning*: There is no teacher. An action is made upon x_t, with an encouraging or discouraging score incurred from environment, which is simulated by Eq. (10) that provides a score $\zeta^a_t \geqslant 0$ that expresses a degree of dissatisfactory on the action, with $\zeta^a_t = 0$ for a complete satisfactory. In this case, we let $\varepsilon_t(\theta^a_j) = \zeta^a_t$ directly.

In a summary, we have

$$\varepsilon_t(\theta^a_j) = \begin{cases} \varepsilon(e_t(\theta^a_j)) & \text{(a) supervised learning,} \\ \zeta^a_t & \text{(b) reinforcement learning.} \end{cases} \tag{15}$$

As a whole, the two parts are compounded together via an operator:

$$\varepsilon_t(\theta_j) = \Omega[\varepsilon_t(\theta^r_j), \varepsilon_t(\theta^a_j), \mathbf{b_j}], \tag{16}$$

where $\mathbf{b_j}$ consists of some compounding parameters. $\varepsilon_t(\theta_j)$ is monotonically increasing with respect to both $\varepsilon_t(\theta^r_j)$ and $\varepsilon_t(\theta^a_j)$. One example is

$$\frac{\varepsilon_t(\theta_j)}{\beta_j} = \begin{cases} \dfrac{\varepsilon_t(\theta^r_j)}{\beta^r_j} & \text{(a) without making action,} \\ \dfrac{\varepsilon_t(\theta^r_j)}{\beta^r_j} + \dfrac{\varepsilon_t(\theta^a_j)}{\beta^a_j} & \text{(b) with an action made.} \end{cases} \tag{17}$$

where $\mathbf{b_j} = \{\beta_j, \beta_j^r, \beta_j^a\}$ that make $\varepsilon_t(\theta_j)$, $\varepsilon_t(\theta_j^r)$, and $\varepsilon_t(\theta_j^a)$ become scale invariant. One typical example is

$$\beta_j = \frac{1}{N}\sum_{t=1}^{N}\varepsilon_t(\theta_j), \quad \beta_j^r = \frac{1}{N}\sum_{t=1}^{N}\varepsilon_t(\theta_j^r),$$
$$\beta_j^a = \frac{1}{N}\sum_{t=1}^{N}\varepsilon_t(\theta_j^a). \tag{18}$$

3.2.1.3. Combining policy. Extending Eq. (6), learners are coordinated via a global policy that combines individual criteria $\{\varepsilon_t(\theta_j)\}_{j=1}^k$. The global policy can be either a criterion for evaluating the overall performance jointly by all the learners, e.g., a simple one is the following weighted sum:

$$\varepsilon(\theta) = \sum_{t=1}^{N}\varepsilon_t(\theta), \quad \varepsilon_t(\theta) = \sum_{j=1}^{k} p_{j,t}\frac{\varepsilon_t(\theta_j)}{\beta_j} \tag{19}$$

or just an allocation scheme on a per sample basis

$$p_{j,t} = \mathrm{ALOC}\left[\left\{\frac{\varepsilon_t(\theta_j)}{\beta_j}\right\}_{j=1}^{k}\right] \tag{20}$$

according to which x_t is assigned to the jth learner with a weight $p_{j,t}$. $p_{j,t} > 0$ means that x_t is at least partially allocated to the jth learner. Three typical examples are given as follows:

- *WTA allocation*: The simplest and widely used one is given by Eq. (6) at $\beta_j = 1$, i.e., x_t is completely allocated to the winner c.
- *Shared allocation*: A group of learners with $p_{j,t} > 0$ provides a sharing mechanism, one example is the following soft-allocation:

$$p_{j,t} = \begin{cases} \dfrac{e^{-\varepsilon_t(\theta_j)/\beta_j}}{\sum_{i\in\mathscr{C}_\kappa} e^{-\varepsilon_t(\theta_i)/\beta_i}} & \text{if } j \in \mathscr{C}_\kappa, \\ 0 & \text{otherwise,} \end{cases}$$

$$\mathscr{C}_\kappa = \left\{ j : \varepsilon_t(\theta_j)/\beta_j \text{ among the } \kappa \text{ smallest ones of } \left\{\frac{\varepsilon_t(\theta_j)}{\beta_j}\right\}_{j=1}^{k} \right\}. \tag{21}$$

One extreme case is $\kappa = 1$ by which Eq. (21) degenerates to the WTA equation (6) at $\beta_j = 1$. The other extreme case is $\kappa = k$ by which Eq. (21) leads to the Bayes posterior allocation by Eq. (27) as $\varepsilon_t(\theta_j)/\beta_j$ given by Eq. (26).

Another example can be found in Kohonen Map [41], where certain neighbors of the winner are all assigned with $p_{j,t} > 0$.

- *Penalized allocation*: We can also have a case with some $p_{j,t} < 0$ means that the jth learner will be repelled from x_t by a little bit. The one by Eq. (9) for RPCL is such an example.

We can regard $p_{j,t}$ by Eq. (9) as a sum of two parts, namely $q_{j,t}$ by Eq. (6) minus $\gamma\pi_{j,t}$ with $\pi_{j,t} = 1$ at $j = r$ and $\pi_{j,t} = 0$ for $j \neq r$. Generally, we can let

$$p_{j,t} = q_{j,t} - \gamma\pi_{j,t}, \quad q_{j,t} \geqslant 0, \quad \sum_j q_{j,t} = 1 \quad \text{and}$$
$$\pi_{j,t} \geqslant 0, \quad \sum_j \pi_{j,t} = 1, \tag{22}$$

where $q_{j,t}$ provides a sharing mechanism, while $\pi_{j,t}$ provides a penalizing mechanism that distributes a penalizing strength γ over learners. One extreme case is Eq. (9) with $q_{j,t}$ by Eq. (6) and $\pi_{j,t} = 1$ at $j = r$ and $\pi_{j,t} = 0$ for $j \neq r$. One another extreme case $\pi_{j,t} = 1/k$.

3.2.2. Probabilistic approach, individual criterion, and combining policy

Each learner may also be given a probabilistic structure

$$p(u_t|\theta_j) = \begin{cases} p(x_t|\theta_j^r) & \text{(a) } u_t = x_t \text{ (no external action),} \\ p(x_t|\theta_j^r)\,p(\zeta_t|x_t, \theta_j^a) & \text{(b) } u_t = \{x_t, \zeta_t\} \text{ (with an action),} \end{cases} \tag{23}$$

where $p(\zeta_t|x_t, \theta_j^a)$ describes a probabilistic response ζ_t incurred from x_t. For the reinforcement learning case in Eq. (15), a randomness of ζ_t may come from either a randomness in making an action or a randomness in scoring an action.

Each learner may also take the proportion α_j and a priori knowledge on θ_j in consideration, such that Eq. (23) can be further modified into

$$\text{(a) } p(u_t, j) = \alpha_j p(u_t|\theta_j),$$
$$\text{(b) } p(u_t, \theta_j) = p(u_t|\theta_j)p(\theta_j),$$
$$\text{(c) } p(u_t, j, \theta_j) = \alpha_j p(u_t|\theta_j)p(\theta_j). \tag{24}$$

Moreover, a parametric model in Eq. (23) actually specifies not only a learner structure but also indirectly an individual criterion $\varepsilon_t(\theta_j)$ by a monotonic decreasing scalar function $\varrho(p)$. Taking $\varrho(p) = -\ln p$ as an example, we have

$$\frac{\varepsilon_t(\theta_j)}{\beta_j} = -\ln p_j,$$
$$p_j = \begin{cases} p(u_t|\theta_j), \\ p(u_t, j), \\ p(u_t, \theta_j), \\ p(u_t, j, \theta_j), \end{cases} \qquad \alpha_j \geqslant 0, \ \sum_j \alpha_j = 1. \tag{25}$$

Taking the case with $p(u_t, j)$ as an example, we have

$$\varepsilon_t(\theta_j)/\beta_j = -[\ln p(u_t|\theta_j) + \ln\alpha_j], \tag{26}$$

and put it into Eq. (21), we further get

$$p_{j,t} = \begin{cases} \dfrac{\alpha_j\, p(u_t|\theta_j)}{\sum_{i\in\mathscr{C}_\kappa} \alpha_i\, p(u_t|\theta_i)} & \text{if } j \in \mathscr{C}_\kappa, \\ 0, & \text{otherwise,} \end{cases}$$
$$p_\kappa(u|\theta) = \sum_{i\in\mathscr{C}_\kappa} \alpha_i\, p(u_t|\theta_i). \tag{27}$$

At the extreme case $\kappa = k$, $p_k(u|\theta)$ is actually a finite mixture [49], and $p_{j,t}$ by Eq. (27) provides a Bayes posterior sharing allocation. Generally, when $\kappa < k$, $p_{j,t}$ by Eq. (27) provides a sharing allocation only among learners proportionally with κ maximum posteriors, which is thus shortly referred as *κ-map sharing*.

Furthermore, if we have a priori knowledge on $p(\theta_j)$, it follows from Eq. (25) with $p(u_t, \theta_j)$ and $p(u_t, j, \theta_j)$ that we get an individual criterion that considers generalization in a Bayesian sense [35,50].

On the other hand, given individual criteria $\{\varepsilon_t(\theta_j)\}_{j=1}^k$, it follows from Eq. (26) that we can also get[1]

$$p_j \propto \mathrm{e}^{-\beta_j^{-1}\varepsilon_t(\theta_j)}, \tag{28}$$

which can be further normalized into the following density:

$$p(\varepsilon_t|\theta_j) = \beta_j^{-1}\mathrm{e}^{-\beta_j^{-1}\varepsilon_t(\theta_j)}, \quad \beta_j = E[\varepsilon_t] = \int \varepsilon\, p(\varepsilon|\theta_j)\, \mathrm{d}\varepsilon, \tag{29}$$

where $\beta_j = E[\varepsilon_t]$ provides a further insight on Eq. (18). We may also turn $\mathrm{e}^{-\beta_j^{-1}\varepsilon_t(\theta_j)}$ into a density on the domain of e_t^r, e_t^a via $\int \mathrm{e}^{-\beta_j^{-1}\varepsilon_t(\theta_j)}\, \mathrm{d}e_t^r\, \mathrm{d}e_t^a$ or on the domain of x_t, ζ_t via $\int \mathrm{e}^{-\beta_j^{-1}\varepsilon_t(\theta_j)}\, \mathrm{d}x_t\, \mathrm{d}\zeta_t$, though these integrals may be difficult to handle except some special cases.

Similarly, an extension of $\varepsilon_t(\theta_j) = \alpha_j\|x_t - \mu_j\|^2$ in Eq. (8) can be an alternative of Eq. (26) as follows:

$$\varepsilon_t(\theta_j)/\beta_j = -\ln p(u_t|\theta_j)^{\alpha_j}, \quad \alpha_j \geqslant 0, \ \sum_j \alpha_j = 1. \tag{30}$$

In Ref. [16], RPCL extensions is classified as Type A if it bases on Eq. (26) and Type B if it bases on Eq. (30). Also, we get the counterpart of Eq. (28) by

$$p(u_t|\theta_j) \propto \mathrm{e}^{-\varepsilon_t(\theta_j)/\alpha_j\beta_j}. \tag{31}$$

Probabilistic approach also provides an alternative direction of seeking a global criterion for combing policy. E.g., it follows from Sections 3.4, 3.4.4 and 3.4.5 that the criterion by Eqs. (19) with $p_{j,t}$ by Eqs. (6) and (21) can be obtained from a special case of BYY harmony learning. A global criterion implies an allocation scheme. For an example, Eq. (6) is obtained by minimizing the criterion by Eq. (19) with respect to $p_{j,t}$ subject to $\sum_j p_{j,t} = 1$, $p_{j,t} \geqslant 0$, and Eq. (21) is obtained by minimizing this criterion with respect to $p_{j,t}$ subject to

$$\sum_j p_{j,t} = 1, \quad p_{j,t} \geqslant 0 \quad \text{and}$$
$$p_{j,t} = c_t\, \exp\left(\frac{-\varepsilon_t(\theta_j)}{\beta_j}\right), \quad j \in \mathscr{C}_\kappa. \tag{32}$$

Also, Eqs. (6) and (21) can be obtained from a special case of BYY harmony learning by Eq. (69). As shown later in Eq. (83), the RPCL one by Eq. (9) and its extension by Eq. (22) can also be obtained from the BYY harmony learning.

Moreover, Eq. (21) with $\kappa = k$ closely relates to a ML criterion. It follows from Eq. (27) that $\nabla_{\theta_j} \ln p_k(u_t|\theta) = -p_{j,t}\nabla_{\theta_j}\varepsilon_t(\theta_j)$, we observe that the updating $\theta_j^{new} = \theta_j^{old} + \eta\nabla_{\theta_j} \ln p(u_t|\theta_j)$ is equivalent to making a gradient ascent updating of the likelihood criterion on $p_k(u_t|\theta)$. That is, it actually leads to an adaptive EM algorithm that implements the ML learning on a finite mixture.

A global criterion or combing policy aims at a best overall performance. For this, the correct number k of structures underlying samples must be determined and the samples from a same structure are allocated to a same leaner (it may compose of more than one learners that form a compound learner) such that each learner performs best according to its corresponding individual criterion. In the literature, the task of determining the number k belongs to what is usually called *model selection*, and the task of determining parameters in each learner belongs to what is usually called *parameter learning*.

In past decades, several typical model selection theories have been proposed for model selection, such as Akaike's information criterion (AIC) [51], Bozdogan's consistent Akaike's information criterion (CAIC) [52], Schwarz's Bayesian inference criterion (BIC) [53] which coincides with Rissanen's minimum description length (MDL) criterion [54]. In implementation, model selection has to be made in a *two-phase* procedure. At the first phase, a number of candidate models are enumerated, and the unknown parameters in each candidate are estimated by the ML principle. At the second phase, one of the above criteria is used on every candidate with estimated parameters to get the best candidate. These existing approaches face two major problems. One is the performance problem. In the cases of a small size of samples, each criterion actually provides a rough estimate that cannot guarantee to give the correct result but has a high chance to give a wrong one. Also, one criterion works better in this case and the other may work better in that case, none can be said better than the others. The second problem is that the enumeration needs a vast computing cost, which is infeasible in many real applications.

Favorably, as to be further discussed in Table 2 in the next subsection, an appropriate combination of a global policy $p_{j,t}$ and an individual criterion can make an appropriate k determined during learning parameters of each learner, which is referred as *automatic model selection* during parameter learning.

3.2.3. A general framework for multi-learner problem solving

To implement a process of problem solving, the three ingredients in Section 3.2.1 are integrated into a general framework

[1] In this paper, $u \propto v$ denotes either 'a scalar u is proportional to a scalar v.' or 'a vector u and a vector v share a same direction.'

