PROCEEDINGS OF

VOLME 2
OF 2

INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING (ICONIP'95)

Oct.30 ~Nov.3,1995

Beijing, China

Co-Sponsored by:

Asia–Pacific Neural Networks Assembly(APNNA)

IEEE Region 10

Technical Co-Sponsored by:

IEEE Communication Society

In Cooperation with:

International Neural Networks Society

European Neural Networks Society

Russian Neural Networks Society

Supported by:

National Natural Science Foundation of China

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

YING-YANG Machine: A Bayesian-Kullback Scheme for Unified Learnings and New Results on Vector Quantization[1]

Lei Xu
1. Dept of Computer Science, The Chinese University of Hong Kong, Hong Kong
2. National Machine Perception Lab, Peking Univ, Beijing

Abstract *A Bayesian-Kullback learning scheme, called Ying-Yang Machine, is proposed based on two complement but equivalent Bayesian representations for joint density and on the iterative Alternative Minimization of the Kullback divergence between the representations. We show that the scheme unifies the EM & em algorithm, multisets modeling, Helmholtz machine, maximum information preservation and other major learning schemes, and also provides new learning schemes and new views. Furthermore, a particular case of the scheme is investigated in depth. We obtain the cluster number selection criteria for the conventional Vector Quantization (VQ) and its algorithms. Moreover, Stochastic VQ and Maximum Posterior VQ are suggested and shown to be superior to the conventional VQ. They are implemented by the EM algorithm and its hard-cut variant respectively, with the appropriate cluster (code-vector) number automatically selected. In addition, an adaptive EM algorithm is proposed, which not only justifies current major competitive learning (CL) algorithms by including them as special cases, but also improves these algorithms in several aspects.*

1. Introduction

Most of the learning problems can be summarized into the estimation of the joint distribution $P(x,y)$ with x from the input space and y from the representation space. By Bayesian formulation, $P(x,y)$ has two complement but equivalent representations $P_{M_2}(x,y) = P_{M_2}(y)P_{M_2}(x|y)$ and $P_{M_1}(x,y) = P_{M_1}(y|x)P_{M_1}(x)$. The two representations are modeled by M_2 and M_1, called YING machine and YANG machine respectively with YING and YANG being two chinese characters for words "female" and "male" respectively. The Kullback divergence $KL(P_{M_1}(x,y), P_{M_2}(x,y))$ is iteratively minimized by alternatively fixing M_1 and M_2 to get all the remaining unknows. We call the *Bayesian-Kullback* learning scheme as *YING-YANG Machine.* Instead of merely using Kullback divergence for matching joint densities in information geometry type alternating minimization learnings (Amari, 1995a&b; Byrne, 1992; Csiszar, 1975&84), the *YING-YANG Machine* combines Bayesian formulation into the Kullback divergence for matching two specific joint densities in the complement but equivalent Bayesian representations.

We show that the YING-YANG machine unifies the existing major learning approaches. One special case reduces to the EM algorithm proposed by Dempster et al(1977), to a cost function by Hathaway (1986) and Neal & Hinton (1993), to the *Information Geometry* theory and the *em* algorithm (Amari, 1995a&b; Byrne, 1992; Csiszar, 1975&84), to MDL autoencoder with a "bits-back" argument by Hinton & Zemel (1994) and the autoencoder of minimizing the bits of uncoded residual errors and the unused bits in the transmission channel's capacity (Xu, 1995). The special case can also reduces to multisets modeling learning (Xu, 1995)–a unified learning framework for several existing major unsupervised and supervised learnings. One other special case of the YING-YANG machine reduces to maximum information preservation (Linsker, 1989; Atick & Redlich, 1990; Bell & Sejnowski, 1995). Another special case of the YING-YANG machine reduces to Helmholtz machine (Dayan et al,1994; Hinton, 1994), and provides a new perspective to the popular model. Moreover, the YING-YANG machine includes also maximum likelihood or least square learning as special cases, and provide new learning schemes for further studies.

We further study one special case of the YING-YANG machine in depth with several new results obtained for Vector Quantization (VQ). A cluster number selection criteria has been discovered for the conventional Vector Quantization (VQ) and its algorithms, which provides a solution for an open problem that remains unsolved for decades. The Stochastic VQ (SVQ) and Maximum Posterior VQ (MAP VQ) are presented and shown to be superior to the conventional VQ. They are implemented by the EM algorithm and its hard-cut variant respectively, with the appropriate

[1] I am grateful to G. E. Hinton and P. Dayan for discussions which give me good understanding on Helmholtz machine, to M.I.Jordan and his group for helpful comments on the early version of this paper. The work was Supported by the HK RGC Earmarked Grant CUHK250/94E.

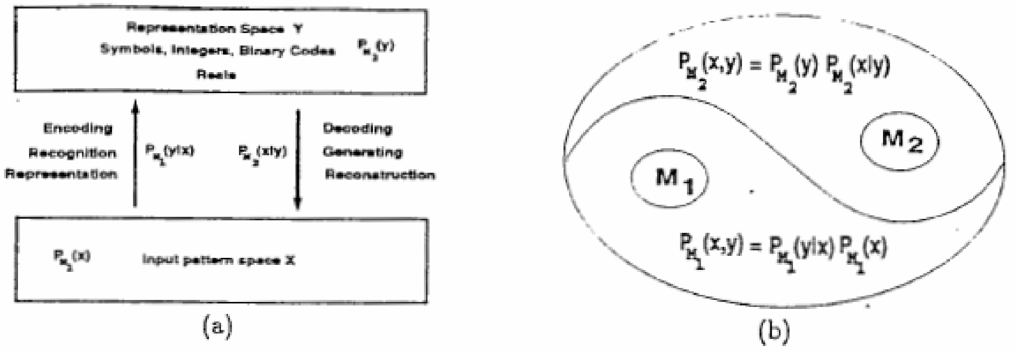

Figure 1: (a) The input space X and the representation space Y. (b) The YING-YANG Machine.

cluster (code-vector) number automatically discovered during the learning. In addition, we propose an adaptive EM algorithm, which not only justifies, by including as special cases, the existing major CL algorithms, such as the conventional CL, *conscience* or *frequency sensitive* CLs (Desieno, 1988; Ahalt at al, 1990) and RPCL (Xu, Krzyzak& Oja, 1993), but also improve these algorithms in several aspects.

2. YING-YANG Machine: A Bayesian-Kullback Learning Scheme

2.1 Learning as joint distribution estimation

Most of the learning problems can be summarized into the problem of setting up the descriptions and their connections for patterns in the input space X and the representation space Y. An appropriate formulization for the purpose is the joint distribution $P(x, y)$, $x \in X$, $y \in Y$ from which we can have all the four components $P(x)$, $P(y)$, $P(y|x)$ and $P(x|y)$. $P(x)$ and $P(y)$ describe the input distribution and its representation in Y. From $P(y|x)$, we have the mapping $X \to Y$. As shown in Fig.1(a), given a x, we get, according to the probability $P(y|x)$, a $y \in Y$ as a label, a code or a representation. E.g., we can randomly choose an y according to $P(y|x)$, this is usually called *stochastic* coding, labeling or recognition. We can also get $y = arg\ max_y P(y|x)$, called as *maximum posteriori* coding, labeling or recognition. Moreover, we can even use $y = E_{P(y|x)}[y]$, called as *regression*. Hence, $P(y|x)$ defines a *Recognition/Representation* model, or R model shortly. On the other hand, from $P(x|y)$ we have the mapping $Y \to X$. As shown in Fig.1(a), given a y, we can get a $x \in X$ as a decoded or reconstructed pattern from y; or in other words, x is generated from y. Similarly, we can randomly choose an x according to $P(x|y)$, or let $x = arg\ max_x P(x|y)$, or $x = E_{P(x|y)}[x]$. That is, $P(x|y)$ defines a *Generative* model, or G model shortly.

