自己組織化と外部知性との結合に関する研究

課題番号 09680379

平成9年度～平成10年度 科学技術研究費補助金
基盤研究 (C)(2)
研究成果報告書

平成11年4月

研究代表者 松山 泰男
(早稲田大学 理工学部 電気電子情報工学科)
研究成果

この研究で扱う自己組織化とは、状態を有する多数個の素子が、連携により、一つの総合状態を創出していくことを意味している。個々の素子には、入力を変換して出力を与える単純な情報処理機能が与えられており、系全体としては複雑な情報処理を行うことが可能となる。このようなシステムは、学習に基づく高度な処理能力を可能にするものとして期待されており、本研究では主として画像のモーフィングを用いて仮想現実を実現することを目指した。開発した手法と結果は以下の通りである。

1. 学習的情報処理に利用できる自己組織化アルゴリズムとしては、従来、特徴マップとよばれるものが主流となっていたが、これは元のデータの情報著しく失う形になっている。そこで、研究者は二重レベルの自己組織化特徴マップを生成できる学習アルゴリズムの開発を行った（多重降下競合学習：研究発表[4],[5])。

2. 静止画像を情報源として多重降下競合学習アルゴリズム適用することにより、領域の特徴マップと標準カラーパターンの両面を可能にした（研究発表[4])。これは、画像情報の圧縮を行ったことになっているが、同時に変形を可能にするメッシュパターンが自己組織化により生成されたことを意味している。

3. 上の結果得られた圧縮画像を、外部知識（システムの利用者）の指定に基づいて変形（モーフィング）することを試みた（研究発表[4],[5])。これは、仮想現実としての動画を生成する手法を与えたことになっている。

4. さらに、行動を与えることを行い、3次元化を図った（研究発表[4])。

このようにして得られた手法は、今後、コンピュータにおけるヒューマンインターフェースの向上に寄与する可能性が高いと認識された。

5. 以上で述べた多重降下競合学習（研究発表[4]）や調和競合学習（研究発表[11]）は、EMアルゴリズムの特別な形として、導出できることを発見し、さらに従来のEMアルゴリズムを特別な場合として含むα-EMアルゴリズムを導出した（研究発表[1],[2],[3],[6],[7],[8],[9])。

以上のよう多くの成果を得て、本研究は終了した。

キーワード：
自己組織化、α-EMアルゴリズム、学習アルゴリズム、特徴マップ、モーフィング、外部知識、仮想現実
Multiple Descent Cost Competition: Restorable Self-Organization and Multimedia Information Processing

Yasuo Matsuyma, Fellow, IEEE

Abstract—Multiple descent cost competition is a composition of learning phases for minimizing a given measure of total performance, i.e., cost. If these phases are heterogeneous toward each other, the total learning algorithm shows a variety of extraordinary abilities; especially in regards to multimedia information processing. In the first phase of descent cost learning, elements of source data are grouped. Simultaneously, a weight vector for minimal learning, (i.e., a winner), is found. Then, the winner and its partners are updated for further cost reduction. Therefore, two classes of self-organizing feature maps are generated. One is called a grouping feature map, which partitions the source data. The other is an ordinary weight vector feature map. The grouping feature map, together with the winners, retains most of the source data information. This feature map is able to assist in a high quality approximation of the original data. Traditional weight vector feature maps lack this ability. Another important capacity of the grouping feature map is that it can change its shape. Thus, the grouping pattern can accept external directions in order to metamorphose. In the text, the total algorithm of the multiple descent cost competition is explained first. In that section, image processing concepts are introduced in order to assist in the description of this algorithm. Then, a still image is first data-compressed (DC). Next, a restored image is morphed using the grouping feature map by receiving directions given by an external intelligence. Next, an interpolation of frames is applied in order to complete animation coding (AC). Thus, multiple descent cost competition bridges “DC to AC.” Examples of multimedia processing on virtual digital movies are given.