Table 2
Allocations versus individual criteria

$P_{j,t}$ \ $\frac{\varepsilon_t(\theta_j)}{\beta_j}$	$\ln[\beta_j p(\varepsilon_t\|\theta_j)]$ e.g., $\|x_t-\mu_j\|^2$	$\alpha_j \ln[\beta_j p(\varepsilon_t\|\theta_j)]$ e.g., $\alpha_j\|x_t-\mu_j\|^2$	$\ln(\alpha_j p_j)$, e.g., $p_j = G(x_t\|\mu_j, \Sigma_j)$
WTA	CL (with dead units)	FS–CL (no model selection ability)	BYY–CL on finite mixture (with automatic model selection)
κ-map sharing	κ-map CL (dead units reduced)	κ-map FS–CL (still poor in model selection)	BYY κ-map (with automatic model selection)
Bayes posteriori sharing	Simplified EM (for *ML* or its special case, not good on model selection)	FS type Simplified EM	EM on finite mixture
Penalized allocation	Bare RPCL (with automatic model selection)	RPCL	BYY–RPCL on finite mixture

With the first column and first row not counted, there are 4×4 blocks, the block (i, j) denotes the cross of the ith column and jth row.

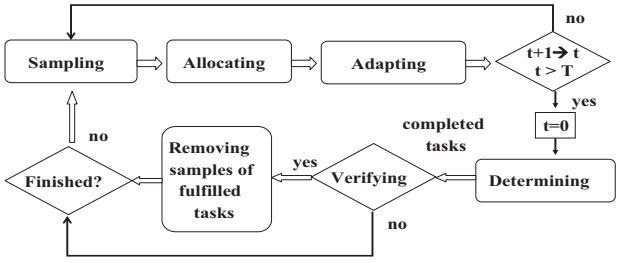

Fig. 9. A general framework for multi-learner problem solving.

as shown in Fig. 9. Being different from the cases in Fig. 4, an assumption solution is provided by a learner, with its knowledge about environment and its action. With each leaner initialized to a given status, samples from the environment are allocated to learners and thus their corresponding assumptions are activated and modified to adapt the environment. Similar to the nature of automatic model selection by the RPCL learning, as learning proceeds, some of learners may become unnecessary and thus be driven away or removed, while the others will finally stabilized with their corresponding assumptions as candidate solutions. Similar to the general framework shown in Fig. 4, the framework in Fig. 9 is still featured by a specific integration of the following five mechanisms:

Sampling: Information is acquired via randomly picking one sample x_t per epoch. However, picking $k > 1$ samples per epoch is applicable by only considering the k samples that basically come from a same object or structure.

Allocating: Similar to the *mapping mechanism* in Fig. 4, information carried by x_t is allocated to learners by an allocation scheme $p_{j,t}$ via Eq. (20). We can simply get $\varepsilon_t(\theta_j)/\beta_j$ via Eq. (26) for a probabilistic model by Eq. (23); otherwise it follows from Eq. (18) that we can update

$$\begin{aligned}&\beta_j^{r\ new} = (1-\eta p_{j,t})\beta_j^{r\ old} + \eta p_{j,t}\varepsilon_t^r(\theta_j),\\ &\beta_j^{a\ new} = (1-\eta p_{j,t})\beta_j^{a\ old} + \eta p_{j,t}\varepsilon_t^a(\theta_j),\\ &\text{get } \varepsilon_t(\theta_j)/\beta_j \text{ by Eq. (17).}\end{aligned} \tag{33}$$

An allocation scheme and an individual criterion jointly determine the nature of the entire learning process. As shown in Table 2, the first two columns consider $\varepsilon_t(\theta_j)$ directly given by Eq. (14) or (17), equivalently $\ln[\beta_j p(\varepsilon_t|\theta_j)]$ by Eq. (29), $p_{j,t}$ by Eq. (6) lead to the previously CL and FSCL, and $p_{j,t}$ by Eq. (9) or (22) leads to RPCL and variants with the nature of automatic model selection. The last column considers a finite mixture $p_k(u|\theta)$ in Eq. (27) at $\kappa = k$ such that $p_{j,t}$ by Eq. (6) leads to an extension from CL to BYY harmony learning (shortly BYY–CL), and $p_{j,t}$ by Eq. (9) or (22) leads to an extension from RPCL to the Bayesian harmony learning (shortly BYY–RPCL), still with a nature of automatic model selection during parameter learning. The details are referred to Section 3.4.4.

We further consider the third row, $p_{j,t}$ by Eq. (21) plus Eq. (26) leads to the Bayes posterior sharing by Eq. (27) when $\kappa = k$. As discussed after Eq. (32) in Section 3.2.2, the block (3,3) leads to an adaptive EM algorithm that implements the ML learning on a finite mixture. Here k needs to be pre-specified since it is well known that the ML learning is not good at model selection, especially on a small size of samples. Moreover, the block (3,1) and the block (3,2) become equivalent at a degenerated case at $p(u_t|\theta) = G(x_t|\mu_j, \sigma^2 I)$, $\beta_j = \sigma^2$, and $\alpha_j = 1/k$.

Balancing the two extreme cases (namely WTA and Bayes posterior sharing), the second row is featured by the κ-map sharing allocation by Eq. (21) with $1 < \kappa < k$. Further discussion will be made at the end of Section 3.3.1.

Adapting: As a further extension of Eq. (7), the activated learners or assumptions are modified to adapt x_t according to the allocation by $p_{j,t}$, e.g.,

$$\begin{aligned}&\theta_j^{new} = \theta_j^{old} + \eta p_{j,t}\delta\theta_j, \quad \delta\theta_j = \text{Reduce}(\varepsilon_t(\theta_j^{old})),\\ &\theta_j^{old}, \theta_j^{new} \in D(\theta_j),\end{aligned} \tag{34}$$

where $\eta > 0$ is a learning step size, and $D(\theta_j)$ is the domain of θ_j, within which θ_j may need to satisfy certain constraints. Moreover, $Reduce(\varepsilon_t(\theta_j))$ denotes an operator that acts on θ_j^{old} and x_t to yield a direction along which $\varepsilon_t(\theta_j)$ reduces within a neighborhood of θ_j^{old}. When $\varepsilon_t(\theta_j)$ is differentiable, $Reduce(\varepsilon_t(\theta_j))$ can be given by $-\nabla_{\theta_j}\varepsilon_t(\theta_j)$ or its projection onto $D(\theta_j)$.

Moreover, each α_j is also updated in proportional to $p_{j,t}$. When $p_{j,t} \geqslant 0$, one simplest way is $\alpha_j = \sum_{t=1}^{N} p_{j,t}/N$ that may be adaptively updated by

$$\alpha_j^{new} = \begin{cases} (\eta + \alpha_j^{old})/(\eta + \sum_{j=1}^{k} \alpha_j^{old}) & \text{if } j = \arg\max_\ell p_{\ell,t}, \\ \alpha_j^{old}/(\eta + \sum_{j=1}^{k} \alpha_j^{old}) & \text{otherwise.} \end{cases} \tag{35}$$

When $p_{j,t} < 0$, the updating needs to satisfy $\alpha_j \geqslant 0$, $\sum_j \alpha_j = 1$, which can be made by

$$v_j^{new} = v_j^{old} + \eta(p_{j,t} - \alpha_j^{old}), \quad \alpha_j = e^{v_j} \Big/ \sum_{j=1}^{k} e^{v_j}. \tag{36}$$

Determining: Similar to the fact that RPCL drives extra learners away, for those in Table 2 with automatic model selection nature, we can determine whether some learners should be removed if they are tending to dead (e.g., not be allocated with samples for a long period), which can be made via monitoring some variables that can be used as indicators. E.g., one typical way is initially letting $\alpha_j = 1/k$ for every j and then acts as follows:[2]

$$\begin{aligned} &\text{if } \alpha_j \to 0 \text{ is detected, discard the } j\text{th structure,} \\ &\quad k \leftarrow k - 1. \end{aligned} \tag{37}$$

For a specific problem there may be also other indicators for this purpose.

Finally, we need to determine whether learners are stabilized to provide candidate solutions.

Verifying: Similar to that in Fig. 4.

3.3. Further details on several typical problem solving tasks

3.3.1. Gaussian mixture and coordinated mechanisms

In 1998 [16,42], RPCL learning is extended to elliptic clusters with each cluster described by a general Gaussian density $G(x|\mu_j, \Sigma_j)$. Here, we further elaborate such a case, i.e., it follows from Eq. (26) that each learner is given probabilistically by $\varepsilon_t(\theta_j) = -\ln G(x|\mu_j, \Sigma_j) - \ln \alpha_j$, from the perspective of the general framework introduced in Fig. 9. Corresponding to the three typical allocation schemes in Section 3.2.1, namely, *WTA*, *shared*, *and penalized*, the discussion made on Table 2 tells us that this general framework will implement one of the following choices:

- An adaptive EM algorithm listed at the block (3,3), that makes the ML learning on the Gaussian mixture $p_k(x|\theta) = \sum_{j=1}^{k} \alpha_j G(x|\mu_j, \Sigma_j)$ with Eq. (21) at $\kappa = k$ for $p_{j,t}$ and Eq. (35) for α_j, as well as Eq. (34) becoming Eq. (7) plus

$$\begin{aligned} &\Sigma_j^{new} = (1 - \eta p_{j,t})\Sigma_j^{old} + \eta p_{j,t} \Delta_{\Sigma_j}, \\ &\Delta_{\Sigma_j} = h^2 I + e_{j,t} e_{j,t}^{\mathrm{T}}, \\ &e_{j,t} = x_t - x_{j,t}, \quad x_{j,t} = x_t(\theta_j) = \mu_j^{old}. \end{aligned} \tag{38}$$

  This algorithm implements not only the ML learning for $h^2 = 0$ in an adaptive version of the well-known EM algorithm [49,77,81], but also a smoothing regularization for a $h^2 > 0$. The details are referred to Section 3.4.4 and Eq. (79).
- BYY–RPCL listed at the block (4,3), that extends the RPCL learning in help of Eq. (9) (or more generally Eq. (22)) for $p_{j,t}$ and Eq. (35) for α_j, with the number of Gaussians determined automatically during learning. In this case, there is $p_{j,t} < 0$ that makes Eq. (38) no longer apply since it no longer guarantees that Σ_j is positive definite. Instead, we replace Eq. (38) with

$$\begin{aligned} &\Sigma_j = S_j S_j^{\mathrm{T}}, \quad S_j^{new} = S_j^{old} + \eta p_{j,t} G_{\Sigma_j}^{old} S_j^{old}, \\ &G_{\Sigma_j} = \Sigma_j^{-1} \Delta_{\Sigma_j} \Sigma_j^{-1} - \Sigma_j^{-1}. \end{aligned} \tag{39}$$

  Again, as will be further introduced in Eq. (79), it implements a smoothed BYY–RPCL for a $h^2 > 0$.
- BYY–CL at the block (1,3), κ-map at the block (2,1), and BYY κ-map at the block (2,3) all extend CL to a RPCL like mechanism in help of $p_{j,t}$ by either Eq. (6) or (21) at $\kappa < k$, with Eq. (35) for α_j and Eq. (38) for Σ_j. It also makes the number of Gaussians determined automatically. The reason will be explained not only later in Section 3.4 from a perspective of the BYY harmony learning but also in the sequel from a perspective of three mechanisms in coordination.

As introduced in Section 3.1, the automatic model selection nature by RPCL comes from a coordination of two opposite mechanisms, namely one for each learner to participate and the other for each learner to leave. The WTA allocation by Eq. (6) with a local criterion $\varepsilon_t(\theta_j) = \|x_t - m_j\|^2$ leads to the previous CCL. It has the *dead unit* problem because learners do not have an equal chance to participate. This problem can be solved via a participating mechanism that let each learner to have an equal initial chance to participate, via either its local criterion or the global criterion.

For the previous FSCL that bases still on the WTA allocation by Eq. (6), its local criterion by Eq. (8) provides a participating mechanism that penalizes those frequent winners and rewards those constant losers. On the other hand, we can still use $\varepsilon_t(\theta_j) = \|x_t - m_j\|^2$ but minimize the global criterion by Eq. (19) under the constraint by Eq. (32), resulting in Eq. (21) that lets all the learners in $\mathscr{C}_\kappa$ to get a chance $p_{j,t} > 0$ to participate per sample comes.

However, all the above cases are lack of a mechanism for an incapable/extra learner to leave. As a result, they will still fail when the number of learners in consideration is larger than an appropriate number. As previously introduced, RPCL improves FSCL via penalizing the rival by Eq. (9), which provides a mechanism to drive an incapable/extra learner away. An appropriate balance between this leaving mechanism and the participating mechanism leads to the favorable RPCL nature of automatic model selection, with the balance controlled by the de-learning rate γ in Eq. (9). Moreover, instead of providing a participating mechanism via a local criterion by Eq. (8), a balance between the mechanisms for participating and leaving

[2] Where a number $\xi \to 0$ means that ξ can be regarded as 0 or approaching 0 according to some detecting technique.

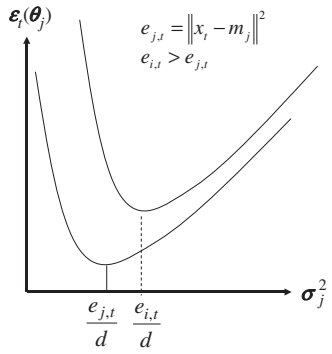

Fig. 10. Rewarding vs penalizing via variances.

can be obtained via the generalized RPCL allocating scheme by Eq. (22).

Also, we can get a leaving mechanism via an appropriate local criterion, e.g., by Eq. (26) instead of using a global allocating policy such as Eq. (9) or (22). Without losing generality, still using the WTA allocating policy by Eq. (6), we consider Eq. (26) with a special Gaussian $G(x|m_j, \sigma_j^2 I)$. It follows from that $\varepsilon_t(\theta_j) = -\ln \alpha_j + 0.5[\|x_t - m_j\|^2/\sigma_j^2 + d \ln(2\pi\sigma_j^2)]$ varies with σ_j^2 as shown in Fig. 10, where d is the dimension of x. If the jth learner wins a lot of samples, σ_j^2 will become larger than a threshold, not only $\varepsilon_t(\theta_j)$ will increase but also the gap between different learners reduces. In other words, the competing ability of learners that won too much in past is self-penalized and the competing ability of weaker learners is rewarded for a relative boosting, which thus provides a participating mechanism similar to α_j in Eq. (8). It can also be observed from the term $0.5\|x_t - m_j\|^2/\sigma_j^2$ that $\sigma_j^{-2} \propto \alpha_j$ takes a role similar to α_j in Eq. (8).