The purpose of learning is to build up $P(x)$, $P(y)$, $P(y|x)$ and $P(x|y)$ or equivalently $P(x, y)$, based on training samples from X or from both X and Y. Given a training set $D_x = \{x_i\}_{i=1}^N$, we can find an empirical $P(x)$. If we further specify the structure of Y (e.g., integer, binary, independent binary bits) as well as the structure of $P(x|y)$ or $P(y|x)$, the extension from $P(x)$ to $P(x, y)$ will not arbitrary and should be consistent to these specified structures. The set of $P(y|x), P(x|y), P(y), P(x)$ that satisfies D_x and are bestly consistent to these constraints as well as the Bayesian law $P(y|x)P(x) = P(x|y)P(y)$ provides the solution of the learning. For particular structures of Y, by $P(y|x)$ we may have clustering, PCA, coding, etc. Moreover, by $P(x|y)$ we are also able to generate a new member x from Y. We call all these learnings as *Unsupervised learning*. Given a training set $D_{x,y} = \{x_i, y_i\}_{i=1}^N$, i.e., we are taught to pair x_i, y_i for a set of samples, the task of building up $P(y|x)$ to be bestly consistent to this teaching is usually referred as *Supervised Learning*. It is a subtask of seeking the densities $P(y|x), P(x|y), P(y), P(x)$, which satisfy $D_{x,y}$ and are bestly consistent to the specified structural constraints and the equality constraint $P(y|x)P(x) = P(x|y)P(y)$. This duty is similar to the duty of unsupervised learning.

In a summary, *all the learning problems can be considered as the problem of joint distribution estimation*, and the difference of *supervised learning* from *unsupervised learning* lies in that more constraints are added.

2.2 YING-YANG Machine That Bases on Bayesian-Kullback Scheme

Under the Bayesian framework, we have two representations for $P(x,y)$. As shown in Fig.1(b). One is $P_{M_1}(x,y) = P_{M_1}(y|x)P_{M_1}(x)$, implemented by a R-model M_1, which we call *Yang*/(male) machine since it performs the task of transferring a pattern/(a real body) into a code/(a seed). The other is $P_{M_2}(x,y) = P_{M_2}(x|y)P_{M_2}(y)$, implemented by a G-model M_2, which we call *Ying* machine since it performs the task of generating a pattern/(a real body) from a code/(a seed). They are complement to each other and together implement an entire circle $x \to y \to x$. Such a scheme shows a compliment to a famous chinese ancient philosophy— YING-YANG theory, represented by a sign similar to that given in Fig. 1(b). The theory believes that every thing in universe consists of YING part and YANG part and that the match or coherence of the two parts makes every thing as well as the entire universe normal.

To make our scheme normally perform certain learning based tasks, we design and then match or couple the YANG machine and YING machine by appropriately setting up $P_{M_1}(x)$, $P_{M_1}(y|x)$, $P_{M_2}(x|y)$ and $P_{M_2}(y)$, based on a given training set for the tasks. This procedure consists of two stages—*Model Design* and *Parameter Estimation*.

On the **Model Design Stage**, we need to design the forms of the density functions and the corresponding implementing architectures for the four components. Each of them should be subject to three types of constraints:

(1) *Free*. The component is free to take any member of the family consisting of all the possible densities of the same type as the component. E.g, if $P_{M_1}(y|x)$ is free, then it can be any member of the density family $\{P(y|x)\}$ consisting of all the possible $P(y|x)$. In this case, no architecture is specified.

(2) *Parametric*. The component must be a member of the given parametric family of densities of the same type as members. E.g., if $P_{M_1}(y|x)$ is parametric, then it must be a member of the parametric density family $\{P(y|x,\theta)\}$ with its function form fixed but θ taking all the possible values. The particular interesting family of $\{P(y|x,\theta)\}$ for recognition purposed representation can be one of the following ones:

$$P_{M_1}(y|x,\theta), with\ y = 1,\cdots,k,\ or\ binary\ y = [y_1,\cdots,y_k],$$
$$P_{M_1}(y|x) = \textstyle\prod_{j=1}^{k} p(y_j|x)^{y_j}(1-p(y_j|x))^{1-y_j},$$
$$P_{M_1}(y|x,\theta) = \textstyle\prod_{j=1}^{k} p(y_j|x,\theta^{(j)})^{y_j}(1-p(y_j|x,\theta^{(j)}))^{1-y_j}. \tag{1}$$

Where eq.(1) is called factorial coding, preferred by Hinton et al (1994). The architectures for implementing them are free as long as they are mapping networks from x to discrete or binary y.

An interesting family of $\{P(y|x,\theta)\}$ for regression purposed representation is

$$P_{M_1}(y|x,\theta) = (2\pi)^{-d/2}|\Sigma_1|^{-1/2}e^{-\frac{1}{2}(y-f(x,W_1))^T\Sigma_1^{-1}(y-f(x,W_1))}, \qquad \theta = \{W_1,\Sigma_1\}, \tag{2}$$

with $E(y|x,\theta) = f(x,W_1)$ representing a feedforward network for the R-model.

Another interesting but more complicated family for $\{P(y|x,\theta)\}$ is the conditional mixture (Xu, Jordan & Hinton, 1994):

$$P_{M_1}(y|x,\theta) = \textstyle\sum_{j=1}^{K} g_j(x,\nu)(2\pi)^{-d/2}|\Sigma_{1j}|^{-1/2}e^{-\frac{1}{2}(y-f_j(x,W_{1,j}))^T\Sigma_{1,j}^{-1}(y-f_j(x,W_{1,j}))}, \tag{3}$$

where θ consists of all the $\nu, W_{1,j}, \Sigma_{1j}$. Each $f_j(x,W_{1,j})$ is a feedforward network. The nonnegative $g_j(x,\nu), j = 1,\cdots,K$ forms an output vector of a *gating network* with the weight set ν such that $\sum_{j=1}^{K} g_j(x,\nu) = 1$. In other words, a *mixtures of experts* model (Xu, Jordan & Hinton, 1994) is used as a whole for performing the R-model.

While a particular interesting family for the generative $\{P(x|y,\psi)\}$ is

$$P_{M_2}(x|y,\psi) = (2\pi)^{-d/2}|\Sigma_2|^{-1/2}e^{-\frac{1}{2}(x-g(y,W_2))^T\Sigma_2^{-1}(x-g(y,W_2))}, \qquad \psi = \{W_2,\Sigma_2\}, \tag{4}$$

with $E(x|y,\psi) = g(y,W_2)$ representing a backward network for implementing a G-model.

(3) *Fixed*. The component has been fixed at a particular empirical density, obtained by non-parametric Parzen window estimate from a training set. We say that $P_{M_1}(x)$ is fixed, it means $P_{M_1}(x) = P_h(x)$ or $P_{M_1}(x) = P_0(x)$:

$$P_h(x) = \tfrac{1}{Nh^d}\textstyle\sum_{i=1}^{N} K(\tfrac{x-x_i}{h}), \quad P_0(x) = \lim_{h\to 0} P_h(x) = \tfrac{1}{N}\textstyle\sum_{i=1}^{N}\delta(x-x_i). \tag{5}$$

where d is the dimension of x, $K(r)$ is a kernel statisfying some condition (sec. 2.2, Xu, Krzyzak & Yuille, 1994), and h is called smooth parameter.

The Parzen window estimate for $P_h(x,y)$ and $P_h(y)$ are obtained via replacing x in eq.(5) by $z = [x, y]$ and y respectively.

How each of the four components $P_{M_1}(x)$, $P_{M_1}(y|x)$, $P_{M_2}(x|y)$ and $P_{M_2}(y)$ is confined to each of the above three types of constraints is based on a particular design. Different designs are suitable for different learning tasks and also provide different models even for a same learning task. In general, We have the following situations:

(1) For *Unsupervised learning* tasks, we fix that $P_{M_1}(x) = P_h(x)$ or $P_{M_1}(x) = P_0(x)$. For the other components, we can have three types of designs:

(a) Backward G-Net. $P_{M_1}(y|x)$ is free and $P_{M_2}(x|y,\psi)$ is parametric (i.e., a backward net directly implements G-model). Depending on whether y is discrete or real, we have three variants:

(i) G-Net-I : $P_{M_2}(y)$ is free in the finite discrete family $\{P_{M_2}(y), y = 1, \cdots, k\}$.

(ii) G-Net-II : $P_{M_2}(y)$ is free for real y.

(iii) G-Net-III : $P_{M_2}(y)$ is in a parametric family $P_{M_2}(y|\psi_y)$ for real y.

(b) Forward R-Net. $P_{M_1}(y|x,\theta)$ is parametric (i.e., a feedforward net directly implements R-model). There may be two useful variants:

(i) R-Net-I : $P_{M_2}(y)$ is free and it is forced that $P_{M_2}(x|y) = P_h(x)$ or $P_{M_2}(x|y) = P_0(x)$.