Index Terms—Competitive learning, coordination with external intelligence, data compression, grouping feature map, image processing, multiple descent cost, self-organization, standard pattern set, vector quantization, virtual movie generation.

I. INTRODUCTION

MULTIPLE descent cost competition minimizes a measure of total performance by a composition of different learning strategies. Every learning agent here accepts a variable-size group of input elements. Each agent’s performance with respect to input is communicated over the entire agent set. This communication finds the most suitably matching agent, which is called a winner. This is the competition mechanism, which is a well-known information processing paradigm for neural networks and computational intelligence [13]. The learning agents are called artificial neurons with competition. “Artificial” is often omitted in information processing literature since there is no possibility of their being confused with real neurons.

Multiple descent cost competition is very different from ordinary types in regards to the form of the winner and its selection. As mentioned earlier, learning cost minimization is composed of different classes of operations. In the first step, the minimization grouping of input elements is attempted. This learning phase is not found in ordinary competitive learning. In the same phase, the winners are also identified. Then, the overall cost function is further minimized by updating the winners. Thus, multiple descent cost competition includes traditional competitive learning as a special case. The significance is that our generalized algorithm shows a completely unique ability for data processing.

The multiple descent cost competition that is described in this paper is different from harmonic competition [8]. Harmonic competition allows for a wide class of cost functions; however, it cannot show variable-size groupings of input elements. Thus, there is a strategy which uses the abilities of both harmonic and multiple descent cost competition. This type of complicated strategy will be presented in a later paper. Instead, a simple example of this type of coalition (which can be used to avoid bad local optimality) will be presented in this paper.

Multiple descent cost competition per se demonstrates its unique power when the composing operations for learning are heterogeneous in regards to each other. Here, each learning operation reduces the entire cost by revealing quite different phenomenon. This situation can be visually understood by starting from an example from still image compression. Fig. 1(a) is a set of input data which is a two-dimensional (2-D) array of input elements, (i.e., pixels). Each pixel belongs to a variable-shape triangular region. Fig. 1(b) is a set of standard triangles which also contains pixels. The pixels in one standard triangle form a weight vector. In multiple descent cost learning, the first phase decides a pattern for a triangle set, as seen in Fig. 1(a). This optimization reduces error measure with respect to the standard triangle set in Fig. 1(b). The arrows in Fig. 1 show corresponding triangles which are decided by competing for the best matching. There is one more different learning phase. The pixels in the standard triangles in Fig. 1(b) are renewed by reflecting the corresponding variable triangles in Fig. 1(a). This causes another minimum error optimization. Thus, there exist very different optimizations in
this example: a group optimization in Fig. 1(a) and a standard pattern optimization in Fig. 1(b).

Fig. 1 also explains a very important advantage of multiple descent cost competition. Fig. 1(a) shows that a grouping of source data is self-organized. This is called a grouping feature map. Fig. 1(b) explains that a structure on the standard sets can also be self-organized. In this illustration, a tree structure is used. This is called a weight vector feature map. It is worthy to note here that, in literature for self-organization [2], [9], [12], weight vector feature maps that correspond to Fig. 1(b) are used to generate planar partitions which look similar to Fig. 1(a). However, those types of feature maps are the result of information-losing data processing, especially in regards to continuous data. However, a grouping feature map (together with appropriate weight vectors) retains most of the input data information. This is the very property that cannot be obtained by ordinary learning for feature maps.

The above ability of multiple descent cost competition can be used in many problems. In this paper, we have focused primarily on virtual image generation. In regards to virtual image generation, the grouping feature map is an optimized partitioning of the source image. This feature map and appropriately assigned winners can restore the source image. This type of process is equivalent to data compression (DC) since the bits needed to describe the grouping feature map and the winner's assignment are much less than the input data. In data compression parlance, the weight vector set is called a codebook. The winner is a selected codeword. However, the two classes of feature maps by self-organization are missing.