More than the above, when the jth learner won too fewer samples with its σ_j^2 dropping below certain threshold, not only $\varepsilon_t(\theta_j)$ will increase but also the corresponding gap between different learners increases. As a result, an incapable/extra learner is self-penalized to fade out, which also provides a leaving mechanism that is different from the RPCL one by Eq. (9). Moreover, as the competing ability of the jth learner decreases, α_j decreases and the other α_i will relatively increase. That is, the term $-\ln \alpha_j$ enhances this self-penalizing featured leaving mechanism. As a result, the local criterion internally seeks an appropriate balance between its participating and leaving mechanisms, with an automatic model selection nature.

Putting $p(x|\theta_j) = G(x|\mu_j, \sigma_j^2 I)$ in Eq. (30), we get that $\varepsilon_t(\theta_j)$ varies with σ_j^2 in a way similar to Fig. 10. Thus, the above discussion applies too. Differently, the role of α_j is similar to that in Eq. (8), which enhances a participating mechanism instead of a leaving mechanism. Generally, with $p(x|\theta_j) = G(x|\mu_j, \Sigma_j)$ in Eq. (30), we are lead to a finite product of Gaussians as an extension of Eq. (8).

The second row in Table 2 can also be understood from the view of seeking a balance between the mechanisms for participating and leaving. Between the two extreme cases (namely WTA and Bayes posterior sharing), the κ-map sharing allocation by Eq. (21) with $1 < \kappa < k$ will enhance the participating mechanism, and thus improves CL with dead units reduced. Similarly, it may also improve BYY–CL with an appropriate κ. However, it has no help on FSCL due to too stronger a participating force by α_j in Eq. (8).

3.3.2. Local subspace and local factor analysis

For a finite size of samples in a high dimension space, Σ_j in $G(x|\mu_j, \Sigma_j)$ becomes singular easily, which means that a Gaussian structure locates actually in a subspace of a lower dimension. It is not adequate to only consider either the general case Σ_j or the degenerated case $\Sigma_j = \sigma_j^2 I$. We further consider Σ_j in the following decomposition:

$$\Sigma_j = \sigma_j^2 I + \sum_{i=1}^{m_j} \lambda_j^{(i)\,2} a_j^{(i)} a_j^{(i)\,\mathrm{T}},$$
$$a_j^{(i)\,\mathrm{T}} a_j^{(i)} = 1, \quad a_j^{(i)\,\mathrm{T}} a_j^{(\iota)} = 0, \quad i \neq \iota, \tag{40}$$

where $\lambda_j^{(1)\,2} \geqslant \lambda_j^{(2)\,2} \cdots \geqslant \lambda_j^{(m_j)\,2}$ with each $\lambda_j^{(i)\,2}$ being the variance of the projection $a_j^{(i)\,\mathrm{T}} x$ on the direction of the ith principal vector $a_j^{(i)}$. This expression by Eq. (40) actually represents a subspace located at μ_j, as shown in Fig. 11(a). Thus, our task becomes finding subspaces at different locations, which is called local subspace analysis. Instead of directly updating Σ_j, we can update $\sigma_j^2, \{\lambda_j^{(i)\,2}, a_j^{(i)}\}_{i=1}^{m_j}$ via Eq. (34) by considering $\nabla_{\{a_j^{(i)}, \lambda_j^{(i)\,2}\}_{i=1}^{m_j}, \sigma_j^2} \varepsilon_t(\theta_j)$ subject to $a_j^{(i)\,\mathrm{T}} a_j^{(i)} = 1$, $a_j^{(i)\,\mathrm{T}} a_j^{(\iota)} = 0$, where $\varepsilon_t(\theta_j)$ is given by $-\ln G(x_t|\mu_j, \Sigma_j) - \ln \alpha_j$. The details are referred to Sections 3.2 and 3.3 of [18,43].

As illustrated in Fig. 11(a), the subspace obtained via the decomposition by Eq. (40) is equivalent to orthogonally projecting each sample x onto a subspace that is located at μ and spanned by vectors a_1, a_2, a_3, such that the average square error $\|e\|^2$ between x and its projection $\hat{x}$ is minimized. When only the first principal component is considered, we get local PCA for detecting multiple lines, as shown in Fig. 11(b). We can also detect multiple planes as shown in Fig. 11(c). Using the transformation technique in Ref. [55], we can also detect multiple curves and surfaces. Some applications are referred to Refs. [22,23].

However, we are unable to determine those unknown dimensions of subspaces, though it follows from the previous subsection that we are able to determine k via implementing a RPCL learning or a BYY harmony learning. Alternatively, it follows from $e = x - \hat{x}$, $\hat{x} = Ay + \mu$ that this subspace analysis is equivalent to the special case $\Sigma_j = \sigma_j^2 I$ of the following factor analysis (FA) [56,57]:

$$x = A_j y + \mu_j + e_j, \quad e_j \sim G(e_j|0, \Sigma_j),$$
$$y \sim G(y|0, I), \quad E(e_j y^{\mathrm{T}}) = 0, \quad \text{or}$$
$$p(x_t|\theta_j) = \int G(x|A_j y + \mu_j, \Sigma_j) G(y|0, I)\, \mathrm{d}y$$
$$= G(x|\mu_j, A_j A_j^{\mathrm{T}} + \Sigma_j), \tag{41}$$

where $u \sim p(u)$ means that u comes from $p(u)$. In a general case $\Sigma_j \neq \sigma_j^2 I$, the project $x \to \hat{x}$ is still linear but its direction

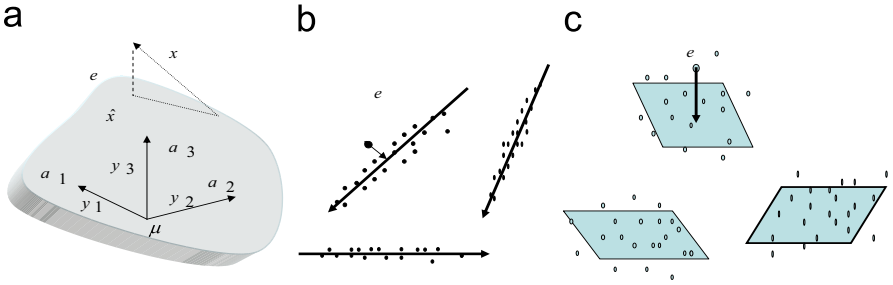

Fig. 11. Subspaces.

is no longer orthogonal to the subspace. Also, the average error $\|e\|^2$ is no longer minimized.

Since a rotation transform on $y \sim G(y|0, I)$ results in $y' \sim G(y|0, I)$ still, it has no difference to the marginal distribution $p(x_t|\theta_j)$ by Eq. (41) whether A_j is a general matrix or subject to the following constraint:

$$A_j = U_j \Lambda_j, \quad U_j U_j^{\mathrm{T}} = I, \quad \Lambda_j = \mathrm{diag}[\lambda_j^{(1)\ 2}, \ldots, \lambda_j^{(m_j)\ 2}]. \tag{42}$$

Moreover, this $p(x_t|\theta_j)$ still remains unchanged when

$$A_j = U_j, \quad y \sim G(y|0, \Lambda_j),$$
$$U_j = \begin{cases} U_j U_j^{\mathrm{T}} = I, \text{ i.e., } U_j \text{ is orthogonal} & \text{(a)}, \\ \text{upper-triangle with diagonal} & \text{(b)}, \\ \quad \text{elements being 1} \end{cases} \tag{43}$$

where the components of y remain uncorrelated but not from $y \sim G(y|0, I)$. When k and $\{m_j\}_{j=1}^k$ are given, the choices by Eqs. (41)–(43) have no difference for making a ML learning.

The situation becomes quite different when k and $\{m_j\}_{j=1}^k$ are unknown. The performance by the ML learning deteriorates since the ML learning is not good at determining these unknowns, while on the choices by Eq. (42) or (43), k and $\{m_j\}_{j=1}^k$ can be determined automatically during either the RPCL learning [16,18] or the BYY harmony learning [45–48]. All of them are included in the framework in Fig. 9, especially those in Table 2 with automatic model selection nature, via the following probabilistic model:

$$\varepsilon_t(\theta_j)/\beta_j = -\ln[p(u|\theta_j)\alpha_j], \quad u_t = (x_t, y_t),$$
$$p(u|\theta_j) = \begin{cases} G(x|A_j y + \mu_j, \Sigma_j) G(y|0, I) & \text{for Eq. (42)}, \\ G(x|A_j y + \mu_j, \Sigma_j) G(y|0, \Lambda_j) & \text{for Eq. (43)}. \end{cases} \tag{44}$$

As each sample x_t comes, we need to get its inner representation y_t to form a $u_t = (x_t, y_t)$ in order to compute $\varepsilon_t(\theta_j)$. It follows from the BYY harmony learning on the FA by Eq. (41) (see, Ref. [45, Eqs. (127) and (129)]) that

$$y_{j,t} = f(x_t, U_j, \Lambda_j),$$
$$f(x, U_j, \Lambda_j) = \begin{cases} [I + A_j^{\mathrm{T}} \Sigma_j^{-1} A_j]^{-1} A_j^{\mathrm{T}} & \text{for Eq. (42)}, \\ \quad \Sigma_j^{-1}(x - \mu_j) & \\ [\Lambda_j^{-1} + A_j^{\mathrm{T}} \Sigma_j^{-1} A_j]^{-1} A_j^{\mathrm{T}} & \text{for Eq. (43)}. \\ \Sigma_j^{-1}(x - \mu_j) & \end{cases} \tag{45}$$

with $u_t = (x_t, y_{j,t})$, we are able to compute $\varepsilon_t(\theta_j)$ and get $p_{j,t}$ by Eq. (20) in one specific form. Then, we update α_j by Eq. (35) or (36) and update θ_j by Eq. (34). Taking Eq. (43)(a) as an example, the detailed form of Eq. (34) becomes

$$\text{(a) } x_{j,t} = U_j^{old} y_{j,t} + \mu_j^{old}, \quad e_{j,t} = x_t - x_{j,t},$$
$$\mu_j^{new} = \mu_j^{old} + \eta p_{j,t} e_{j,t},$$
$$g_{U_j} = \Sigma_j^{old\ -1}(e_{j,t} y_t^{\mathrm{T}} + U_j^{old} \Gamma_j),$$
$$U_j^{new} = U_j^{old} + \eta(g_{U_j} - U_j^{old} g_{U_j}^{\mathrm{T}} U_j^{old}),$$
$$\text{update } \Sigma_j \text{ by Eq. (38) or (39) but with;}$$
$$\Delta_{\Sigma_j} = h^2 I + U_j^{old} \Gamma_j U_j^{old\ \mathrm{T}} + e_{j,t} e_{j,t}^{\mathrm{T}},$$
$$\text{(b) } \Lambda_j^{new} = (1 - \eta p_{j,t}) \Lambda_j^{old} + \eta p_{j,t} \Delta_{\Lambda_j},$$
$$\Delta_{\Lambda_j} = \mathrm{diag}[y_{j,t} y_{j,t}^{\mathrm{T}} + \Gamma_j],$$
$$\text{or in the case with } p_{j,t} \text{ for some } j, \text{ we update } \Lambda_j = V_j^2 \text{ via}$$
$$V_j^{new} = V_j^{old} + \eta p_{j,t} G_{\Lambda_j}^{old} V_j^{old},$$
$$G_{\Lambda_j} = \Lambda_j^{-1} \Delta_{\Lambda_j} \Lambda_j^{-1} - \Lambda_j^{-1}. \tag{46}$$

where $\mathrm{diag}[A]$ denotes the diagonal part of A. Learning is regularized via $h^2 > 0$ and $\Gamma_j > 0$, with the details referred to Section 3.4.5. This regularization can be easily shut off by setting $h^2 = 0$ and $\Gamma_j = 0$.

In addition to determining k by Eq. (37), there is also a problem of determining the dimension of each local subspace, for which $\lambda_j^{(i)}$ acts as an indicator. Initially, letting $\Lambda_j = I$ for every j, and during learning we act as follows:

$$\begin{aligned} &\text{if } \lambda_j^{(i)} \to 0 \text{ is detected, remove the } i\text{th} \\ &\text{coordinate in the } j\text{th subspace,} \\ &\text{then reduce its dimension by } m_j \leftarrow m_j - 1. \end{aligned} \tag{47}$$

Finally, it deserves to mention that the local subspaces have also been extended to multiple temporal state spaces by considering temporal relations. The details are referred to [46, Section IV], especially its Table 2 and Eq. (72).

3.3.3. MML and object detection

Proposed in 1994 [19,20], the multi-sets modelling approach not only relates to but also further extends Gaussian mixture and local subspaces for object detection. An object is described by a

parametric structure S_j^r either by Eq. (3) for a shape such as line, curve, and ellipses or by Eq. (4) for a shape in a template. Then, $e_t^r(\theta_j^r)$ is obtained by Eq. (11), with several typical examples given in Ref. [21, Eqs. (32)–(34)]. Specifically, $e_t^r(\theta_j^r)$ has an analytic expression when S_j^r represents either a linear structure (i.e., a subspace that includes a line, a plane, etc., as shown in Fig. 11) or a circle; while $e_t^r(\theta_j^r)$ has no analytic expression when S_j^r represents a nonlinear structure other than a circle. In such cases, $e_t^r(\theta_j^r)$ is obtained via an optimization procedure, readers are referred to Ref. [58] for an example of detecting ellipses as shown in Fig. 6.

The tasks of object detection can also be implemented by the general framework in Fig. 9, especially those in Table 2 with automatic model selection on the number of objects. The cases with the allocation scheme by Eqs. (6) and (21) at $\kappa = k$ were early suggested in 1994 [19,20]. The probabilistic model by Eq. (29) was firstly suggested also in Ref. [20] and then further extended to consider a density $p(e_t^r|\psi_j)$ in Section 3.3 of Ref. [21] for which it follows from Eq. (11) that $\delta\theta_j^r$ in Eq. (34) can be computed via

$$\delta\theta_j^r = -[\partial \ln p(e_t^r|\psi_j)/\partial e_t^r][\partial x_t(\theta_j^r)/\partial\theta_j^r]. \tag{48}$$

when S_j^r represents a linear structure, such as a line, a plane, a hyperplane, and a subspace, as shown in Fig. 11, as well as a template by Eq. (4) as shown in Fig. 3, we can regard that e_t^r comes from $p(e_t^r|\psi_j) = G(e_t^r|0, \Sigma_j)$, which leads to local PCA with S_j^r for a line, to a local MCA with S_j^r for a plane, and further to two different types of local subspaces with S_j^r for a subspace [21].