(ii) R-Net-II : $P_{M_2}(x|y)$ is free, and $P_{M_2}(y)$ is in a parametric family $P_{M_2}(y|\psi_y)$ for real y.

The variant that correspond to the free $P_{M_2}(x|y)$ and $P_{M_2}(y)$ is a useless under-determined model.

(c) Forward-Backward RG-Net. Both $P_{M_1}(y|x,\theta)$ and $P_{M_2}(x|y,\psi)$ are parametric; i.e., a feedforward net directly implements R-model, and a backward net directly implements G-model. Depending on whether y is discrete or real, we have three variants:

(i) RG-Net-I : $P_{M_2}(y)$ is free in the finite discrete family $\{P_{M_2}(y), y = 1, \cdots, k\}$.

(ii) RG-Net-II : $P_{M_2}(y)$ is free for real y.

(iii) RG-Net-III : $P_{M_2}(y)$ is in a parametric family $P_{M_2}(y|\psi_y)$ for real y.

(2) For *Supervised learning* tasks, we can have empirical estimates $P_h(x)$, $P_h(y)$ and $P_h(x,y)$. We can use any pair of them to fix two of the four components. In total, we can have also three types of designs:

(a) Backward G-Net. $P_{M_1}(x,y) = P_h(x,y)$ or $P_{M_1}(x,y) = P_0(x,y)$ is fixed, $P_{M_2}(x|y,\psi)$ is parametric, and $P_{M_2}(y) = P_h(y)$ or $P_{M_2}(y) = P_0(y)$ is fixed.

(b) Forward R-Net. $P_{M_1}(x) = P_h(x)$ or $P_{M_1}(x) = P_0(x)$ is fixed, $P_{M_1}(y|x,\theta)$ is parametric, and $P_{M_2}(x,y) = P_h(x,y)$ or $P_{M_2}(x,y) = P_0(x,y)$ is fixed.

(c) Forward-Backward RG-Net. $P_{M_1}(x) = P_h(x)$ or $P_{M_1}(x) = P_0(x)$ is fixed, $P_{M_1}(y|x,\theta)$ is parametric, $P_{M_2}(x|y,\psi)$ is parametric, and $P_{M_2}(y) = P_h(y)$ or $P_{M_2}(y) = P_0(y)$ is fixed.

On the **Parameter Estimation Stage**, we estimate the unknown parameters to match or couple the above designed YING machine and YANG machine, which is based on an important constraint for $P_{M_1}(x)$, $P_{M_1}(y|x)$, $P_{M_2}(x|y)$ and $P_{M_2}(y)$— the Bayesian law:

$$P_{M_1}(y|x)P_{M_1}(x) = P_{M_1}(x,y) = P_{M_2}(x,y) = P_{M_2}(x|y)P_{M_2}(y). \tag{6}$$

However, the direct solving based on eq.(6) has two disadvantages. First, eq.(6) gives nonlinear equations which may be difficult to be solved explicitly. Second, for some restrictive cases like Forward-Backward RG-Net, we may be not able to find a solution to exactly satisfy the equation. We need to relax the tight satisfaction of eq.(6) by requesting that the two representations should be as close as possible. With these considerations, we minimize the Kullback divergence[2]

$$KL(M_1, M_2) = KL(P_{M_1}(x,y), P_{M_2}(x,y)) = \int_{x,y} P_{M_1}(y|x)P_{M_1}(x) \log \frac{P_{M_1}(y|x)P_{M_1}(x)}{P_{M_2}(x|y)P_{M_2}(y)} dxdy \tag{7}$$

to let M_2 to couple M_1 as close as possible such that M_1, M_2 are coordinately decided. Obviously, eq.(7) includes eq.(6) as a special case that $KL(M_1, M_2) = 0$.

The minimization of $KL(M_1, M_2)$ can be made by *Alternative Minimization* (ALTMIN):

Step 1 Fix $M_2 = M_2^{old}$, to get $M_1^{new} = arg\ Min_{M_1}\ KL(M_1, M_2^{old})$ or equivalently $M_1^{new} = Min_{M_1} \int_{x,y} P_{M_1}(y|x)P_{M_1}(x) \log P_{M_1}(y|x)P_{M_1}(x) dxdy$. Or simply find a M_1^{new} such that $KL(M_1^{new}, M_2^{old}) \le KL(M_1^{old}, M_2^{old})$ and then we update $M_1^{old} = M_1^{new}$.

Step 2 Fix $M_1 = M_1^{old}$, to get $M_2^{new} = arg\ Min_{M_2}\ KL(M_1^{old}, M_2)$ or simply find a M_2^{new} such that $KL(M_1^{old}, M_2^{new}) \le KL(M_1^{old}, M_2^{old})$

[2]Since $KL(M_1, M_2) \neq KL(M_2, M_1)$, we can also have variants $KL(M_2, M_1)$ or $KL(M_1, M_2) + KL(M_2, M_1)$, which can be investigated similarly by following the line we study $KL(M_1, M_2)$ in the rest of the paper.

The ALTMIN iteration will finally converge to a local minimum of $KL(M_1, M_2)$ since the both steps reduce or at least not increases $KL(M_1, M_2)$. We call a pair $\{M_1, M_2\}$ obtained by such a procedure as *Bayesian-Kullback Coupled YING-YANG Machine* or YING-YANG machine shortly.

It deserves to mention that the YING-YANG machine is related to the information geometry type learnings (Amari, 1995a&b; Byrne, 1992; Csiszar, 1975&84) which also use the Kullback divergence for matching two joint densities (i.e., the density from data submanifold and the density from model submanifold) via alternating minimization. However, those learnings only consider the densities on the space of all the variables without going deep into the structure of the space. The YING-YANG Machine considers the joint space of the input space and the representation space via two complement but equivalent Bayesian formulations implemented by two complement structures—generative model and recognition model. The two formulations are coupled through the Kullback divergence. In other words, instead of only using the Kullback divergence for two joint densities, the *YING-YANG Machine* is a combination of Bayesian formulation and Kullback divergence for matching two structural joint densities, with their four components specifically designed as stated above. This difference is significant and makes it possible that the YING-YANG Machine can act as a very general learning scheme for unifying the exisiting major unsupervised and supervised learning schemes for neural networks.

3. Unified Learnings by YING-YANG Machine

3.1 Unsupervised Backward G-Net and The EM Related Learning Schemes

For the G-Net-I, let $P_{M_1}(x) = P_0(x)$ as given by eq.(5) and put it into eq.(7), through certain mathematics we can get

$$K(M_1, M_2) = H - Q + D, \quad Q = \frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{k} P_{M_1}(y|x_i)\log[P_{M_2}(x_i|y,\psi)P_{M_2}(y)]$$
$$H = \frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{k} P_{M_1}(y|x_i)\log P_{M_1}(y|x_i), \quad D = -\lim_{h\to 0}\log Nh^d. \tag{8}$$

Since D is independent of M_1, M_2, the minimization of $K(M_1, M_2)$ is equivalent to minimize $H - Q$. Since $P_{M_1}(y|x)$ and $P_{M_2}(y)$ are free, the Step 1 of the ALTMIN scheme can be explicit solved. As a result, the ALTMIN scheme now becomes:

Step 1 (E-step) Compute $P_{M_1}(y|x_i) = P_{M_2}(x_i|y,\psi^{old})P^{old}_{M_2}(y)/\sum_{y=1}^{k} P_{M_2}(x_i|y,\psi^{old})P^{old}_{M_2}(y)$,

$$P^{new}_{M_2}(y) = \frac{1}{N}\sum_{i=1}^{N} P_{M_1}(y|x_i), \quad Q(\psi|\psi^{old}) = \frac{1}{N}\sum_{i=1}^{N} E_{P_{M_1}(y|x_i)}[\log P_{M_2}(x_i|y,\psi)]. \tag{9}$$

Step 2 (M-step) $\psi^{new} = arg\ max_{\psi}\ Q(\psi|\psi^{old})$ or find a ψ^{new} such that $Q(\psi^{new}|\psi^{old}) \geq Q(\psi^{old}|\psi^{old})$.