The ability to restore input data is an obvious advantage as compared to ordinary feature maps. In the text, however, a much more important advantage of the grouping feature map is emphasized: The grouping feature map can change its shape. This means that the grouping pattern can accept external directions and can metamorphose. Thus, the grouping feature map can receive supervisory information during, and/or after, self-organization. The resulting grouping pattern (together with appropriate winners) can create source data that is subject to specified morphing. This type of process fits into a new class of image processing. A temporal interpolation of grouping feature maps gives a series of frames without any cross-fading. These can be played back as a virtual digital movie. This is an animation coding process (AC). Therefore, multiple descent cost competition bridges "DC to AC." A class of intelligence can be imported over this bridge. The examples in this paper will focus on the introduction of cognitive information from an external intelligence. However, this is only a part of our goal, which is to integrate unsupervised neural nets (including self-organization), supervised neural nets, and static neural nets together with external intelligence. This concept is illustrated in Fig. 2. Among the possible coordinations in this figure, this paper gives examples that connect self-organization and external intelligence. It is this link that enables one to create virtual movies with data compression from a single still image.

Besides the image processing in the text, other media signals can be handled by multiple descent cost competition. A computation method for speech is given in Appendix C. Combinatorial optimization problems can also be discussed. This type of problems are omitted in this paper since the sophistication of the integration of the agents in Fig. 2 is the objective of this paper.

II. COST FUNCTION FOR MULTIPLE MINIMIZATION

As previously mentioned, multiple descent cost competition is a composition of heterogeneous learning strategies. Each learning process alternately reduces the total cost. Thus, the form of the cost function directly affects multiple descent cost competition. The cost itself has an abstract nature independent of the inputs' physical entity. Here, we will introduce, (in Fig. 3), image processing, so that the readers' accessibility to the presenting algorithm can be facilitated. Fig. 3(a) is a region of a digital image. Its size is, say, 512 x 512 pixels. This original image is partitioned into triangular regions (16 triangles in this example). The regular pattern in Fig. 3(a) changes as the learning proceeds. Fig. 3(b) is an example of a learned grouping feature map. Fig. 3(c) is a set of initial pixels that are packed in standard triangles. As the learning proceeds, the pixel values change. Fig. 3(d) illustrates the learned weight super-vectors, i.e., standard patterns. These standard patterns are deformed and are packed into Fig. 3(b). This is a data compressed image with a grouping feature map in Fig. 3(b). After doing the above preparation, we will start to mathematically define the tools that compose multiple descent cost competition.

A. Training Data and Weight Super-Vector Set

Let X = \{x_i; t = 0, \ldots, T - 1\} be a set of input data. This corresponds to one still image [as the original in Fig. 3(a)].
Fig. 3. Progress of learning in the grouping feature map and in the standard pattern set. (a) Initial grouping. (b) Learned grouping. (c) Initial standard pattern set. (d) Learned standard pattern set.

$x_i$ is an $M$ dimensional real vector in $\mathbb{R}^M$. This is one pixel in a still image. Vectors $x_i$ are used to form a set of super-vectors $V = \{v_j; j = 0, \cdots, J - 1\}$ by minimization. $M$ and $J$ are fixed integers. Thus, $V$ is a partitioned pattern of finite set $X$. In Fig. 3(a) and (b), $v_j$ is one triangular region. $V$ corresponds to a total triangular partition, which is equal to $X$. Let $U$ be the set of possible partitions, i.e., the set of the grouping patterns of $X$. Obviously, $\|U\|$ is finite since $\|X\|$ is. A specific grouping pattern $u \in U$ will be selected so that cost, with regards to a weight super-vector set

$$C = \{c_0, \cdots, c_{N-1}\}, \quad \|C\| = N < \infty$$

is minimized. Here, $u$ is an index that can specify a triangular partition pattern, such as in Fig. 3(a) or (b). Each $c_n$ is an element in $\mathbb{R}^M \times \cdots \times \mathbb{R}^M = \mathbb{R}^{LM}$. $L$ is a fixed integer. Note that this $c_n$ corresponds to a triangular pixel set in Fig. 3(c) or (d). Weight super-vector set $C$ is often decomposed into product form [5], [10] by separating distinctive physical entities