In general, it follows from Eqs. (11) and (5) that the map from x_t to e_t^r is implemented via an optimization process. In some cases, we may find an analytic expression $e_t^r = g(x_t, \theta_j^r)$, from which we can get a density supported directly on the domain of x_t as follows:

$$\begin{aligned} &p(x|\theta_j) = p(e_t^r|\psi_j)|\partial g(x, \theta_j^r)/\partial x|, \\ &\varepsilon_t(\theta_j^r) = -\beta_j^{-1}\ln[\alpha_j p(x|\theta_j)]. \end{aligned} \tag{49}$$

3.3.4. Mixture-of-experts, RBF nets, and agents: with external action

Mixture-of-experts: Having been widely studied and applied in the past two decades, it jointly performs function approximation type tasks via the following original model [59]

$$p(\zeta|x, \theta) = \sum_{j=1}^{k} p(j|x, \theta_j^g)p(\zeta|x, \theta_j^a), \tag{50}$$

for a probabilistic mapping $x \to \zeta$. Usually $p(\zeta|x, \theta_j^a) = G(\zeta|\zeta(x|\theta_j^\zeta), \beta_j^a I)$ that provides an action $\zeta(x|\theta_j^\zeta)$ by Eq. (10) under a Gaussian noise of a variance β_j^a. Also, from Eq. (50) we can get the following regression:

$$E[\zeta|x] = \sum_{j=1} p(j|x, \theta_j^g)E[\zeta|x, \theta_j^a], \quad E[\zeta|x, \theta_j^a] = \zeta(x|\theta_j^a). \tag{51}$$

All the unknown parameters are estimated via the ML learning on a set of pairs $\{x_t, \zeta_t\}_{t=1}^N$. However, the gating net $p(j|x, \theta_j^g)$ is itself a parametric model that makes the ML learning on this part unable to be implemented by an exact EM algorithm.

An alternative ME model has also been proposed as follows [60]:

$$p(j|x, \theta_j^g) = \alpha_j p(x|\theta_j^r) \Big/ \sum_{i=1}^{k} \alpha_i p(x|\theta_i^r)\ , \tag{52}$$

with two advantages. First, the ML learning on

$$p(\zeta|x, \theta_j) = \sum_{j=1}^{k} \alpha_j p(x|\theta_j^r)G(\zeta|\zeta(x|\theta_j^\zeta), \beta_j^a I) \tag{53}$$

can be easily implemented by an exact EM algorithm when $p(x|\theta_j) = G(x|\mu_j, \Sigma_j)$. Second, it follows from

$$\varepsilon_t(\theta_j)/\beta_j = -\ln[\alpha_j p(x_t|\theta_j)G(\zeta_t|f(x_t|\phi_j), \rho_j^2 I)], \tag{54}$$

that we can implement the general framework in Fig. 9, especially those in Table 2 with k determined automatically via pushing $\alpha_j \to 0$, together with the parameters updated by Eq. (46) for θ_j plus the following one:

$$\begin{aligned} &\varepsilon_{j,t} = \zeta_t - f(x_t|\phi_j^{old}), \\ &\phi_j^{new} = \phi_j^{old} + \eta p_{j,t}\frac{\partial f^{\mathrm{T}}(x_t|\phi_j)}{\partial\phi_j}\varepsilon_{j,t}, \\ &\rho_j^{new} = \rho_j^{old} + \eta p_{j,t}(\|\varepsilon_{j,t}\|^2/d_\zeta - \rho_j^2)/\rho_j^{old}, \\ &\quad d_\zeta \text{ is the dimension of } \zeta. \end{aligned} \tag{55}$$

RBF nets: The above alternative ME model is simplified into normalized RBF nets and extended normalized RBF nets at the following special case [80]:

$$f(x|\phi_j) = w_j^{\mathrm{T}}x + c_j, \quad \alpha_j = |\Sigma_j| \Big/ \sum_{i=1}^{k} |\Sigma_i|\ . \tag{56}$$

The updating of ϕ_j is simplified into $w_j^{new} = w_j^{old} + \eta\varepsilon_{j,t}x_t$, $c_j^{new} = c_j^{old} + \eta\varepsilon_{j,t}$. As a result, the conventional suboptimal two stage approach for RBF net learning can be replaced by the ML learning via an adaptive EM algorithm. Again, we can implement the general framework in Fig. 9, especially those in Table 2 with automatic model selection on k either via pushing $\alpha_j \to 0$ explicitly or via $\alpha_j = |\Sigma_j|/\sum_{i=1}^k |\Sigma_i|$ with some $|\Sigma_j|$ pushed to 0. Readers are referred to further details in Section 22.9.1(d) of Ref. [48].

Multi-agents: We further consider that each agent is learnt in help of $\varepsilon_t(\theta_j)$ by Eqs. (15)–(17) via the general framework in Fig. 9, especially those in Table 2 with the number of agents determined automatically.

3.4. BYY system and BYY harmony learning

3.4.1. BYY system

Firstly proposed in 1995 [44] and developed in the past decade, the BYY harmony learning is featured by modelling

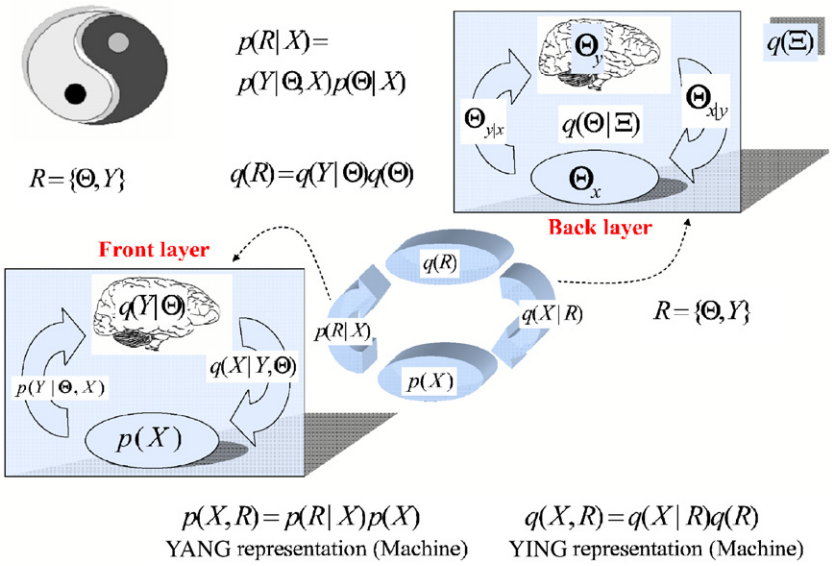

Fig. 12. Bayesian Ying–Yang system.

the problem solving tasks of Types I and II jointly via a unified statistical framework called BYY system in which all the unknowns are learned by a theory under the name of best harmony [45–48]. Here we briefly introduce its key points.

As shown in Fig. 12, we consider a general two-pathway approach from the joint distribution of the external observations X and its inner representations R in the following two types of Bayesian decomposition:

$$p(X, R) = p(R|X)p(X), \quad q(X, R) = q(X|R)q(R). \tag{57}$$

In a compliment to the famous Chinese ancient Ying–Yang philosophy, the one of $p(X, R)$ coincides the Yang concept with a visible domain $p(X)$ for a Yang space and a forward pathway by $p(R|X)$ as a Yang pathway. Thus, $p(X, R)$ is called Yang machine. Similarly, $q(X, R)$ is called Ying machine with an invisible domain $q(R)$ for a Ying space and a backward $q(X|R)$ as a Ying pathway. Such a Ying–Yang pair is called *Bayesian Ying–Yang (BYY) system*.

Specifically, $R = \{Y, \theta\}$ can be divided into two parts, and thus the system is also divided into two layers, as shown in Fig. 12. One is Y that consists of a set of inner encoding or state variables, which timely respond the external environment per sample or per several samples observed. This Y is supported by a parametric substructure $q(Y|\theta_y)$. On one hand, taking a key role in the information flow within the front layer, Y is the source of the information flow to fit the observations $\mathcal{X}_N$ via the top-down pathway that implements the abilities of Type I by another parametric substructure $q(X|Y, \theta_{x|y})$. On the other hand, featuring the abilities of Type II, this Y comes from the information flow via a bottom-up pathway $p(Y|X, \theta_{y|x})$ from $\mathcal{X}_N$, or in a smoothed form:

$$p_h(X) = \prod_{t=1}^{N} G(x_t|\bar{x}_t, hI). \tag{58}$$

In a summary, the front layer itself is a parametric Ying–Yang pair:

$$p(X, Y|\theta_p) = p(Y|X, \theta_{y|x})p_h(X),$$
$$q(X, Y|\theta_q) = q(X|Y, \theta_{x|y})q(Y|\theta_y), \tag{59}$$

which consists of four substructures with each depending on a subset of parameters $\theta = \{\theta_p, \theta_q\}$ with $\theta_p = \{\theta_{y|x}, h\}$ and $\theta_q = \{\theta_y, \theta_{x|y}\}$. Being different from Y that responds the environment per sample, θ is determined collectively from all the samples and represents the common regularities or dependencies among data as its knowledge about the world. This θ is accommodated in the back layer with some a priori structure to back up the front layer. A feedback from the front layer to the back layer goes via $p(\theta|X)$. Moreover, the back layer may be modulated by a meta knowledge from a deeper layer.

The Ying–Yang system in Fig. 12 are featured by two manifolds $p(X, Y|\theta_p)$ and $q(X, Y|\theta_q)$. After the structures $q(Y|\cdot), q(X|Y, \cdot)$, and $p(Y|X, \cdot)$ are pre-designed, the task is to determine three levels of unknowns:

Association or relaxation: The task is to determine Y from X via certain dependence or association within the front layer and is usually referred to under the terms of reasoning, mapping, repression, etc. Also, Y may be indirectly determined via minimizing an energy or cost that incurs from violating certain constraints in S, under the term of relaxation.

Parameter learning: The task is to determine θ, which includes determining Y as a subtask. Determined collectively by all the samples, θ is updated as samples come in a speed much slower than Y. This parameter learning is made via optimizing the performances of implementing either or both of Types I and II abilities, subject to either none or some priori knowledge $q(\theta)$ from the back layer.

Model selection: The scale for representing R (or equivalently the scale of the entire system) is featured jointly by the scale $\mathbf{k}_p$ of $p(X, Y|\theta_p)$ and the scale $\mathbf{k}_q$ of $q(X, Y|\theta_q)$.

One common part shared by both $\mathbf{k}_p$ and $\mathbf{k}_q$ is the scale $\mathbf{k}_Y$ for representing Y. Moreover, $q(X|Y,\theta_{x|y})$ may be designed via a combined architecture, which contributes a rest part $\bar{\mathbf{k}}_q$ of $\mathbf{k}_q$ after taking off the effect of $\mathbf{k}_Y$. Also, $p(Y|X,\theta_{y|x})$ may be designed via a combined architecture too, which contributes another rest part $\bar{\mathbf{k}}_p$ of $\mathbf{k}_p$ after taking off the effect of $\mathbf{k}_Y$. Therefore, the challenging task of model selection actually includes determining each of $\mathbf{k}_Y$, $\bar{\mathbf{k}}_q$, and $\bar{\mathbf{k}}_p$.

Taking clustering analysis by a Gaussian mixture as an example, $\mathbf{k}_Y = k$ with k being the number of Gaussians, while $\mathbf{k}_q = k - 1 + dk + 0.5dk(k+1)$ denotes the effective free parameter number in the set of $m_j, \Sigma_j, j = 1,\ldots,k$ as well as $p(Y=j)$, $j=1,\ldots,k$, where $\mathbf{k}_p$ is equal to the effective number of free parameters for representing the discriminating boundaries.

3.4.2. BYY harmony learning theory and automatic model selection

An analogy of the Ying–Yang system in Fig. 12 to the ancient Chinese Ying–Yang philosophy motivates to determine all the three level unknowns under a best harmony principle, which is mathematically implemented by maximizing the following harmony measure:

$$\begin{aligned} H(p\|q) &= \int p(R|X)p_h(X)\ln[q(X|R)q(R)]\mu(\mathrm{d}X)\mu(\mathrm{d}R) \\ &= \int p(\theta|X)\left\{\int p(Y|X,\theta_{y|x})p_h(X)\right. \\ &\quad \times \ln[q(X|Y,\theta_{x|y})q(Y|\theta_y)q(\theta)] \\ &\quad \left.\times\ \mu(\mathrm{d}X)\mu(\mathrm{d}Y)\right\}\mu(\mathrm{d}\theta), \end{aligned} \tag{60}$$

with respect to $\mathbf{k}_Y$, $\bar{\mathbf{k}}_q$, and $\bar{\mathbf{k}}_p$ as well as $p(\theta|X)$.

Instead of seeking a best θ^* value, it seeks a density $p^*(\theta|X)$ that represents the uncertainty on estimating θ from a finite size of samples. However, it is difficult to have an appropriate priori $q(\theta)$ and to design an appropriate structure for $p(\theta|X)$. To avoid the difficulty, we only seek $p(\theta|X)$ at $X=\mathscr{X}_N$ instead of being as a function of both θ and X, i.e., we consider $p(\theta|X)=p(\theta|\mathscr{X}_N)$ such that Eq. (60) can be further rewritten into

$$\begin{aligned} H(p\|q) &= \int p(\theta|\mathscr{X}_N)H(p\|q,\theta)\,\mathrm{d}\theta, \\ H(p\|q,\theta) &= H_f(p\|q,\theta) - Z(\theta), \\ H_f(p\|q,\theta) &= \int p(Y|X,\theta_{y|x})p(X) \\ &\quad \times \ln[q(X|Y,\theta_{x|y})q(Y|\theta_y)]\,\mathrm{d}X\,\mathrm{d}Y, \end{aligned} \tag{61}$$

where $Z(\theta) = -\ln q(\theta)$ represents a priori knowledge for regularization.

Further noticing the following inequity

$$H(p\|q) = \int p(\theta|\mathscr{X}_N)H(p\|q,\theta)\,\mathrm{d}\theta \leqslant \max_\theta H(p\|q,\theta), \tag{62}$$

we can avoid the difficulty for the maximization of $H(p\|q)$ via approximately seeking the maximization of its upper bound $\max_\theta H(p\|q,\theta)$. That is, maximizing $H(p\|q)$ with respect to $\mathbf{k}_Y$, $\bar{\mathbf{k}}_q$, $\bar{\mathbf{k}}_p$, and $p(\theta|X)$ is replaced by

$$\max_{\{\theta,\mathbf{k}_Y,\bar{\mathbf{k}}_q,\bar{\mathbf{k}}_p\}} H(p\|q,\theta). \tag{63}$$

A simplest and also mostly encountered case is that there is no priori knowledge for $q(\theta)$. Let $q(\theta)=1$ or $Z(\theta)=0$, $\max_h H(p\|q,\theta)$ will force $h=0$ and thus $p_h(X)=p_0(X)$ by Eq. (58). Provided that $\bar{\mathbf{k}}_q$, $\bar{\mathbf{k}}_p$ are pre-given, Eq. (61) becomes

$$\begin{aligned} \max_{\{\theta,\mathbf{k}_Y\}} \Big\{ H_f(p\|q,\theta) &= \int p(Y|\mathscr{X}_N,\theta_{y|x}) \\ &\quad \times \ln[q(\mathscr{X}_N|Y,\theta_{x|y})q(Y|\theta_y)]\,\mathrm{d}Y \Big\}, \end{aligned} \tag{64}$$

which seeks a best harmony within the front layer Ying–Yang system in Fig. 12.