This is exactly the EM algorithm proposed by Dempster et al(1977). They showed via incomplete data theory that the EM algorithm gives the maximum likelihood estimation of mixture model. Here we can have a very easy proof for this point. From eq.(9), we have $P_{M_2}(x|y,\psi)P_{M_2}(y) = P_{M_2}(x|\psi)P_{M_1}(y|x)$ with

$$P_{M_2}(x|\psi) = \sum_{y=1}^{k} P_{M_2}(x|y,\psi)P_{M_2}(y), \tag{10}$$

being the companion representation for M_2 in the form of finite mixture(Xu & Jordan, in press). Putting these into $K(M_1, M_2)$, we can obtain $K(M_1, M_2) = -L + D$ with D being the same as given in eq.(8) and

$$L = \frac{1}{N}\sum_{i=1}^{N}\int_y P_{M_1}(y|x_i)\log P_{M_2}(x_i|\psi)dy = \frac{1}{N}\sum_{i=1}^{N}\log P_{M_2}(x_i|\psi)$$

It indicates that $Min\ K(M_1, M_2)$ is equivalent to maximum likelihood estimation of mixture model. So we here get a new perspective for the EM algorithm.

Also, $H - Q$ in eq.(8) is the same as the $D(W, \theta^r)$ function given in Hathaway (1986) and the $-F(P,\theta)$ function given by Neal & Hinton (1993), which was interpreten as the "free energy". The same F function has also been obtained by Hinton & Zemel (1994) via using Minimum Description Length (MDL) with a "bits-back" argument for building an autoencoder, as well as by Xu (1995) via minimizing the bits of encoding errors and the unused bits of the transmission channel's capacity during the communication[3]. Although the two encoding schemes give the same solution, the scheme by Xu (1995) is for on-line encoding and can avoid the "bits-back" argument and its related unnatural assumption " the receiver waits until all of the training vectors have been

[3] Moreover, the term D actually represents the cost by the quantization accuracy.

communicated losslessly". Here, we get a new interpretation—-$F = H - Q$ *is the major part of the Kullback divergence* $KL(M_1, M_2)$ *in the case* $P_{M_1}(x) = P_0(x)$.

In this case, the representation $P_{M_1}(x, y)$ is also equivalent to the data submanifold $\mathcal{D}$ in the *Information Geometry* theory and the *em* algorithm proposed by (Amari, 1995a&b), and the representation $P_{M_2}(x, y)$ is equivalent to his submanifold $\mathcal{M}$ of an exponential family when $P_{M_2}(x, y)$ belongs to an exponential family. In other words, the ALTMIN scheme for minimizing $KL(M_1, M_2)$ shares the same idea of alternatively mapping between $\mathcal{D}$ and $\mathcal{M}$ for the *Information Geometry* theory and the *em* algorithm. As a result, this specific YING-YANG machine provides a new easy-understandable perspective and a new implementation for Amari's *Information Geometry* theory. There is no need for the concepts of differential geometry, e-geodesic, m-geodesic, and the alternative minimization is not treated in sufficient statistics and limited in the exponential family. Instead, the specific YING-YANG machine is for any density family, although the implementation will be much simpler when $P_{M_2}(x, y)$ belongs to an exponential family. Moreover, the general YING-YANG machine can provide several new extensions for the existing *Information Geometry* scheme. For example, when $h \neq 0$, $P_{M_1}(y|x)$ and $P_h(x)$ together define a new data submanifold $\mathcal{D}$, which is more smooth and thus the learned model will be better in generalization. Furthermore, the cases of G-NET-II, G-NET-III, the two R-Nets, as well as the three RG-nets described in Sec.2.2 can all be regarded as new developments from the existing *Information Geometry* scheme.

Moreover, the above discussed also indicates that *the EM algorithm and the em algorithm is actually equivalent.* In fact, the EM algorithm defined by Amari(1995a&b), which is based on the sufficient statistics of the exponential family, is slightly different from the conventional EM algorithm. Its E step (and thus M step also) is not equivalent to the above eq.(9) (and the M step). Amari has shown that his EM algorithm is equivalent to his *em* algorithm (thus the conventional EM algorithm) in most cases and also asymptotically, although not always(Amari, 1995b).

Finally, we mention two interesting issues. First, we can use the factorial distribution eq.(1) for $P_{M_1}(y|x)$ in eq.(8) and the ALTMIN scheme in order to get the factorial recognition code in a way similar to Hinton and Zemel (1994). Second, the above specific YING-YANG machine only provides us with the G-model $P_{M_2}(x|y, \psi)$ and a set of samples $P_{M_1}(y|x_i)$ for R-model. This is enough for clustering purpose only. However, for encoding or recognition on a testing data set, we can get $P_{M_2}(x|\psi)$ by the following extrapolating formula:

$$P_{M_2}(x|\psi) = \sum_{y=1}^{k} P_{M_2}(x|y, \psi) P_{M_2}(y) = \frac{1}{N} \sum_{i=1}^{N} \sum_{y=1}^{k} P_{M_1}(y|x_i) P_{M_2}(x|y, \psi)$$
$$P_{M_1}(y|x) = \frac{P_{M_2}(x|y, \psi) P_{M_2}(y)}{P_{M_2}(x|\psi)} = \frac{1}{N} \frac{P_{M_2}(x|y, \psi)}{P_{M_2}(x|\psi)} \sum_{i=1}^{N} P_{M_1}(y|x_i).$$

3.2 Unsupervised Backward G-Net and Multisets Modeling Learning

By Multisets Modeling Learning (MML) (Xu, 1994 & 95a), multiple sets $M(x, W_y), y = 1, \cdots, m$ in the same type of parametric model are used to a multi-modes data set $\mathcal{D}_x$ which comes from different objects or models. We call a sample x_i satisfying $M(x, W_y)$ if $x_i \in M(x, W_y)$. If x_i does not satisfy $M(x, W_y)$, we have a cost $\varepsilon^q(x_i, W_y) = \beta_r r(W_y) + \min_{x \in M(x, W_y)} |T(x_i - x)|^q$ with $|T(x_i - x)|^2 = \|x_i - x\|^2$ when $q = 2$ and $T = I$. Since the cost can define a Gibbs density

$$P(x|W_y) = \frac{1}{Z_y} exp(-\varepsilon^q(x_i, W_y)), \quad Z_y = Z_y(\beta, W_y) = \sum_{i=1}^{N} exp(-\varepsilon^q(x_i, W_y))$$

we can relate MML to the G-Net-I by letting $P_{M_2}(x|y, \psi_1) = P(x|W_y)$ with the *Equal Volume Assumption* $Z_y = Z, y = 1, \cdots, m$. For example, for those special cases with $T = I$ and $q = 2$, the Q given in eq.(8) becomes

$$Q = -\sum_{y=1}^{m} \sum_{i=1}^{N} P(y|x_i) \varepsilon^q(x_i, W_y) - \log Z.$$

That is, MML is a special case of the Backward G-Net of the YING-YANG Machine.

3.3 Unsupervised Forward R-Net and Maximum Information Preservation

For the R-Net-I, let $P_{M_1}(x) = P_0(x)$ and $P_{M_2}(x|y) = P_0(x)$ with $P_0(x)$ as given by eq.(5), and then put them into eq.(7), through certain mathematics we can get

$$K(M_1, M_2) = H - Q, \quad Q = \frac{1}{N} \sum_{i=1}^{N} \sum_{y=1}^{k} P_{M_1}(y|x_i, \theta) \log P_{M_2}(y)$$
$$H = \frac{1}{N} \sum_{i=1}^{N} \sum_{y=1}^{k} P_{M_1}(y|x_i, \theta) \log P_{M_1}(y|x_i, \theta). \tag{11}$$

Since $P_{M_2}(y)$ is free, the minimization of this $K(M_1, M_2)$ gives $P_{M_2}(y) = \frac{1}{N} \sum_{i=1}^{N} P_{M_1}(y|x_i)$ which is also $P_{M_1}(y)$ from $P_{M_1}(x,y)$ directly. Thus, putting it into the above Q, we get that

$$-K(M_1,M_2)=I(y,x)=H(y)-H(y|x), \quad H(y)=-Q=-\frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{k}P_{M_1}(y|x_i,\theta)\log[P_{M_1}(y)]$$
$$H(y|x)=-H=-\frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{k}P_{M_1}(y|x_i,\theta)\log P_{M_1}(y|x_i,\theta). \quad (11)$$

That is, $I(y,x)$ is the mutual information that the output of the R-model contains about its input x. Or in the other words, $I(y,x)$ is the information transmitted from x through the R-model to y. Therefore, the minimization of this $K(M_1,M_2)$ is equivalent to the maximization of the information transmitted or preserved — an existing unsupervised learning scheme investigated by several authors (Linsker, 1989; Atick & Redlich, 1990; Nadal & Parga, 1994; Bell & Sejnowski, 1995). Here, we can also use the factorial distribution eq.(1) for $P_{M_1}(y|x,\theta)$ in eq.(11) to get factorial codings.