$$C = \prod_{q=0}^{Q-1} C^{(q)}$$

$$C^{(q)} = \{c_0^{(q)}, \cdots, c_{n_q}^{(q)}, \cdots, c_{N_q-1}^{(q)}\}$$

$$c_{n_q}^{(q)} \in \mathbb{R}^{L_qM}, \quad (q = 0, \cdots, Q - 1)$$

$$\|C\| = \prod_{q=0}^{Q-1} \|c^{(q)}\| = \prod_{q=0}^{Q-1} N_q = N; \quad \sum_{q=0}^{Q-1} L_q = L.$$

Here, one specific weight super-vector in $C$ is

$$c_n = \text{col}(c_{n_0}^{(0)}, \cdots, c_{n_{Q-1}}^{(Q-1)}).$$

$Q = 3$ appears in later experiments since digital color images in terms of (Red, Green, Blue) are used.

We have used the terminology “super-vector” to describe $v_j$, $c_n$, and $c_{n_q}$ since they have multiple $M$ dimensions for $x_i$. However, “super” will be omitted in the rest of the text since its combination with other terminologies is lengthy.

B. Transformation of $v_j$

The $v_j$ class is restricted so that the data processing of $X$ that is used to form partition $V(u)$ is meaningful. This class is denoted by $G$. A one-dimensional (1-D) example for $G$ is a set of segments with arbitrary lengths. Two-dimensional examples are triangular patches and convex quadrilateral patches. Fig. 3(a) and (b) illustrate triangular patches. The class of weight vector $c_{n_1}$ (say $G_0$), however, is restricted to a regular subclass of $G$. A 1-D example of $G_0$ is a unit-length segment. Two-dimensional examples are a right-angled isosceles triangle and a square with unit-length sides. Fig. 3(c) and (d) are for the case of the isosceles triangle.

Next, we will consider a comparison of $v_j$ with $c_n$. This process is equivalent to measuring their distance or relative distortion. In this example, the form of $c_n$ is transformed so that the measurement makes sense. This transformation is described by $w$. This operation corresponds to the transformation of one standard triangle in Fig. 3(d) so that it can fit another triangle in Fig. 3(b)."
changes number \( L \) into a different integer. Therefore, the distortion measurement, (which is an important aspect of competition), includes an approximation.

C. Cost Function and Competition

The cost function defines the distortion between two vectors \( \mathbf{v}_j \in \mathcal{G} \) and \( \mathbf{c}_n \in \mathcal{G}_0 \)

\[
d_{\mathcal{G}}(\mathbf{v}_j, \mathbf{c}_n) = d_{\mathcal{G}}(\mathbf{v}_j, \mathbf{w}(\mathbf{c}_n, \mathbf{v}_j)) \in [0, \infty).
\]

(1)

with

\[
d_{\mathcal{G}}(\mathbf{v}_j, \mathbf{v}_k) = 0 \iff \mathbf{v}_j = \mathbf{v}_k.
\]

Equation (1) means that a weight vector (i.e., a standard pattern \( \mathbf{c}_n \)) is space-warped to \( \mathbf{w} \), which has the same regional shape as \( \mathbf{v}_j \). Then, the pixels in \( \mathbf{v}_j \) and \( \mathbf{w} \) are compared in order to measure distortion \( d_{\mathcal{G}} \). In order to describe distortion measure \( d_{\mathcal{w}} \) for multiple descent cost competition, we will introduce the following indexed notation in regards to the standard pattern set:

\[
C[k_0, \ldots, k_{Q-1}]
\]

\[
= \prod_{q=0}^{Q-1} C(q)[k_q]
\]

\[
= \prod_{q=0}^{Q-1} \{C(q)[k_q], \ldots, C_{\lambda_q}[k_q], \ldots, C_{\lambda_q-1}[k_q] \}.
\]