By Eq. (63), the maximization of $H(p\|q)$ by Eq. (60) is approximately implemented because the feedback interaction from the front to the back layer has been ignored with $p(\theta|\mathscr{X}_N)$ not considered. If we take $p(\theta|\mathscr{X}_N)$ in consideration, an improvement would be obtained by seeking $\mathbf{k}=\{\mathbf{k}_Y,\bar{\mathbf{k}}_q,\bar{\mathbf{k}}_p\}$ to maximize an improved estimate of $H(p\|q)$. For this purpose, we expand $H(p\|q,\theta)$ with respect to θ around the resulted θ^* obtained from Eq. (63) up to the second order, ignoring its first order term since $\nabla_\theta H(p\|q,\theta)=0$ at $\theta=\theta^*$, by which the task can be approximately decomposed into the following two stages:

$$\begin{aligned} &\textit{Stage}\ \mathrm{I}:\ \theta* = \arg\max_\theta H(p\|q,\theta) \\ &\qquad\qquad \text{for a set of values of } \mathbf{k}. \\ &\textit{Stage}\ \mathrm{II}:\ \mathbf{k}^* = \arg\min_{\mathbf{k}} J(\mathbf{k}), \quad J(\mathbf{k}) = -\hat{H}(p\|q), \\ &\qquad\qquad \hat{H}(p\|q) = H(p\|q,\theta^*) - 0.5d(\theta^*), \end{aligned} \tag{65}$$

where $d(\theta^*) = -\mathrm{Tr}[\Sigma\partial^2 H(p\|q,\theta)/\partial\theta\partial\theta^{\mathrm{T}}]_{\theta=\theta^*}$ with $\Sigma = \int(\theta-\theta^*)(\theta-\theta^*)^{\mathrm{T}}p(\theta|\mathscr{X}_N)\,\mathrm{d}\theta$. Though it involves the unknown $p(\theta|\mathscr{X}_N)$ and thus is unable to compute, we can still get a rough estimate from the number n_f of free parameters in θ [48,61,62]:

$$d(\theta^*) = \begin{cases} n_f & \text{(a) an under-constraint choice,} \\ 2n_f & \text{(b) an over-constraint choice.} \end{cases} \tag{66}$$

The criterion $J(\mathbf{k})$ in Eq. (65) provides a further improvement on typical existing model selection criteria because the scale of a BYY system is considered separately for the part $\mathbf{k}_Y$ and for the rest parts by $\bar{\mathbf{k}}_q$ and $\bar{\mathbf{k}}_p$. This separation makes the contributions of two parts in $J(\mathbf{k})$ estimated differently. The contribution by $\mathbf{k}_Y$ can be estimated more accurately while the contribution by the rest part is still estimated roughly in a way similar to those typical model selection criteria. As a result, $J(\mathbf{k})$ in Eq. (65) can outperform the typical model selection criteria, as confirmed by experiments [63,64,78,79].

Even interestingly, the model selection problems of many typical learning tasks [62] can be reformulated into selecting merely the $\mathbf{k}_Y$ part in a BYY system. This favorable feature makes both parameter learning for θ and model selection for $\mathbf{k}_Y$ implemented simultaneously during the maximization

of Eq. (64) via a new mechanism, namely automatic model selection during parameter learning. For $q(Y)$ in a scale reducible structure with $\mathbf{k}_Y$ initialized at values large enough, an appropriate $\mathbf{k}_Y$ will be determined automatically during implementing parameter learning $\max_\theta H_f(p\|q,\theta,\mathscr{X}_N)$ by Eq. (64) via maximizing $\int p(Y)\ln q(Y|\theta_y)\,\mathrm{d}Y$ with $p(Y)=\int p(Y|X,\theta_{y|x})p(X)\,\mathrm{d}X$ [61]. Detailed discussions are referred to Sections 22.5 and 23.3.2 in Ref. [48], Section II(B) in Ref. [46], and Section III(C) in Ref. [62]. In other words, the BYY harmony learning by Eq. (64) provides a general theory for learning algorithms with a RPCL-like mechanism that implements model selection automatically during learning.

3.4.3. Approximate implementations and regularization techniques

A detailed implementation of Eq. (63) depends on the types of dependence among samples in $\mathscr{X}_N$. There are two typical scenarios. The simplest but most widely studied case is that samples in $\mathscr{X}_N$ are i.i.d., while the other includes those with a temporal or serial dependence among samples. The later has been studied under the name of temporal BYY learning, with details referred to Refs. [46,65,66]. In the sequel, we only introduce the i.i.d. one.

The inner representation of each sample x may consist of either or both of a set y of a finite number of real variables and a set L of a finite number of a discrete variables. With $p(X)=p_h(X)$ by Eq. (58), Eq. (61) becomes

$$H(p\|q,\theta)=\sum_{t=1}^{N}\sum_{L}p(L|x_t)H_{L,t}-Z(\theta), \tag{67}$$

$$H_{L,t}=\int p(y|x_t,L)\left\{\varrho(x_t,y,\theta)-\frac{1}{2}h^2\mathrm{Tr}[\Psi_L^x(x_t,y,\theta_{x|y})]\right\}\mathrm{d}y,$$
$$\varrho(x_t,y,\theta)=\ln[q(x_t|y,L,\theta_{x|y})q(y|\theta_y,L)q(L)],$$
$$\Psi_L^x(x,y,\theta_{x|y})=-\frac{\partial^2\ln q(x|y,L,\theta_{x|y})}{\partial x\partial x^{\mathrm{T}}},$$

where θ consists of all the unknown parameters in $q(x_t|y,L,\theta_{x|y})q(y,L|\theta_y)q(L)$. Moreover, the second term $h^2\mathrm{Tr}[\cdot]$ comes from $\int G(x|x_t,h^2I)Q(x)\,\mathrm{d}x$ in help of the following type of approximation:

$$\int G(\xi|\mu,\Gamma)Q(\xi)\,\mathrm{d}\xi\approx Q(\xi)_{\xi=\mu}+\frac{1}{2}\mathrm{Tr}\left[\Gamma\frac{\partial^2Q(\xi)}{\partial\xi\partial\xi^{\mathrm{T}}}\right]_{\xi=\mu}. \tag{68}$$

Moreover, if there is no priori constraint on $p(y|x_t,L)$, the integral over y will also disappear during maximizing $H(p\|q,\theta)$, resulting in

$$p(y|x_t,L)=\delta(y-g_L(x_t,\phi))_{g_L(x_t,\phi)=f_L(x_t,\theta)},$$
$$f_L(x_t,\theta)=\arg\max_y\ln[q(x_t|y,L,\theta_{x|y})q(y|\theta_y,L)]. \tag{69}$$

To update θ, we need to compute $\nabla_\theta H(p\|q,\theta)$ via the chain rule on $f_L(x_t,\theta)$. In many cases $f_L(x_t,\theta)$ has no analytic expression but just implements a maximization procedure. One easiest handling is simply ignoring the relation $f_L(x_t,\theta)$ to θ, which is a crude approximation that usually affects performances. A better way is to consider $p(y|x_t,L)$ in an appropriate structure that makes not only the relation to θ get an analytic expression (or partially) but also the integral over y become implementable. One solution is to let

$$p(y|x_t,L)=G(y|g_L(x_t,\phi_L),\Gamma_L), \tag{70}$$

with $g(\mathscr{X}_N,\phi)$ being a pre-specified parametric model. It follows from Eq. (68) that we further get

$$H_{L,t}=\left\{Q(x_t,y,\theta)-\frac{1}{2}\mathrm{Tr}[h^2\Psi_L^x(x_t,y,\theta)]-\frac{1}{2}\mathrm{Tr}[\Gamma_L\Psi_L^y(y,\theta)]\right\}_{y=g_L(x_t,\phi_L)}+\frac{1}{2}\mathrm{Tr}\left[\Gamma_L\frac{\partial^2\mathrm{Tr}[\Psi_L^x(x_t,y,\theta)]}{\partial y\partial y^{\mathrm{T}}}\right]_{y=g_L(x_t,\phi_L)},$$
$$\Psi_L^y(y,\theta)=-\frac{\partial^2\ln[q(x|y,L,\theta_{x|y})q(y|\theta_y,L)]}{\partial y\partial y^{\mathrm{T}}}, \tag{71}$$

where the last term in $H_{L,t}$ is a high order derivatives that vanishes exactly in some cases or can be approximately ignored in many cases.

The parameters h,Γ_L vanish during maximizing $H(p\|q,\theta)$ if $q(\Theta)$ is irrelevant to the parameters. From Eq. (71), we get $H_{L,t}=Q(x_t,y,\theta)_{y=g_L(x_t,\phi_L)}$ that becomes the extreme ridge of the joint likelihood confined to $y=g_L(x_t,\phi_L)$, which leads to a maximum extremal joint likelihood or shortly Max-EJL. Readers are referred to Ref. [61, Sections 3.6 and 5.2] for details. With the improvements, h,Γ_L will be determined in help of certain priori constraints on h^2,Γ_L, e.g., the so-called *data smoothing* and *normalization* [17,18,61,62,65] for details. Here, we suggest two other types of constraints.

One is directly imposed on h,Γ_L, under the name of *equal covariance*, by which an explicit relation between $h^2=h(\theta)$ can be obtained from

$$Var_{p(x|\theta)}(x)=Var_{p_h(x)}(x)\quad\text{with}$$
$$Var_{p_h(x)}(x)=h^2I+S_N, \tag{72}$$

$$p_h(x)=\frac{1}{N}\sum_{t=1}^{N}G(x|x_t,h^2I),\quad \bar{x}=\frac{1}{N}\sum_{t=1}^{N}x_t,$$
$$S_N=\frac{1}{N}\sum_{t=1}^{N}(x_t-\bar{x})(x_t-\bar{x})^{\mathrm{T}},$$
$$p(x|\theta)=\sum_{L}q(x_t|L,\theta)q(L),$$
$$q(x_t|L,\theta)=\int q(x_t|y,L,\theta_{x|y})q(y|\theta_y,L)\,\mathrm{d}y.$$

Also, we can get an explicit relation $\Gamma_L=\Gamma(\theta)$ as follows:

$$\Gamma_L=Var_{G(y|g_L(x_t,\phi_L),\Gamma_L)}(y)=Var_{q(y|x_t,L)}(y),$$
$$q(y|x,L)=q(x_t|y,L,\theta_{x|y})q(y|\theta_y,L)/q(x_t|L,\theta). \tag{73}$$

When both $q(x_t|y,L,\theta_{x|y})$ and $q(y|\theta_y,L)$ are Gaussian, we can analytically not only solve the integral over y for $q(x_t|L,\theta)$

but also get analytical expressions for both $h^2 = h(\theta)$ and $\Gamma_L = \Gamma(\theta)$.

The other type is to indirectly impose a priori $Z(\theta) = -\ln q(h^2) - \ln q(\Gamma_L)$. This $q(h^2)$ is in a form of a scalar density $p(\zeta)$, $\zeta \geqslant 0$. One choice is $p(\zeta) = C^{-1}\zeta^b e^{-a\zeta}$, $a > 0$, $b > 0$, e.g., a χ^2 distribution or a Gamma distribution, as well as other distributions in the table in p. 413 of Ref. [67]. In this case, a trade off occurs on maximizing $f(\zeta) = -0.5\zeta\pi - a\zeta + b\ln\zeta + c$. It follows from $-f'(\zeta) = 0.5\pi + a - b/\zeta = 0$ that $\zeta = b/(0.5\pi + a)$. Specifically, we have $\pi = -\mathrm{Tr}[\Psi_L^x(x_t, y, \theta)]_{y=g_L(x_t,\phi_L)}$ for $\zeta = h^2$, where a, b depend on what type of distribution is chosen, e.g., for a Gamma distribution, $a = 0.5$ and $b = 0.5r - 1$ with r being the 'number of degrees of freedom' that relates to the number N of samples, and the dimension d of x for h. Considering $\Gamma_L = \gamma_L I$, we can also determine γ_L in a similar way.

Furthermore, we move to consider $p(L|x_t)$. If there is no priori constraint on $p(L|x_t)$, the sum over L in Eq. (67) will also disappear during maximizing $H(p\|q, \theta)$, resulting in

$$p(L|x_t) = \begin{cases} 1 & \text{if } L = L_t, \\ 0 & \text{otherwise,} \end{cases} \qquad L_t = \arg\max_L H_{L,t}. \tag{74}$$

It saves the computing cost that is needed to enumerate over all the values of L, which can be huge especially when L consists of quite a number of discrete variables. Similar to the case by Eq. (69), updating θ also needs to compute $\nabla_\theta H(p\|q, \theta)$ via the relation $L_t = \arg\max_L H_{L,t}(\theta)$ but it is not differentiable. We have to approximately ignore the relation with certain suffering on performance. One way to still take the relation back in consideration is the following Bayesian posteriori structure:

$$p(L|x_t) = q(x_t|L, \theta)q(L) \Big/ \sum_L q(x_t|L, \theta)q(L)\ , \tag{75}$$

but with an expensive computing cost. Anyway, it is still workable in some cases, e.g., when L consists of only one integer ℓ and the above integral for $q(x_t|L, \theta)$ is analytically solvable. A trade off between the two conflict purposes is letting $L_t = \arg\max_L H_{L,t}(\theta)$ to be approximated by a parametric model $f(x_t, \psi)$, e.g., a sigmoid semi-linear mapping in Ref. [68, Eq. (34)].