The YING-YANG machine at the case R-Net-II is the dual case of the Backward net G-model. It is particularly suitable for unsupervised training of a recognizer/encoder that can be applied directly to the testing data. This is a new scheme that has not been studied in the literature yet.

3.4 Unsupervised Forward-Backward RG-Net and Helmholtz Machine

For the case of RG-Net-I, let $P_{M_1}(x)=P_0(x)$ as given by eq.(5) and put it into eq.(7), we have

$$K(M_1,M_2)=H-Q+D, \quad Q=\frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{k}P_{M_1}(y|x_i,\theta)\log[P_{M_2}(x_i|y,\psi)P_{M_2}(y)]$$
$$H=\frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{k}P_{M_1}(y|x_i,\theta)\log P_{M_1}(y|x_i,\theta), \quad D=-\lim_{h\to 0}\log Nh^d. \quad (12)$$

The ALTMIN scheme now becomes:

Step 1 Fix ψ, $\theta^{new}=arg\ max_\theta\ (Q-H)$ or find a θ^{new} such that $\Delta_\theta(Q-H)\geq 0$.
Step 2 Fix θ, find $\psi^{new}=arg\ max_\psi\ Q$
or find a ψ^{new} such that $\Delta_\psi Q\geq 0$ and $P_{M_2}^{new}(y)=\frac{1}{N}\sum_{i=1}^{N}P_{M_1}(y|x_i,\theta^{new})$.

It is interesting to see that $Q-H$ is $-F(d;\theta,Q)$ used by Dayan et al (1994) for Helmholtz machine when $P_{M_1}(y|x_i,\theta)$ is represented by a Boltzmann machine like network. We have that (i) $P_{M_1}(y|x_i,\theta)$ corresponds to Q_α; (ii) $\log P_{M_2}(x,y,\psi)$ corresponds to $-E_\alpha$; (iii) P_α is $P_{M_2}(y|x,\psi)=P_{M_2}(x|y,\psi)P_{M_2}(y)/\sum_{y=1}^{k}P_{M_2}(x|y,\psi)P_{M_2}(y)$ —the companion recognition model of $P_{M_2}(x|y,\psi)$.

We can find *the YING-YANG machine in this case is equivalent to Helmholtz machine.* The Step 1 of ALTMIN corresponds to the "sleep" phase algorithm (Hinton, et al, 1994) for the recognition weights, and the Step 2 of ALTMIN correspond to the "wake" phase algorithm for the generative weights. So, we get a new perspective for Helmholtz machine. Moreover, the new perspective also provides two new views. First, an algorithm that uses gradient approach to implement the above ALTMIN will guarantee to converge, although we have not yet known the convergence of the "wake-sleep" algorithm. Second, from the YING-YANG machine in the case of Backward G-Net-I, we know that the key point for $K(M_1,M_2)=-L+D$ is that

$$P_{M_2}(x|y,\psi)P_{M_2}(y)=P_{M_2}(x|\psi)P_{M_1}(y|x_i)$$

holds when M_2 is fixed. This is always possible when $P_{M_1}(y|x_i)$ is free. However, for the case of Forward-Backward RG-Net-I, $P_{M_1}(y|x_i)=P_{M_1}(y|x_i,\theta)$ must be in the parametric family. Among this family, there may or may not be an $P_{M_1}(y|x_i)$ to let the equality hold. So the YING-YANG machine and thus Helmholtz machine is not equivalent to maximum likelihood estimation of mixture model in general. They are equivalent only when the family of $P_{M_2}(x|y,\psi)$ or/and $P_{M_1}(y|x_i,\theta)$ is large enough such that the above equality can hold eventually.

Furthermore, the YING-YANG machine for the cases of RG-Net-II and RG-Net-III can be regarded as two variants or extensions of the Helmholtz machine.

3.5 Supervided Learning

Backward G-Net We get $K(M_1,M_2)=-\frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{k}\log P_{M_2}(x_i|y,\psi)+D$ with D being independent to ψ. That is, $Min\ K(M_1,M_2)$ is equivalent to the conventional maximum likelihood estimation of ψ. When $P_{M_2}(x|y,\psi)$ is given by eq.(4), it is also equivalent to the least square training of an inverse networks.

Forward R-Net Instead of $Min\ K(M_1,M_2)$, we need to use $Min\ K(M_2,M_1)$ which is equivalent to the conventional maximum likelihood estimation of θ for $P_{M_1}(y|x,\theta)$. When $P_{M_1}(y|x,\theta)$ is given by eq.(2), it is equivalent to the least square training of a feedforward network $f(x,W_1)$. When $P_{M_1}(y|x,\theta)$ is given by eq.(3), it is equivalent to the *mixtures of experts* model (Jacobs, Jordan, Nowlan & Hinton, 1991; Jordan & Jacobs, 1994; Xu, Jordan & Hinton, 1994).

Forward-Backward RG-Net is a new supervised learning scheme by which a forward and an inversed networks are trained together.

3.6 New Learning Schemes

Several special cases of the YING-YANG machine also provide us new learning schemes, such as Backward G-Net-II and G-Net-III, Forward R-Net-II and Forward-Backward RG-Net-II and RG-Net-III. These new schemes as well as the issue of using Parzen window estimator $P_h(x)$ instead of the empirical estimator $P_0(x)$, and the issues on variants $K(M_2, M_1)$ and $K(M_1, M_2) + K(M_2, M_1)$ deserve to be further explored.

4. The EM Algorithm and Variants for SVQ, MAP-VQ and VQ

4.1 The EM Algorithm and Stochastic Vector Quantization

Let us consider eq.(8) for the case of G-Net-I in depth. Suppose that

$$P_{M_2}(x_i|y,\psi) = P_{M_2}(x_i|y, m_y, \Sigma_y) = (2\pi)^{-d/2}|\Sigma_y|^{-1/2}e^{-\frac{1}{2}(x-m_y)^T\Sigma_y^{-1}(x-m_y)}. \tag{13}$$

Let $L_{stocha} = K(M_1, M_2)$ denoting the current Kullback divergence, it follows from eq.(8) that

$$L_{stocha} = L^1_{stocha} + L^2_{stocha},$$

$$L^1_{stocha} = D - Q + \sum_{y=1}^k P_{M_2}(y)\log P_{M_2}(y), \qquad L^2_{stocha} = H - \sum_{y=1}^k P_{M_2}(y)\log P_{M_2}(y), \tag{14}$$

$$L^1_{stocha} = \frac{1}{2N}\sum_{i=1}^N\sum_{y=1}^k P_{M_1}(y|x_i)[(x_i - m_y)^T\Sigma_y^{-1}(x_i - m_y) + \log|\Sigma_y|] - \log(h^d N) + b_0, \tag{15}$$

$$L^2_{stocha} = \frac{1}{N}\sum_{i=1}^N\sum_{y=1}^k P_{M_1}(y|x_i)\log P_{M_1}(y|x_i) - \sum_{y=1}^k P_{M_2}(y)\log P_{M_2}(y). \tag{16}$$

where $b_0 = 0.5d\log 2\pi$ is a constant. The corresponding ALTMIN scheme now becomes

$$\text{E-step} \quad P_{M_1}(y|x_i) = \frac{P_{M_2}(x_i|y, m_y^{old}, \Sigma_y^{old})P^{old}_{M_2}(y)}{\sum_{y=1}^k P_{M_2}(x_i|y, m_y^{old}, \Sigma_y^{old})P^{old}_{M_2}(y)}, \; P^{new}_{M_2}(y) = \frac{n_y}{N}, \; n_y = \sum_{i=1}^N P_{M_1}(y|x_i), \tag{17}$$

$$\text{M-step} \quad m_y^{new} = \frac{1}{n_y}\sum_{i=1}^N P_{M_1}(y|x_i)x_i, \; \Sigma_y^{new} = \frac{1}{n_y}\sum_{i=1}^N P_{M_1}(y|x_i)[x_i - m_y^{old}][x_i - m_y^{old}]^T. \tag{18}$$

which is the EM algorithm for the finite Gaussian mixture specified by eq.(10) and eq.(13). It surely converges with several advantages (Xu & Jordan, in press). The experimental results on unsupervised clustering have also shown its superior performance in comparison with the k-means algorithm (Xu & Jordan, 1993).

With $P_{M_1}(y|x_i)$, we can code each vector x_i into a discrete code y by *stochastic coding*, i.e., picking y with probability $P_{M_1}(y|x_i)$; then after sending to the receiver end,we can decode from y a reconstruction of x_i by $\hat{x}_i = E[x|y] = m_y$. In other words, we have a Vector Quantization (VQ) scheme which quantizing x into $m_y, y = 1, \cdots, k$, we call the scheme *Stochastic VQ (SVQ)* .

The codebooks given by SVQ minimize L_{stocha} — the Kullback divergence between the data density and the density represented by the codebooks. This Kullback divergence is a more reasonable criterion than the conventional Mean Square Error (MSE) for measuring the quality of a VQ system. This point will become more clear when we further set up their relationship later in sec. 5. It can be justified also by two further facts. One is that L_{stocha} is also the stochastic complexity for the autoencoder by Hinton & Zemel (1994), which represents the Minimum Description Length (MDL) for the SVQ system. The other is that L^1_{stocha} represents the average bits for encoding the residual errors by the VQ system and L^2_{stocha} represents the wasted bits of the transmission channel's capacity specified by the VQ system (Xu, 1995a).

4.2 The EM Variant and Maximum Posterior Vector Quantization

A useful variant of SVQ is *Maximum Posterior VQ (MAP-VQ)*, by which we pick $y = arg\ max_y\ P_{M_1}(y|x_i)$, or equivalently we pick y with probability

$$I(y|x_i) = \begin{cases} 1, & \text{if } y = arg\ max_y\ P_{M_1}(y|x_i) \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

It also means that we minimize $K(M_1, M_2)$ subject to the constraint that $P_{M_1}(y|x_i)$ must come from the family of densities $P(y|x)$ such that $P(y|x) = 1$ for only one y and $P(y|x) = 0$ for all the other y. Putting the constraint into eqs.(13)-(16), we have

$$L_{map} = L^1_{map} - \sum_{y=1}^k P_{M_2}(y)\log P_{M_2}(y),$$

$$L^1_{map} = \frac{1}{2N}\sum_{i=1}^N\sum_{y=1}^k I(y|x_i)[(x_i - m_y)^T\Sigma_y^{-1}(x_i - m_y) + \log|\Sigma_y|] - d\log h - \log N + b_0, \tag{20}$$

with a hard-cut version of the EM algorithm:

E-step $P_{M_1}(y|x_i) = P_{M_2}(x_i|y, m_y^{old}, \Sigma_y^{old}) P_{M_2}^{old}(y) / \sum_{y=1}^{k} P_{M_2}(x_i|y, m_y^{old}, \Sigma_y^{old}) P_{M_2}^{old}(y)$,

$I(y|x_i)$ is obtained by eq.(19); and $P_{M_2}^{new}(y) = \frac{n_y}{N}, n_y = \sum_{i=1}^{N} I(y|x_i)$.

M-step $m_y^{new} = \frac{1}{n_y} \sum_{i=1}^{N} I(y|x_i) x_i, \quad \Sigma_y^{new} = \frac{1}{n_y} \sum_{i=1}^{N} I(y|x_i)[x_i - m_y^{old}][x_i - m_y^{old}]^T$.

Furthermore, letting L_{stocha}^{min} and L_{map}^{min} denote the minimum of L_{stocha} and L_{map} respectively, and noticing that the above discussed constraint, we can directly get **a useful conclusion** $L_{stocha}^{min} \le L_{map}^{min}$. That is, the *Maximum Posterior VQ* is inferior to *Stochastic VQ*.

4.3 The Relationship to Vector Quantization

In the special case that $\Sigma_y = \sigma^2 I$ in eq.(13) and $P_{M_2}(y) = 1/k$, we have that eq.(19) is equivalent to

$$I(y|x_i) = \begin{cases} 1, & \text{if } y = arg\ min_y \|x_i - m_y\|^2 \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

That is, the Maximum Posterior coding becomes the nearest neighbor coding. We call the special case *Nearest Neighbor VQ (NN-VQ)*. In addition, eq.(20) is simplified into

$$L_{NN} = \frac{1}{2\sigma^2} E_{MSE} + \log k + d \log \sigma - d \log h - \log N + b_0, \tag{22}$$

$$E_{MSE} = \frac{1}{N} \sum_{y=1}^{k} \sum_{i=1}^{N} I(y|x_i) \|x_i - m_y\|^2, \tag{23}$$

and the E step of the hard-cut EM algorithm is simplied into:

E-step $P_{M_1}(y|x_i) = \frac{e^{-\frac{1}{2(\sigma^{old})^2}\|x_i - m_y^{old}\|^2}}{\sum_{y=1}^{k} e^{-\frac{1}{2(\sigma^{old})^2}\|x_i - m_y^{old}\|^2}}$, $I(y|x_i)$ is given by eq.(21), and $n_y = \sum_{i=1}^{N} I(y|x_i)$

We call this EM algorithm variant by *the simplified hard-cut EM algorithm*.

It follows from eq.(22) that the minimization of L_{NN} is equivalent to the minimization of E_{MSE} for the fixed number k of code vectors. That is, the NN-VQ is actually the conventional VQ in this special case. The classical approaches for the conventional VQ include the k means algorithm, the LBG algorithm and various CL algorithms. The above hard cut EM algorithm provides a new alternative.

Since L_{NN} is a special case of L_{map} subject to some further constraints, we can directly get **another conclusion** $L_{map}^{min} \le L_{NN}^{min}$, *with* L_{NN}^{min} *denoting the minimum of* L_{NN}. *That is, the NN-VQ or the conventional VQ is inferior to the MAP-VQ.* The reason is that the convectional VQ imposes two extra assumptions that each codebook covers the same number of samples (i.e., $P_{M_2}(y) = 1/k$) and that each codebook covers a cluster with the same spheric-shape volume (i.e., variance), while the MAP-VQ has no these two assumptions. Together with the conclusion given at the end of sec.4.2, we give the strongest recommendation on SVQ.

5. Selecting Number of Code-Vectors and Its Automatic Implementation

5.1 A Criterion for Selecting Number of Code-Vectors

For all the existing clustering or VQ as well as CL approaches, the number k of clusters or code book is fixed in advance manually. How to appropriately select k remains a crucial hard open problem for decades in both the literature of statistics and VQ engineering. In Xu, Krzyzak, Oja (1993), an appropriate k is obtained by a heuristic approach called Rival Penalized Competitive Learning (RPCL). Here, we show that the Kullback divergence $K(M_1, M_2)$ (that is, L_{stocha}, L_{map} and L_{NN}) provides a systematical solution for selecting k.

We start at selecting k by using L_{NN} for the conventional VQ. From eq.(22), we observe that E_{MSE} decreases towards to zero when k increases towards to N. However, the term $log k$ increases to its maximum $log N$ as k increases towards to N. Thus, *the minimization of* L_{NN} *with respect to* k *forces an appropriate* k. From L_{NN}, we also see that the appropriate k depends on the variance σ^2. The minimization of L_{NN} with respect to σ^2 results in that the best variance is $\sigma^2 = \frac{1}{d} E_{MSE}$. Putting it into L_{NN}, we have $L_{NN} = \log k + \frac{d}{2} \log E_{MSE} - d \log h - \log N + b_0$. In other words, we have a simple number k selection criterion:

$$Min_k\ J(k), \quad J(k) = \log k + \frac{d}{2} \log E_{MSE}. \tag{24}$$

This criterion brings almost no extra computing to the exisiting VQ algorithms. What needs to do is to try a number of k by increasing k until $J(k)$ reaches a minimum.

Similarly we can also select k for MAP VQ and SVQ by using L_{map} and L_{stocha}. In fact in eq.(20) and eq.(16) the term $0 \leq -\sum_{y=1}^{k} P_{M_2}(y) \log P_{M_2}(y) \leq \log k$ has the same function as the term $\log k$ in L_{NN}.