With all the above preparations, we are ready to develop a detailed algorithm for updating θ and ϕ as well via computing $\nabla_{\{\theta,\phi\}} H(p\|q, \theta)$. It deserves to note that the order of removing the integral over x and the integral over y from Eq. (61) will lead to a difference. If we consider to remove the integral over y by Eq. (70) first and then we consider $p(X) = p_h(X)$ by Eq. (58) to remove the integral over x in help of Eq. (68), $H_{L,t}$ in Eq. (71) becomes

$$\begin{aligned} H_{L,t} = {} & \{\varrho(x_t, y, \theta) - \tfrac{1}{2}\mathrm{Tr}[\Gamma_L \Psi_L^y(x, y, \theta)]\}_{y=g_L(x_t,\phi_L)} \\ & - \tfrac{1}{2}\mathrm{Tr}[h^2 \Psi_L^x(x_t, \theta)] \\ & - \tfrac{1}{2}\mathrm{Tr}[h^2 \partial^2 \mathrm{Tr}[\Gamma_L \Psi_L^y(x, y, \theta)]_{y=g_L(x_t,\phi_L)}/\partial x \partial x^{\mathrm{T}}]_{x=x_t}, \end{aligned}$$

$$\Psi_L^y(x, y, \theta) = -\frac{\partial^2 \ln[q(x|y, L, \theta_{x|y})q(y|\theta_y, L)]}{\partial y \partial y^{\mathrm{T}}},$$

$$\Psi_L^x(x, \theta) = -\frac{\partial^2 \ln[q(x|y, L, \theta_{x|y})q(y|\theta_y, L)]_{y=g_L(x,\phi_L)}}{\partial x \partial x^{\mathrm{T}}}. \tag{76}$$

Again, the last term in $H_{L,t}$ is a high order derivative that vanishes exactly in some cases or can be approximately ignored in many cases. Moreover, Eq. (76) becomes equivalent to Eq. (67) simply with $\Gamma_L = 0$ and $g_L(x_t, \phi) = f_L(x_t, \theta)$.

3.4.4. BYY harmony learning on Gaussian mixture

In the special case that $y = \emptyset$ and L consists of only one integer j, it follows from Eqs. (67) and (71) that we are related to $\varepsilon(\theta)$ by Eq. (19) via $\varepsilon_t(\theta_j)/\beta_j = -\ln[q(x_t|\theta_j)\alpha_j]$ as follows:

$$H(p\|q, \theta) = -\varepsilon(\theta) - \omega(\theta) - Z(\theta), \quad Z(\theta) = -\ln q(\theta), \tag{77}$$

$$\varepsilon(\theta) = \sum_{t=1}^N \sum_{j=1}^k p_{j,t}\varepsilon_t(\theta_j), \quad p_{j,t} = p(j|x_t),$$

$$\varepsilon_t(\theta_j) = -\ln[q(x_t|\theta_j)\alpha_j],$$

$$\omega(\theta) = \sum_{t=1}^N \sum_{j=1}^k p_{j,t}\varrho_t(\theta_j, h^2),$$

$$\varrho_t(\theta_j, h^2) = -\frac{1}{2}\mathrm{Tr}\left[h^2 \frac{\partial^2 \ln q(x|\theta_j)}{\partial x \partial x^{\mathrm{T}}}\right].$$

That is, the minimization of $\varepsilon(\theta)$ is regularized by both a term $\omega(\theta)$ for the effect of removing integrals and a term $Z(\theta)$ of a priori knowledge for a finite number of samples. Moreover, $p_{j,t} = p(j|x_t)$ is given by either Eq. (74) or (75), which becomes

$$p_{j,t} = \begin{cases} 1 & \text{if } j = j_t, \\ 0 & \text{otherwise,} \end{cases} \qquad j_t = \arg\min_j [\varepsilon_t(\theta_j) + \varrho_t(\theta_j, h^2)]$$

or

$$p_{j,t} = e^{-\varepsilon_t(\theta_j)} \Big/ \sum_{\ell=1}^k e^{-\varepsilon_t(\theta_\ell)}\ . \tag{78}$$

Particularly, when $h^2 = 0$ and $Z(\theta) = 0$, the above $p_{j,t}$ is equivalent to $p_{j,t}$ by Eq. (21) or (27) with x_t in place of u_t, where the extreme case $\kappa = k$ leads to the Bayes allocation by Eq. (27) or equivalently by Eq. (75).

Considering $q(x_t|\theta_j) = G(x_t|\mu_j, \Sigma_j)$ and thus a Gaussian mixture $q(x_t|\theta) = \sum_{j=1}^k \alpha_j G(x_t|\mu_j, \Sigma_j)$, it follows from Eq. (77) that

$$\varepsilon_t(\theta_j) = -\ln[\alpha_j G(x_t|\mu_j, \Sigma_j)], \quad \varrho_t(\theta_j, h^2) = h^2 \mathrm{Tr}[\Sigma_j^{-1}]. \tag{79}$$

Maximizing $H(p\|q, \theta)$ by Eq. (77) leads to the algorithm with Eqs. (7) and (38) plus Eq. (35) or (36). In the simplest case that we ignore $Z(\theta)$, we are lead to $h = 0$, $p_{j,t}$ by Eq. (6), and an algorithm by Eq. (28) in [21] that was firstly obtained in 1995 [44] under the name of the hard-cut EM algorithm. In general, we also need to determine $h^2 \neq 0$. It follows from Eq. (72) that we get $h^2 I + S_N = \sum_{j=1}^k \alpha_j(\mu_j\mu_j^{\mathrm{T}} + \Sigma_j) - \mu\mu^{\mathrm{T}}$

with $\mu = \sum_{j=1}^{k} \alpha_j \mu_j$, that is,

$$h^2 = \frac{1}{d}\left\{ \sum_{j=1}^{k} \alpha_j \|\mu_j\|^2 - \left\| \sum_{j=1}^{k} \alpha_j \mu_j \right\|^2 + \mathrm{Tr}\left[\sum_{j=1}^{k} \alpha_\ell \Sigma_j - S_N \right] \right\}. \quad (80)$$

Alternatively, we can also determine h^2 by either $Z(\theta) = -\ln q(\theta)$ via data-smoothing [21] or $Z(\theta) = -\ln q(h^2)$ via a χ^2 or a Gamma distribution.

During learning, the selection on k is made via Eq. (37) automatically. Moreover, a better k can be determined with an expensive computing cost at the Stage II. With $\alpha_j = (1/N)\sum_{t=1}^{N} p_{j,t}$, it gets the following detailed form:

$$k^* = \arg\min_k J(k), \quad J(k) = J_C(k) + 0.5d(\theta^*)/N,$$

$$J_C(k) = 0.5\sum_{j=1}^{k} \alpha_j \ln|\Sigma_j| + h^2 \sum_{j=1}^{k} \alpha_j \mathrm{Tr}[\Sigma_j^{-1}] - \sum_{j=1}^{k} \alpha_j \ln \alpha_j - 0.5\frac{kd}{N}, \quad (81)$$

where $d(\theta^*)$ is given by Eq. (66) with $n_f = k - 1 + kd + 0.5d(d+1)k$ with $k-1$ for α_j, kd for μ_j, and $0.5d(d+1)k$ for Σ_j, respectively.

It also deserves to note that we may also be lead to $p_{j,t}$ in a type by Eq. (22) that is similar to RPCL learning [17,18], by considering the case $h^2 = 0$ with

$$q(\theta) \propto 1/V^{\gamma_0}(\theta), \quad 0 < \gamma_0 < 1, \quad V(\theta) = \sum_{t=1}^{N} q(x_t|\theta), \quad (82)$$

for *normalization* [18,21] via $q(x|\theta) = \sum_{i\in\mathscr{C}_\kappa} \alpha_i G(x_t|\mu_i, \Sigma_i)$ with $\mathscr{C}_\kappa$ by Eq. (32). It follows from Eqs. (63) and (79) that $H(p\|q,\theta) = -\varepsilon(\theta) + \ln q(\theta)$ and that $\nabla_{\theta_j} \ln q(\theta) = N^{-1}\gamma_0 \sum_{t=1}^{N} \gamma_t \pi_{j,t} \nabla_{\theta_j} \ln[\alpha_j G(x_t|\mu_j, \Sigma_j)]$ with $\pi_{j,t}$ by Eq. (21) and $\gamma_t = q(x_t|\theta)/\bar{q}(x_t|\theta)$, where $\bar{q}(x_t|\theta) = N^{-1}\sum_{t=1}^{N} q(x_t|\theta)$. We have also $\nabla_{\theta_j}\varepsilon(\theta) = -\sum_{t=1}^{N} q_{j,t} \nabla_{\theta_j} \ln[\alpha_j G(x_t|\mu_j, \Sigma_j)]$ with $q_{j,t}$ given by Eq. (21) too. It further follows from $H(p\|q,\theta) = -\varepsilon(\theta) + \ln q(\theta)$ that

$$\nabla_{\theta_j} H(p\|q,\theta) = \frac{1}{N}\sum_{t=1}^{N} p_{j,t} \nabla_{\theta_j} \ln[\alpha_j G(x_t|\mu_j, \Sigma_j)],$$
$$p_{j,t} = q_{j,t} - \gamma_0 \gamma_t \pi_{j,t}, \quad (83)$$

i.e., we get $p_{j,t}$ in a type by Eq. (22). Another RPCL-like mechanism is also obtained from the BYY harmony learning with $Z(\theta) = 0$ and $p_{j,t}$ by Eq. (27), as suggested in Ref. [17, Eq. (40)] and further demonstrated in Ref. [69].

3.4.5. BYY harmony learning on local subspaces and extensions

For the local FA, the BYY harmony learning in Eq. (42) or (43) is able to make k and $\{m_j\}_{j=1}^{k}$ automatically determined during learning. E.g., in Eq. (43) it follows that Eq. (77) is modified into

$$\varepsilon_t(\theta_j) = -\ln[\alpha_j G(x_t|U_j y + \mu_j, \Sigma_j) G(y|0, \Lambda_j)]_{y=W_j(x_t-\mu_j)},$$
$$\varrho_t(\theta_j, h^2) = h^2 \mathrm{Tr}[\Sigma_j^{-1}] + \mathrm{Tr}[\Gamma_j(U_j^{\mathrm{T}}\Sigma_j^{-1}U_j + \Lambda_j^{-1})],$$
$$y = W_j(x - \mu_j) = \arg\max_y \ln[\alpha_j G(x|U_j y + \mu_j, \Sigma_j) G(y|0, \Lambda_j)],$$

with

$$W_j(x-\mu_j) = \begin{cases} f(x, U_j, \Lambda_j) & \text{(a) by Eq. (45),} \\ \Lambda_j U_j^{\mathrm{T}}(U_j^{\mathrm{T}}\Lambda_j U_j + \Sigma_j)^{-1}(x-\mu_j) & \text{(b) or equivalently.} \end{cases} \quad (84)$$

Then, maximizing $H(p\|q,\theta)$ by Eq. (77) leads to the algorithm by Eq. (46). Again, we can get h^2 by Eq. (80) with Σ_j replaced by $U_j^{\mathrm{T}}\Lambda_j U_j + \Sigma_j$, and it further follows from Eq. (73) that

$$\Gamma_j = \Lambda_j - \Lambda_j U^{\mathrm{T}}(U_j^{\mathrm{T}}\Lambda_j U_j + \Sigma_j)^{-1} U \Lambda_j. \quad (85)$$

Alternatively, we can also determine h^2, Γ_j by either $Z(\theta) = -\ln q(\theta)$ via data-smoothing [21] or $Z(\theta) = -\ln q(h^2)$ via a χ^2 or a Gamma distribution.

During implementing the algorithm, again the model selection on k is made via Eq. (37) with α_j updated by Eq. (35) or (36), while $k^*, \{m_j^*\}_{j=1}^{k}$ are determined automatically via Eq. (47). On a small sample size N, a better $k^*, \{m_j^*\}_{j=1}^{k}$ can be selected as follows:

$$\{k^*, \{m_j^*\}_{j=1}^{k}\} = \arg\min_{k,\{m_j\}} \{J(k, \{m_j\}) + 0.5d(\theta^*)\},$$

$$J(k, \{m_j\}) = \frac{1}{2}\sum_{j=1}^{k} \alpha_j [\ln|\Sigma_j| + \ln|\Lambda_j| + m_j(\ln(2\pi)+1)] - \sum_{j=1}^{k} \alpha_j \ln \alpha_j + \sum_{j=1}^{k} \alpha_j \{h^2 \mathrm{Tr}[\Sigma_j^{-1}] + \mathrm{Tr}[\Gamma_j(U_j\Sigma_j^{-1}U_j^{\mathrm{T}} + \Lambda_j^{-1})]\} - \frac{kd + \sum_{j=1}^{k} m_j}{N}, \quad (86)$$

where $d(\theta^*)$ is still given by Eq. (66) but with $n_f = dk + k - 1 + 0.5d(d+1)k + \sum_{j=1}^{k}(m_j + d_{U_j})$ and $d_{U_j} = m_j d - 0.5m_j(m_j+1)$.

A number of variants and extensions can be obtained. Several typical ones are briefly introduced as follows:

- From Eq. (84), we can obtain other algorithms for implementing Eq. (43), including the ones in Table 2 and the temporal extension by Eq. (72) and Table 2 in Ref. [46] with $B_j = 0, \forall j$. Applications to financial market are referred to Ref. [70].

- The algorithm by Eq. (46) is developed without considering the composite relation via $y = W_j(x_t - \mu_j)$ in computing $\nabla_{\theta_j}\varepsilon_t(\theta_j)$. More precisely, an extended algorithm can be developed by using $(\partial y^{\mathrm{T}}/\partial\theta_j)\nabla_{\theta_j}\varepsilon_t(\theta_j)$ in the place of $\nabla_{\theta_j}\varepsilon_t(\theta_j)$ for the updating rules for $\Lambda_j, U_j, \Sigma_j, \mu_j$.
- Another algorithm can be developed from the $H_{L,t}$ in Eq. (76), i.e., the version of removing the integral over y first and then removing the integral over x.
- We consider other choices for $q(y, L)$ in Eq. (67), one is $L = \{\ell, j, i_1, \ldots, i_{m_\ell}\}$ and $q(y, L) = q(\ell)\prod_{j=1}^{m_\ell} q(y^{(j)}|i_j, \ell)q(i_j|\ell)$, $q(y^{(j)}|i_j, \ell) = G(y^{(j)}|\mu_{i_j,\ell}, \sigma^2_{i_j,\ell})$ such that each independent dimension in a non-Gaussian $q(y^{(j)}|\ell)$ is jointly described by several scalar Gaussian $G(y^{(j)}|\mu_{i_j,\ell}, \sigma^2_{i_j,\ell})$. Readers are referred to Ref. [47, Section IV(C)] for details.
- Even generally, we can consider $q(y, L)$ in a direct graphical model instead of a graphical model. One example is a tree structure with dependence relations from leaves directed up to the root layer by layer. We may explore automatic model selection on the number of nodes (probably a structure as well) by the BYY harmony learning.