Let $L(k)$ denoting L_{stocha}, L_{map} and L_{NN} at k, an obvious selection procedure for all the three cases is plotting a curve $L(k)$ versus k via attempting a number values for k. At each k, the EM algorithm or its variant given in sec.4 is used to obtain a solution with the value $L(k)$. However, the disadvantage of the procedure is quite computing consuming.

5.2 A General VQ Procedure with Automatic Selection on Number of Code Vectors

For MAP-VQ and SVQ, we can have the following much effective procedure which can automatically selecting an appropriate k during the implementation of the EM algorithm or its variants:

Initialization: Given a large enough k such that it is larger than the number of true clusters. Set the starting values of σ_y or $\Sigma_y = \sigma_y^2 I$ for all y to be equal by letting σ_y to be the same positive number that is around one k th of the variance of the whole data set. Let all m_y to be selected such that they are random located among the data samples, e.g., let them been located at k randomly picked samples from the data set with a small random deviation (i.e., $m_y = x_i + e$ with e being a random vector from Gaussian $N(0, \sigma^2 I)$ with a small σ^2). Set $P_{M_2}(y) = 1/k$ for all the y.

Step 1: Run the EM or the Hard-cut EM for SVQ/MAP-VQ respectively for one E-step and one M-step.

Step 2: Check σ_y^{new} or Σ_y^{new}, If σ_y^{new} is zero (or below a threshold) or Σ_y^{new} is singular, discard the code book m_y and all its related parameters, then let $k = k - 1$ and goto Step 1; otherwise directly goto Step 1.

The two steps are repeatedly implemented until the procedure converges or is forced to stop.

This procedure will get the code books with those redundant ones discarded. This procedure can be theoretically justified. When $\sigma_y^2 = 0$ or singular Σ_y, we have $P(x|y; m_y, \Sigma_y) = \delta(x - m_y)$. Thus, as long as $m_y \neq x_i$ for any i, we have $P_{M_1}(y|x_i) = 0$ (and thus $I(y|x_i) = 0$) for all x_i, and in turn $P_{M_2}(y) = 0$[4]. Therefore, the contribution of code y to all the updating of the EM algorithm is zero and thus is equivalent to being discarded. That is, *the code y is selected out*. This is a very interesting point— *the code number selection occurs automatically.*

We can also clear up an influential earlier misunderstanding on the EM algorithm and the Gaussian mixtures. For the cases of unequal $\Sigma_y = \sigma_y^2 I$ or Σ_y with different y, it was sometimes observed experimentally that some σ_y^2 may approach zero or that Σ_y becomes singular (Duda& Hart, 1973; and others later). It was regarded as a problem and a number of measures, e.g., bounding σ_y^2 from below (Hathaway, 1985), were proposed to avoid this problem. Now, we can get a very interesting new insight on the old "problem"—it is actually a favorable property that makes the selection of the number of code-vectors possible.

6. Adaptive EM Algorithm and Competitive Learnings

6.1 An Adaptive EM Algorithm

The EM algorithm and its variant should be implemented when the whole data set is available. Neal & Hinton (1993) proposed a family of incremental EM algorithms based on sufficient statistics and a set of sparse algorithms that emphasize a small set of significant probabilities. However, these algorithms still need to keep a memory for all the samples encountered, and thus are not adaptive or *on line* algorithms.

We start at an incremental EM algorithm that is somewhat similar to one by Neal & Hinton (1993), but not based sufficient statistics. The incremental formula is obtained from the EM algorithm eqs(17)&(18) by directly considering the changes of $P_{M_2}^{(t)}(y)$ to $P_{M_2}^{(t+1)}(y)$, $m_y^{(t)}$ to $m_y^{(t+1)}$ and $\Sigma_y^{(t)}$ to $\Sigma_y^{(t+1)}$, caused by the change of $P_{M_1}^{(t)}(y|x_t)$ into $P_{M_1}^{(t+1)}(y|x_t)$ for the sample x_t. We get

$$P_{M_1}^{(t+1)}(y|x_t) = P_{M_2}^{(t)}(y) P_{M_2}(x_t|m_y^{(t)}, \Sigma_y^{(t)}) / \sum_{r=1}^{k} P_{M_2}^{(t)}(r) P_{M_2}(x_t|m_r^{(t)}, \Sigma_r^{(t)}) \tag{25}$$

$$h(y|x_t) = [P_{M_1}^{(t+1)}(y|x_t) - P_{M_1}^{(t)}(y|x_t)]/t, \quad P_{M_2}^{(t+1)}(y) = P_{M_2}^{(t)}(y) + h(y|x_t),$$
$$\alpha_y = P_{M_2}^{(t)}(y)/P_{M_2}^{(t+1)}(y), \quad \beta_y = 1/P_{M_2}^{(t+1)}(y), \quad m_y^{(t+1)} = \alpha_y m_y^{(t)} + \beta_y h(y|x_t) x_t,$$
$$\Sigma_y^{(t+1)} = \alpha_y \Sigma_y^{(t)} + \beta_y h(y|x_t)[x_t - m_y^{(t)}][x_t - m_y^{(t)}]^T. \tag{26}$$

[4]It also means that an alternative way is to discard the code book m_y by checking if $P_{M_2}(y) = 0$.

Since $P_{M_1}^{(t)}(y|x_t)$ needs to be kept for every x_t, the algorithm is not *adaptive* but incremental. A simple trick to avoid the problem is to let $P_{M_1}^{(t)}(y|x_t) = 0$. However, the trick is too rough as an approximation to keep the solution right.

We can get a better estimate for $P_{M_1}^{(t)}(y|x_t)$ by using eq.(25) again via a set of the second latest values $m_y^{(t-1)}, \Sigma_y^{(t-1)}, P_{M_2}^{(t-1)}(y)$ that are kept in memory. With the estimate, the algorithm indeed becomes adaptive and has no need to keep any past samples.

In sequel, we justify this adaptive algorithm by showing that it brings several very interesting insights to the existing major CL algorithms.

6.2 Its Relationship to Competitive Learnings

First, we do the winner-take-all on $P_{M_1}^{(t+1)}(y|x_t)$, i.e., let $P_{M_1}^{(t+1)}(y|x_t) = 0$ for all the y except of $P_{M_1}^{(t+1)}(c|x_t) = 1$ for $c = arg\ max_r\ P_{M_1}^{(t+1)}(r|x_t)$, and also we simply let $P_{M_1}^{(t)}(y|x_t) = 0$ for all the y. In this case, $h(c|x_t) = 1/t$ and $h(y|x_t) = 0$ for other y. From eq.(26), we have $m_y^{(t+1)} = m_y^{(t)}$ for all the y except of

$$m_c^{(t+1)} = m_c^{(t)} + (\beta_c/t)(x_t - m_c^{(t)}), \quad c = arg\max{}_r\ P_{M_1}^{(t+1)}(r|x_t). \tag{27}$$

This looks the same as the conventional CL! Actually, eq.(27) provides one critical improvement over it by making the winner-take-all based on $P^{(t+1)}(y|x_t)$ or

$$y = arg\ min_y\ [(x_t - m_y^{(t)})^T (\Sigma_y^{-1})^{(t)} (x_t - m_y^{(t)}) + \log|\Sigma_y^{(t)}| - \log P_{M_2}^{(t)}(y)]. \tag{28}$$

It reduces to $y = arg\ min_y\ ||x_t - m_y^{(t)}||^2$ only in the special case that $P_{M_2}^{(t)}(y) = 1/k, \Sigma_y^{(t)} = \sigma^2 I$. In other words, the conventional CL works well with this very restrictive assumption. It has ignored four important factors. The first is $P_{M_2}^{(t)}(y)$ which represents the winning frequency of the unit y. It acts interestingly in the opposite direction to those exisiting heuristic approaches, based on *conscience* or *frequency sensitive* (Desieno, 1988; Ahalt at al, 1990 and others). In fact, the action attempts to drive the losing code away such that the automatic selection of code vectors can be realized. The other three factors are each cluster's volume size, shape and orientation. They are considered in eq.(28) via the general $\Sigma_y^{(t)}$ without assumption. Furthermore, we can also use $\beta_c h(c|x_t) = \beta_c[1 - P_{M_1}^{(t)}(c|x_t)]/t$ as an adaptive estimate of the learning rate.