4. An unified problem solving paradigm

4.1. A5 problem solving paradigm: A–D featured versus D–A featured

Both the frameworks in Figs. 4 and 9 can be regarded as examples of an unified problem solving paradigm (shortly A5 paradigm) as shown in Fig. 13, consisting of five essential mechanisms as follows:

- *Acquisition*: It is same as in Figs. 4 and 9.
- *Allocation*: It is same as in Fig. 9. Also, the mapping mechanism in Fig. 4 can be regarded as allocating evidences.
- *Amalgamation*: It integrates newly obtained evidence into the activated assumptions, e.g., either by the accumulating mechanism in Fig. 4 or by the adapting mechanism in Fig. 9.
- *Admission*: It admits those activated assumptions as candidate solutions, via the determining mechanism both in Figs. 4 and 9.
- *Affirmation*: It further affirms the candidates as final solutions if they pass the verifying mechanism in both Figs. 9 and 4.

Sometimes, the last mechanism may be waived. Moreover, each of the other four mechanisms may have a number of choices. A specific integration of the choices leads to a specific approach of problem solving. The framework in Fig. 4 and the framework in Fig. 9 are different in their ways of handling two coupled core tasks. One is combining pieces of evidences carried by samples, and the other discriminates according to their evidences. In Fig. 4, information carried by samples are mapped into evidences for assumptions, which are integrated via accumulation. Then, a cross-assumption interaction or comparison occurs in implementing its fourth mechanism, such that an appropriate discrimination is made in the parameter space. While in Fig. 9, such a cross-agent interaction happens during implementing its second mechanism that discriminates and allocates evidences across assumptions. Then, the allocated evidences are amalgamated via its adapting mechanism. The amalgamation–discrimination (A–D) featured scheme in Fig. 4 and the discrimination–amalgamation (D–A) featured scheme in Fig. 9 lead to different characteristics in their implementations, performances, computing costs, and applicable scopes, etc.

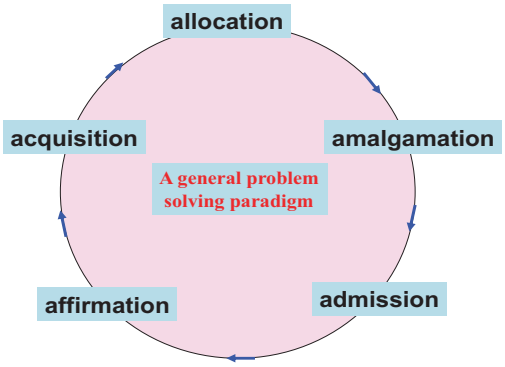

Fig. 13. A5 paradigm.

4.2. Cross-perspectives: insights and extensions

It is also insightful to look HT like approaches as in Fig. 4 from a perspective of learning based approaches. We rewrite Eq. (77) into

$$H_f(p\|q, \theta) = \sum_{t=1}^{N}\sum_{j=1}^{k} h_t(\theta),$$
$$h_t(\theta) = \frac{1}{N}\sum_{j=1}^{k} p_{j,t}\ln[\alpha_j p(x_t|\theta_j)]. \quad (87)$$

That is, information carried by each sample is considered separately, and then integrated via additive accumulation. Actually, the ML learning and many cost based methods also have such an additive nature. The second mechanism in Fig. 4 can be regarded as that each sample x_t contributes a piece of evidence or score $h_t(\theta)$ at every point in the parameter space of θ. So, $\sum_{t=1}^{N} h_t(\theta)$ can be regarded as a batch implementation of the accumulating mechanism in Fig. 4. Then the next mechanism performs $\max_\theta H_f(p\|q, \theta)$ via comparing all the assumptions (i.e., all the points) to determine a solution.

When $h_t(\theta)$ is a nonzero constant merely for points on a manifold in the parameter space, only assumptions on this manifold is accumulated. Considering that each of θ_j, $j = 1, \ldots, k$ has a same dimension, we let all the distributions $p(x|\theta_j)$, $j = 1, \ldots, k$ share a common accumulation grid in the parameter space for a sake of saving space and computing cost, which leads us to exactly the conventional HT. Such a situation happens only in those idealistic cases without noises or distortions. Taking noise or distortion in consideration, we can improve the accumulating mechanism of HT by the choices (b)–(d) in

Section 2.2.3. Also, we can implement

$$a(\theta_j|X, x_t) = a(\theta_j|X) + c_t(\theta_j),$$
$$c_t(\theta_j) \propto \begin{cases} \alpha_j p(x_t|\theta_j) & \text{for HT,} \\ \alpha_j p(\mathbf{x}^m|\theta_j) & \text{for RHT,} \end{cases} \quad (88)$$

where $a(\theta|X)$ denotes the score that the activated assumption θ_j were accumulated via a set X of past samples. Moreover, $p(x_t|\theta_j)$ or $p(\mathbf{x}^m|\theta_j)$ describes the fit of x_t or $\mathbf{x}^m$ to the object represented by θ_j, where $\mathbf{x}^m$ is a subset consisting of m samples. For example, it can be obtained from e_t by Eq. (5) via a Gaussian distribution truncated within a certain band.

Instead of performing $\max_\theta H(k, \theta)$ in a batch by making comparison across all the assumptions, learning can also be performed via the D–A featured scheme in Fig. 9 adaptively per sample x_t. That is, the second mechanism in Fig. 9 discriminates and allocates x_t via $p_{j,t}$, which is computed across all α_j, $p(x_t|\theta_j)$, $j=1, \ldots, k$ that are then updated to adapt the evidence carried by x_t to increase $h_t(\theta)$ by certain extent. In this case, implementing the subsequent admission mechanism does not need a coordination across all the activated assumptions. Alternatively, this observation motivates that a multi-learner based approach can be improved from a perspective of HT like approaches too.

We add a cross-assumption coordination in the implementation of admission mechanism. Each θ_j is also associated with a score $a(\theta_j)$. Not only an activated θ_j is adapted according to $\theta_j^{new} = \theta_j^{old} + \eta p_{j,t} \delta\theta_j$ but also the corresponding score is updated by $a(\theta_j^{new}) = (1-\eta)a(\theta_j^{new}) + \eta c_t(\theta_j^{new})$ with $c_t(\theta_j^{new}) \propto \alpha_j p(x_t|\theta_j^{new})$, such that a comparison across all the activated assumptions can be made to select those assumptions with enough scores.

In RHT, we can also implement accumulation mechanism jointly with the mechanisms of allocation and adaptation. After an assumption θ is activated via mapping mechanism, we check its difference from its closest neighbor θ^* among those activated assumptions in past. If the difference is larger than a threshold, we take this activated assumption with a score by Eq. (88); otherwise, we allocate evidences carried by $\mathbf{x}^m$ to θ^*, and then update

$$\theta^{*new} = \theta^* + \eta\delta\theta, \quad \delta\theta = \theta - \theta^*,$$
$$a(\theta^{*new}|X, x_t) = (1-\eta)a(\theta^*|X) + \eta c_t(\theta^*),$$
$$c_t(\theta^*) \propto \alpha_j p(\mathbf{x}^m|\theta). \quad (89)$$

4.3. Evidence combination: further insights

Started from handwritten character recognition at the end of 1980s, studies on evidence combination have become popular in the fields of pattern recognition and information fusion [82,71], which can be summarized from two major aspects. One considers which level a combination is made on (e.g., decision level, feature level, and data level). The other aspect is to consider what types of rules a combination bases on. These rules can be further summarized roughly into two classes. One is featured by combinations made either explicitly or implicitly within a probabilistic framework, e.g., a majority voting rule or a Bayesian rule in Ref. [82]. The other consists of combinations made beyond the probabilistic framework, e.g., the rules from D–S evidence theory in Ref. [82].

In this paper, combination is made on choosing best assumptions as solutions (i.e., on decision level), which can be classified into the two categories:

- One is on assumptions that are obtained from a same piece or collection of evidences but via different mechanisms. These mechanisms may be either of different types (e.g., a diverging mapping, a converging mapping, an adaptation) or of a same type (e.g., the converging mapping is used on different numbers of pixels to assume different lines to be detected), respectively. More formally, on a same piece of evidence x we have k mechanisms $M_j : x \to \theta$, $j = 1, \ldots, k$ to make or activate assumptions, which are shortly denoted as $M_j(\theta|x)$, what we want is to find a $M(\theta|x) = Combine[M_1(\theta|x), M_2(\theta|x), \ldots, M_k(\theta|x)]$.
- The other category is on assumptions that are obtained from applying a same mechanism on different evidences, i.e., with m evidences $x_1, x_2, \ldots, x_m$ and one mechanism M, what we want is to find a $M(\theta|x_1, x_2, \ldots, x_m) = Combine[M(\theta|x_1), M(\theta|x_2), \ldots, M(\theta|x_m)]$.

Moreover, the above cases may happen jointly too, for which we need to integrate two combining categories coordinately. In a probabilistic form, the above two categories can be formulated as follows:

- *Product rule*: Let $p(\theta|x_1), p(\theta|x_2), \ldots, p(\theta|x_m)$ denote the supports on θ that comes from applying one same mechanism on different evidences $x_1, x_2, \ldots, x_m$, respectively, which can be samples from either the same feature space or different feature spaces. When $x_1, x_2, \ldots, x_m$ are mutually independent, we have the following product combination rule [71]:

$$p(\theta|x_1, x_2, \ldots, x_m) = p(\theta|x_1)p(\theta|x_2)\cdots p(\theta|x_m)/p^{m-1}(\theta). \quad (90)$$

- *Summation rule*: Each $M_j(\theta|x)$ describes a distribution $p(\theta|x, j)$, we want to find $p(\theta|x)$ that combines $p(\theta|x, j)$, $j = 1, \ldots, k$. A best choice for $p(\theta|x)$ is the marginal of the joint distribution $p(j, \theta|x) = p(j|x)p(\theta|x, j)$:

$$p(\theta|x) = \sum_{j=1}^{k} p(j|x)p(\theta|x, j),$$
$$\sum_{j=1}^{k} p(j|x) = 1, \quad 0 \leqslant p(j|x) \leqslant 1, \quad (91)$$

where $p(j|x)$ denotes the proportional role of the jth mechanism in the combination. This is just the mixture-of-experts (ME) by Eq. (50).

Started from 1991, several efforts has been made in Eq. (91) for combining classifiers, which are summarized as follows:

(a) In [72], a special case of Eq. (91) is proposed, where $p(j|x)$ is implemented by a three layer net with sigmoid output learners $0 \leqslant o_j(x, W) \leqslant 1$, $j = 1, \ldots, k$, under the name of a switching controller. This controller is trained via maximizing the likelihood of a multiple Bernoulli $\prod_j o_j(x, W)^{d_j(x)}[1 - o_j(x, W)]^{1-d_j(x)}$ on a set of training pairs, with $d_j(x) = 1$ if the jth classifier classifies the sample x correctly.

(b) In [73], another special case of Eq. (91) is studied, with

$$p(j|x) = q(x|v_j) \Big/ \sum_{i=1}^{k} q(x|v_i),$$
$$q(x|v_j) \geqslant 0 \text{ is a parametric function.} \quad (92)$$

(c) In [60,74], we further extend Eq. (92) by considering each $q(x|v_j)$ from the exponential family subject to a priors α_j, $j = 1, \ldots, k$, i.e., we have

$$p(j|x) = \alpha_j q(x|v_j) \Big/ \sum_{i=1}^{k} \alpha_i q(x|v_i),$$
$$0 \leqslant \alpha_j \leqslant 1, \quad \sum_{j=1}^{k} \alpha_j = 1, \quad (93)$$

which leads to the alternative ME by Eqs. (52) and (51).

The last but not least, it deserves to show that the *mixture using variance* (MUV) approach for ensemble learning [75,76] can be regarded as a degenerated case of Eq. (91). The MUV approach suggests the following combination:

$$\mu(x) = \sum_{j=1}^{k} p_j \mu_j(x), \quad p_j = \sigma_j^{-2} \Big/ \sum_{i=1}^{k} \sigma_i^{-2}, \quad (94)$$

where σ_j^2 is the variance of the estimate $\mu_j(x)$. From $\mu(x) = \int \theta p(\theta|x)\, \mathrm{d}\theta$, $\mu_j(x) = \int \theta p(\theta|x, j)\, \mathrm{d}\theta$, it is not difficult to see that Eq. (94) is actually a degenerated case of Eq. (51), where $p(j|x) = p_j$ is estimated collectively from a set of samples instead of depending on each individual sample x. That is, considering $q(x|v_j) = p(\theta|x, j) = G(\theta|m_j(x), \Sigma_j)$ under the constraint:

$$(\theta - m_1(x))^{\mathrm{T}} \Sigma_1^{-1} (\theta - m_1(x)) = \cdots (\theta - m_k(x))^{\mathrm{T}} \Sigma_k^{-1} (\theta - m_k(x)), \quad (95)$$

it follows from Eq. (92) that $p(j|x) = p_j = |\Sigma_j|^{-0.5} / \sum_{i=1}^{k} |\Sigma_i|^{-0.5}$, which further degenerates to Eq. (94) when $\Sigma_j = \sigma_j^2 I$. Similarly, it follows from Eq. (93) that

$$p(j|x) = p_j = \alpha_j |\Sigma_j|^{-0.5} \Big/ \sum_{i=1}^{k} \alpha_i |\Sigma_i|^{-0.5}. \quad (96)$$

5. Concluding remarks

A general problem solving paradigm has been elaborated to provide not only a unified perspective on two different frameworks for problem solving but also a number of new results on HT and RHT computing, mixture based learning approaches, and evidence combination, for typical pattern recognition tasks that involve data clustering and structure mining, object detection and motion estimation, as well as multi-agent coordinated problem solving. This general paradigm is implemented by an integration of five essential mechanisms, namely, *acquisition, allocation, amalgamation, admission, and affirmation*. Different implementations of these mechanisms and differences in a specific integration may bring us new results and potential directions for future studies. Also, the difference of the two problem solving frameworks studied in this paper comes from their ways of handling two coupled core tasks, namely amalgamating evidences and discriminating differences.

References

[1] J. Illingworth, J. Kittler, A survey of the Hough transform, Comput. Vision Graphics Image Process. 43 (1988) 221–238.

[2] P.V.C. Hough, Method and means for recognizing complex patterns, U.S. Patent 3069654, December 18, 1962.

[3] L. Xu, E. Oja, P. Kultanen, A new curve detection method: randomized Hough transform (RHT), Pattern Recognition Lett. 11 (1990) 331–338.

[4] L. Xu, E. Oja, Randomized Hough transform (RHT): basic mechanisms, algorithms and complexities, Comput. Vision Graphics Image Process.: Image Understanding 57 (1993) 131–154.

[5] O. Chutatape, L. Guo, A modified Hough transform for line detection and its performance, Pattern Recognition 32 (1999) 181–192.

[6] S.M. Kruse, Scene segmentation from dense displacement vector fields using randomized Hough transform, Signal Process.—Image Commun. 9 (1996) 29–41.

[7] T. Behrens, K. Rohr, H.S. Stiehl, Robust segmentation of tubular structures in 3-D medical images by parametric object detection and tracking, IEEE Trans. Syst. Man Cybern. Part B–Cybernetics 33 (2003) 554–561.

[8] J. Heikkonen, Recovering 3-d motion parameters from optical-flow field using randomized Hough transform, Pattern Recognition Lett. 16 (1995) 971–978.

[9] R.A. McLaughlin, Randomized Hough transform: improved ellipse detection with comparison, Pattern Recognition Lett. 19 (1998) 299–305.

[10] Y.X. Chen, F.H. Qi, A new ellipse detection method using randomized Hough transform, J. Infrared Millimeter Waves 19 (2000) 43–47.

[11] A. Imiya, Detection of piecewise-linear signals by the randomized Hough transform, Pattern Recognition Lett. 17 (1996) 771–776.

[12] N. Milisavljevic, Comparison of three methods for shape recognition in the case of mine detection, Pattern Recognition Lett. 20 (1999) 1079–1083.

[13] Q. Ji, Y. Xie, Randomised Hough transform with error propagation for line and circle detection, Pattern Anal. Appl. 6 (2003) 55–64.

[14] L. Xu, A. Krzyzak, E. Oja, Rival penalized competitive learning for clustering analysis, RBF net and curve detection, IEEE Trans. Neural Networks 4 (1993) 636–649.

[15] L. Xu, A. Krzyzak, E. Oja, Unsupervised and supervised classifications by rival penalized competitive learning, in: Proceedings of 11th International Conference on Pattern Recognition, vol. I, Hauge, Netherlands, August 30–September 3, 1992, pp. 672–675.

[16] L. Xu, Rival penalized competitive learning, finite mixture, and multisets clustering, in: Proceedings of IEEE-INNS IJCNN98, Anchorage, Alaska, vol. II, May 5–9, 1998, pp. 2525–2530.

[17] L. Xu, Best harmony, unified RPCL and automated model selection for unsupervised and supervised learning on Gaussian mixtures, ME-RBF models and three-layer nets, Int. J. Neural Syst. 11 (2001) 3–69.

[18] L. Xu, BYY harmony learning, structural RPCL, and topological self-organizing on unsupervised and supervised mixture models, Neural Networks 15 (2002) 1125–1151.

[19] L. Xu, Multisets modeling learning: an unified theory for supervised and unsupervised learning, Invited Talk, in: Proceedings of IEEE ICNN94, vol. I, Orlando, Florida, June 26–July 2, 1994, pp. 315–320.

[20] L. Xu, A unified learning framework: multisets modeling learning, Invited Talk, in: Proceedings of WCNN95, vol. I, Washington, DC, July 17–21, 1995, pp. 35–42.

[21] L. Xu, Data smoothing regularization, multi-sets-learning, and problem solving strategies, Neural Networks 16 (2003) 817–825.

[22] Z.Y. Liu, K.C. Chiu, L. Xu, Strip line detection and thinning by RPCL—based local PCA, Pattern Recognition Lett. 24 (2003) 2335–2344.

[23] Z.Y. Liu, K.C. Chiu, L. Xu, Improved system for object detection and star/galaxy classification via local subspace analysis, Neural Networks 16 (2003) 437–451.

[24] T.M. Nair, et al., Rival penalized competitive learning (RPCL): a topology-determining algorithm for analyzing gene expression data, Comput. Biol. Chem. 27 (2003) 565–574.

[25] P.R. Chang, W.H. Yang, Environment-adaptation mobile radio propagation prediction using radial basis function neural networks, IEEE Trans. Veh. Technol. 46 (1997) 155–160.

[26] S. Nakkrasae, P. Sophatsathit, An RPCL-based indexing approach for software component classification, Int. J. Software Eng. Knowl. Eng. 14 (2004) 497–518.

[27] G. Acciani, et al., A feature extraction unsupervised neural network for an environmental data set, Neural Networks 16 (2003) 427–436.

[28] T. Nakamura, et al, The RoboCup-NAIST: a cheap multisensor-based mobile robot with visual learning capability, in: M. Asada, H. Kitano (Eds.), RoboCup-98, Lecture Notes in Artificial Intelligence, vol. 1604, Springer, Berlin, 1999, pp. 326–337.

[29] G. Castellano, A.M. Fanelli, T. Roselli, Mining categories of learners by a competitive neural network, in: Proceedings of IJCNN01, vol. 2, 2001, pp. 845–950.

[30] P. Avesani, A. Perini, F. Ricci, CBET: a case base exploration tool, in: Proceedings of $AI*IA$97, Roma, Italy, September 16–19, 1997.

[31] H. Furukawa, T. Ueda, M. Kitamura, A systematic method for rational definition of plant diagnostic symptoms by self-organizing neural networks, Neurocomputing 13 (1996) 171–183.

[32] R. Li, et al., Fast image vector quantization using a modified competitive learning neural network approach, Int. J. Imaging Syst. Technol. 8 (1997) 413–418.

[33] A.G. Bors, I. Pitas, Optical flow estimation and moving object segmentation based on median radial basis function network, IEEE Trans. Image Process. 7 (1998) 693–702.

[34] H. Kalviainen, P. Hirvonen, L. Xu, E. Oja, Probabilistic and non-probabilistic Hough Transforms: overview and comparisons, Image Vision Comput. 5 (1995) 239–252.

[35] R.E. Kass, A.E. Raftery, Bayes factors, J. Am. Stat. Assoc. 90 (430) (1995) 773–795.

[36] S. Grossberg, Competitive learning: from iterative activation to adaptive resonance, Cognitive Sci. 11 (1987) 23–63.

[37] S.C. Ahalt, et al., Competitive learning algorithms for vector quantization, Neural Networks 3 (1990) 277–291.

[38] D. Desieno, Adding a conscience to competitive learning, in: Proceedings of IEEE International Conference on Neural Networks, vol. I, 1988, pp. 117–124.

[39] D.E. Rumelhart, D. Zipser, Feature discovery by competitive learning, Cognitive Sci. 9 (1985) 75–112.

[40] R. Hecht-Nielsen, Counterpropagation networks, Appl. Opt. 26 (1987) 4979–4984.

[41] T. Kohonen, Self-organized formation of topologically correct feature maps, Biol. Cybern. 43 (1982) 59–69.

[42] L. Xu, Bayesian–Kullback Ying–Yang learning scheme: reviews and new results, in: Proceedings of ICONIP96, vol. 1, Hong Kong, September 24–27, 1996, pp. 59–67.

[43] L. Xu, An overview on unsupervised learning from data mining perspective, in: N. Allison et al. (Eds.), Advances in Self-Organizing Maps, Springer, Berlin, 2001, pp. 181–210.

[44] L. Xu, Bayesian–Kullback coupled Ying–Yang machines: unified learnings and new results on vector quantization, in: Proceedings of ICONIP95, Beijing, October 30–November 3, 1995, pp. 977–988.

[45] L. Xu, Independent component analysis and extensions with noise and time: a Bayesian Ying–Yang learning perspective, Neural Inf. Process. Lett. Rev. 1 (2003) 1–52.

[46] L. Xu, Temporal BYY encoding, Markovian state spaces, and space dimension determination, IEEE Trans. Neural Networks 15 (2004) 1276–1295.

[47] L. Xu, Advances on BYY harmony learning: information theoretic perspective, generalized projection geometry, and independent factor auto-determination, IEEE Trans. Neural Networks 15 (2004) 885–902.

[48] L. Xu, Bayesian Ying Yang learning: (I) a unified perspective for statistical modeling, (II) a new mechanism for model selection and regularization, in: N. Zhong, J. Liu (Eds.), Intelligent Technologies for Information Analysis, Springer, Berlin, 2004, pp. 613–697.

[49] R.A. Redner, H.F. Walker, Mixture densities, maximum likelihood, and the EM algorithm, SIAM Rev. 26 (1984) 195–239.

[50] D. Mackey, A practical Bayesian framework for backpropagation, Neural Comput. 4 (1992) 448–472.

[51] H. Akaike, A new look at the statistical model identification, IEEE Trans. Autom. Control 19 (1974) 714–723.

[52] H. Bozdogan, Model selection and Akaike's information criterion: the general theory and its analytical extension, Psychometrika 52 (1987) 345–370.

[53] G. Schwarz, Estimating the dimension of a model, Ann. Stat. 6 (1978) 461–464.

[54] J. Rissanen, Stochastic Complexity in Statistical Inquiry, World Scientific, Singapore, 1989.

[55] L. Xu, E. Oja, C.Y. Suen, Modified hebbian learning for curve and surface fitting, Neural Networks 5 (1992) 393–407.

[56] T.W. Anderson, H. Rubin, Statistical inference in factor analysis, Proceedings of the Third Berkeley Symposium Mathematical Statistics and Probability. vol. 5, UC Berkeley, 1956, pp. 111–150.

[57] R. McDonald, Factor Analysis and Related Techniques, Lawrence Erlbaum, London, 1985.

[58] Z.Y. Liu, H. Qiao, L. Xu, Multisets mixture learning based ellipse detection, Pattern Recognition 39 (2006) 731–735.

[59] R.A. Jacobs, et al., Adaptive mixtures of local experts, Neural Comput. 3 (1991) 79–87.

[60] L. Xu, M.I. Jordan, G.E. Hinton, An alternative model for mixtures of experts, in: J. Cowan, G. Tesauro, J. Alspector (Eds.), Advances in Neural Information Processing Systems, vol. 7, MIT Press, Cambridge, MA, 1995, pp. 633–640.

[61] L. Xu, A trend on regularization and model selection in statistical learning: a perspective from bayesian ying yang learning, in: W. Duch et al. (Eds.), Challenges to Computational Intelligence, Springer, Berlin, 2007, in press.

[62] L. Xu, Fundamentals, challenges, and advances of statistical learning for knowledge discovery and problem solving: a BYY harmony perspective, keynote talk, in: Proceedings of International Conference on Neural Networks and Brain, vol. 1, Beijing, China, October 13–15, 2005, pp. 24–55.

[63] L. Shi, L. Xu, Local factor analysis with automatic model selection: a comparative study and digits recognition application, in: Artificial Neural Networks—ICANN 2006, Lecture Notes in Computer Applications, vol. 4132, Springer, Berlin, 2006, pp. 260–269.

[64] L. Shi, L. Xu, Comparative investigation on dimension reduction and regression in threelayer feed-forward neural network, in: Artificial Neural Networks—ICANN 2006, Lecture Notes in Computer Applications, vol. 4131, Springer, Berlin, 2006, pp. 51–60.

[65] L. Xu, BYY harmony learning independent state space and generalized APT financial analyses, IEEE Trans. Neural Networks 12 (2001) 822–849.
[66] L. Xu, Temporal BYY learning for state space approach, hidden markov model and blind source separation, IEEE Trans. on Signal Processing 48 (2000) 2132–2144.
[67] L. Rade, B. Westergren, S. Lund, Mathematics Handbook for Science and Engineering, Birkhauser, Basel, 1995.
[68] L. Xu, BYY learning, regularized implementation, and model selection on modular networks with one hidden layer of binary units, Neurocomputing 51 (2003) 227–301.
[69] J. Ma, T. Wang, L. Xu, A gradient BYY harmony learning rule on Gaussian mixture with automated model selection, Neurocomputing 56 (2004) 481–487.
[70] K.X. Chiu, L. Xu, Arbitrage pricing theory based Gaussian temporal factor analysis for adaptive portfolio management, Decision Support Syst. 37 (2004) 485–500.
[71] J. Kittler, et al., On combining classifiers, IEEE Trans. Pattern Anal. Mach. Intell. 20 (3) (1998) 226–239.
[72] L. Xu, A. Krzyzak, C.Y. Sun, Associative switch for combining multiple classifiers, in: Proceedings of IJCNN91, vol. I, Seattle, WA, July 8–12, 1991, pp. 43–48.
[73] L. Xu, M.I. Jordan, EM learning on a generalized finite mixture model for combining multiple classifiers, Proceedings of WCNN93, Portland, OR, vol. IV, July 11–15, 1993, pp. 227–230.
[74] L. Xu, M.I. Jordan, G.E. Hinton, A modified gating network for the mixtures of experts architecture, in: Proceedings of WCNN94, vol. 2, San Diego, CA, June 4–9, 1994, pp. 405–410.
[75] M. Perrone, L. Cooper, When networks disagree: ensemble methods for hybrid neural networks, in: R.J. Mammone (Ed.), Neural Networks for Speech and Image Processing, Chapman & Hall, London, 1993.
[76] T. Dietterich, Machine learning research: four current directions, Arif. Intell. Magazine 18 (4) (1997) 97–136.
[77] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[78] X.L. Hu, L. Xu, Investigation on several model selection criteria for determining the number of cluster, Neural Inf. Process.—Lett. Rev. 4 (2004) 1–10.
[79] X.L. Hu, L. Xu, A comparative study of several cluster number selection criteria, in: Proceedings of IDEAL03, Lecture Notes in Computer Science, vol. 2690, Springer, Berlin, 2003, pp. 195–202.
[80] L. Xu, RBF nets, mixture experts, and Bayesian Ying–Yang learning, Neurocomputing 19 (1998) 223–257.
[81] L. Xu, M.I. Jordan, On convergence properties of the EM algorithm for Gaussian mixtures, Neural Comput. 8 (1) (1996) 129–151.
[82] L. Xu, A. Krzyzak, C.Y. Sun, Several methods for combining multiple classifiers and their applications in handwritten character recognition, IEEE Trans. System Man Cybern. 22 (1992) 418–435.

About the Author—LEI XU (IEEE Fellow and IAPR Fellow), a professor of Chinese University of Hong Kong where he joined in 1993. He completed his Ph.D. Thesis at Tsinghua University in 1986, and worked at several universities during 1987–1993, including Peking University, Harvard and MIT. Prof. Xu has published a number of well-cited papers in the literatures of neural networks, statistical learning, and pattern recognition (e.g., his ten most frequently cited papers scored 1000 citations according to SCI-Expended). He gave keynote/plenary/invited/tutorial talks in international conferences (IJCNN, WCNN, ICONIP, ICNN, etc.), served or has been serving as associate editor for several international journals, a governor of International neural network society (01–03), a past president of Asian-Pacific neural networks assembly (APNNA), and also a member of engineering panel on several Chinese and HK research funding committees, as well as a nominator for the prestigious Kyoto prize (2003–2004). Prof. Xu has received several Chinese national prestigious academic awards (including 1993 National Nature Science Award) and international awards (including 1995 INNS Leadership Award). Recently, he has received the 2006 APNNA Outstanding Achievement Award. He is an IEEE Fellow (2001–) and a Fellow of International Association for Pattern Recognition (2002–), and an academician of European Academy of Sciences (2002–). Prof. Xu is also serving as a member of Fellow committee of IEEE Computational Intelligence Society (2006–).