Second, by considering two sets of nonnegative real numbers $a_j, b_j, j = 1, \cdots, k$ with $\sum_j a_j = 1, \sum_j b_j = 1$, we will surely have $a_j - b_j > 0$ and $a_i - b_i < 0$ for some $i \neq j$ unless $a_j = b_j$ for all $j = 1, \cdots, k$. From $h(y|x_t) = [P_{M_1}^{(t+1)}(y|x_t) - P_{M_1}^{(t)}(y|x_t)]/t$, we know that there is some $i \neq j$ such that $h(j|x_t) > 0$ and $h(i|x_t) < 0$ in the same time, unless $h(y|x_t) = 0, y = 1, \cdots, k$ which means that the algorithm has converged already. This fact justifies the use of de-learning in RPCL (Xu, Krzyzak& Oja). Furthermore, let $c = arg\ max_y\ P_{M_1}^{(t+1)}(y|x_t)$ being the winner that is the same as above, and $r = arg\ max_{y, y\neq c}\ P_{M_1}^{(t+1)}(y|x_t)$ being the rival of the winner, we enforce $P_{new,M_1}^{(t+1)}(y|x_t) = P_{new,M_1}^{(t)}(y|x_t) = 0$ for all y except of

$$P_{new,M_1}^{(t+1)}(c|x_t) = \frac{P^{(t+1)}(c|x_t)}{P^{(t+1)}(c|x_t) + P^{(t+1)}(r|x_t)}, \quad P_{new,M_1}^{(t+1)}(r|x_t) = 1 - P_{new,M_1}^{(t+1)}(c|x_t),$$
$$P_{new,M_1}^{(t)}(c|x_t) = \frac{P^{(t)}(c|x_t)}{P^{(t)}(c|x_t) + P^{(t)}(r|x_t)}, \quad P_{new,M_1}^{(t)}(r|x_t) = 1 - P_{new,M_1}^{(t)}(c|x_t).$$

In this case, from eq.(26) we have $m_y^{(t+1)} = m_y^{(t)}$ for all y except of

$$m_c^{(t+1)} = m_c^{(t)} + \gamma_c(x_t - m_c^{(t)}), \ m_r^{(t+1)} = m_r^{(t)} + \gamma_r(x_t - m_r^{(t)}),$$
$$\gamma_c = P_{new,M_1}^{(t+1)}(c|x_t) - P_{new,M_1}^{(t)}(c|x_t), \quad \gamma_r = P_{new,M_1}^{(t+1)}(r|x_t) - P_{new,M_1}^{(t)}(r|x_t). \tag{20}$$

For the γ_c, γ_r, one must be positive and the other negative, according to the above argument. Eq.(20) is almost the same as RPCL except two points. One is that it impoves RPCL by picking the winner and rival based on $P_{M_1}^{(t+1)}(y|x_t)$ instead of on $||x_t - m_y^{(t)}||^2$. The other is that the learning unit is not fixed at the winner, can be the rival too. This is less restrictive. Actually, if we want, we can continue the winner-take-all on $P_{M_1}^{(t+1)}(c|x_t), P_{M_1}^{(t+1)}(r|x_t)$ until $P_{M_1}^{(t+1)}(c|x_t)$ grows larger than

$P_{M_1}^{(t)}(c|x_t)$, then in that case we get exactly that the winner learns and the rival de-learns, i.e., we get RPCL. Not only this justifies RPCL as an approximation of our adaptive EM algorithm and provides us new hints for selecting the learning and de-learning rates, but also it explains why RPCL can automatically select an appropriate cluster number.

9. Concluding Remarks

The YING-YANG machine unifies naturally the existing major unsupervised and supervised learnings. It also provides new learning schemes, which deserve to be further explored.

The YING-YANG machine may also be used as a new coherence learning model for visual processing. We can regard that the visual information processor have two passages—the bottom-up recognition/encoding passage and the top-down generative passage. The YING-YANG machine learning attempts to make the two passages as a whole to be maximum coherence instead of only the outputs of the two passages to be maximum coherence (Becker & Hinton, 1992) or the distribution of the output of one passage and a priori distribution to be maximum coherence (Yuille, Smirnakis & Xu, 1995).

The Stochastic VQ is strongly recommended because of its minimum Kullback divergence between the data density and the density represented by the code books, the easy on parallel coding, its implementation by the EM algorithm and the beauty on automatically selecting the appropriate cluster number during the parameters' learning. Also, it is able to turn the original EM algorithm into an *adaptive* version, which is not only able to justify the current major CL algorithms, but also improve them in several aspects.

References

Ahalt, S.C. et al (1990), *Neural Networks* , Vol.3, 1990, pp277-291.

Amari, S(1995a) " Information geometry of the EM and em algorithms for neural networks", *Neural Networks*, to appear.

Amari, S(1995b), *Neural Computation* 7, pp13-18.

Atick, J.J. & Redlich, A.N. (1990), *Neural Computation* Vol.2, No.3, pp308-320.

Becker, S. & Hinton, G.E., (1992), *Nature*, Vol.355, pp161-163.

Dempster, et all (1977), *J. of Royal Statistical Society, B39*, 1-38.

A. J. Bell & T. J. Sejnowski (1995), "Information-maximisation approach to blind separation and blind deconvolution", TR-INC-9501, Feb, 1995, Institute for Neural Computation, UCSD.

Byrne, W. (1992), *IEEE Trans. Neural Networks 3*, pp612-620.

Csiszar, I., (1975), *Annals of Probability 3*, pp146-158.

Csiszar, I. & Tusnady, G. (1984), *Statistics and Decisions*, Supplementary Issue, No.1, pp205-237.

Dayan, P., Hinton, G. E., Neal, R. N. & Zemel, R.S. (1994), " The Helmholtz machine", preprint.

Desieno, D. (1988), *Proc. IEEE ICNN-1988*, Vol.I, pp117-124, New york: IEEE.

Duda, R.O, & Hart, P.E. (1972), *Pattern Classification and Science Analysis*, John Wiely.

Dempster, A.P., Laird, N.M., & Rubin, D.B. (1977), *J. Royal Statist. Society, B39*, 1-38.

Hathaway, R.J.(1985), *The Annals of Statistics*, Vol.13, No.2, pp795-800.

Hathaway, R.J.(1986), *Statistics & Probability Letters 4*, pp53-56.

Hinton, G. E., Dayan, P., Frey, B.J. & Neal, R. N. (1994), "The wake-sleep algorithm for unsupervised learning neural networks", CS Dept., Univ. of Toronto, preprint.

Hinton, G. E. & Zemel, R.S. (1994), *Advances in NIPS 6*, pp3-10.

Jacobs, R.A., Jordan, M.I., Nowlan, S.J., & Hinton, G.E., (1991), *Neural Computation, 3*, pp 79-87.

Jordan, M.I. & Jacobs, R.A. (1994), *Neural Computation 6*, 181-214.

Linsker, R. (1989), *Advances in NIPS 1*, pp186-194.

Nadal, J.P. & Parga, N. (1994), *Networks*, pp565-581.

Neal, R. N.& Hinton, G. E(1993), *A new view of the EM algorithm that justifies incremental and other variants*, CS Dept., Univ. of Toronto, preprint.

Xu, L. (1995a), "Cluster Number Selection, Adaptive EM Algorithms and Competitive Learnings", submitted.

Xu, L. (1995b), " A Unified Learning Framework: Multisets Modeling Learning", Invited paper, Proceedings of World Congress On Neural Networks, July, 1995, Washington, DC, Vol.I, pp35-42.

Xu, L. (1994), Proceedings of IEEE ICNN 1994, June 26-July 2, Vol.I, pp315-320.

Xu, L., Krzyzak, A. & Oja, E.(1993), *IEEE Tr. Neural Networks*, Vol.4, No.4, pp636-649.

Xu, L., & Jordan, M.I. (1993), *Proc. of WCNN'93*, Portland, OR, Vol. II, 431-434.

Xu, L., & Jordan, M.I. (in press), "On convergence properties of the em algorithm for gaussian mixtures", *Neural Computation*.

Xu, L, Krzyzak, A. & Yuille, A.L. (1994), *Neural Networks, Vol.7*, No. 4, pp609-628, 1994.

Yuille, A.L., Smirnakis, S.M., & Xu, L (1995), *Neural Computation 7*, pp580-593.