Here, the number \( k_q \) is used for a loop count of a learning phase. The summation \( \sum_{q=0}^{Q-1} k_q = m \) is used as an iteration index for the grouping. Therefore, \( \mathbf{u}_m \) is a grouping that is obtained at the \( m \)th update. Then, the total distortion between the input vectors \( \{\mathbf{x}_i\}_{i=0}^{T-1} \) with the grouping \( \mathbf{u}_m \) and the standard pattern set \( \prod_{q=0}^{Q-1} C(q)[k_q] \) can be expressed as follows:

\[
D[k_0, \ldots, k_{Q-1}; m] = D(\{\mathbf{x}_i\}_{i=0}^{T-1}, \prod_{q=0}^{Q-1} C(q)[k_q] \mid \mathbf{u}_m)
\]

\[
= D(\mathbf{v}_j(\mathbf{u}_m)), \prod_{q=0}^{Q-1} C(q)[k_q]
\]

\[
= \sum_{j=0}^{J-1} d_{\mathcal{w}}(\mathbf{v}_j(\mathbf{u}_m), \prod_{q=0}^{Q-1} C(q)[k_q]).
\]

(2)

The last line of the above distortion is the result of the following competition:

\[
D_{\mathcal{w}}(\mathbf{v}_j(\mathbf{u}_m), \prod_{q=0}^{Q-1} C(q)[k_q])
\]

\[
= \underset{\{0 \leq n_q \leq N_q; 0 \leq q < Q\}}{\text{min}} d_{\mathcal{G}}(\mathbf{v}_j(\mathbf{u}_m), \mathbf{c}_n)
\]

\[
= \underset{\{0 \leq n_q \leq N_q; 0 \leq q < Q\}}{\text{min}} d_{\mathcal{G}}(\mathbf{v}_j(\mathbf{u}_m), \mathbf{w}(\prod_{q=0}^{Q-1} C(q)[k_q], \mathbf{v}_j(\mathbf{u}_m))).
\]

Finding the argument of the minimization is the process of competition itself. Note that this minimization is jointly operated over all \( d_{\mathcal{G}} \) arguments. Each component’s minimization will be discussed later. Note also that (2) and (3) explain the measurement of the total distortion: Pack the best matching standard pattern of Fig. 3(d) into each triangular region of Fig. 3(b); then, compute the total distortion between the original and the pixel-packed images.

Next, we define a partition on \( \{\mathbf{v}_j(\mathbf{u}_m)\}_{j=0}^{J-1} \)

\[
P_{n_0, \ldots, n_{Q-1}}(k_0, \ldots, k_{Q-1}; \mathbf{u}_m)
\]

\[
= \left\{ \mathbf{v}_j(\mathbf{u}_m) \right\}_{j=0}^{J-1} \text{ achieving min with respect to } \prod_{q=0}^{Q-1} C_{\lambda_q}(q)[k_q],
\]

\[
n_q = 0, \ldots, N_q - 1; \quad q = 0, \ldots, Q - 1.
\]

This is the classification and collection of triangles in Fig. 3(b) according to each standard pattern in Fig. 3(d). Then, we can denote (by mapping \( \theta \)) the competition for making this kind of partition

\[
\theta \left( \{\mathbf{x}_i\}_{i=0}^{T-1}, \prod_{q=0}^{Q-1} C(q)[k_q] \mid \mathbf{u}_m \right)
\]

\[
= \theta \left( \mathbf{v}_j(\mathbf{u}_m), \prod_{q=0}^{Q-1} C(q)[k_q] \right)
\]

\[
= \{ \{P_{n_0, \ldots, n_{Q-1}}(\mathbf{u}_m)\}_{n_0=0}^{N_0-1} \} \cdots \{P_{n_{Q-1}=0}^{N_{Q-1}-1} \} = 0.
\]

(4)

In other words, mapping \( \theta \) stands for the above joint operation of classification and collection for the triangles in Fig. 3(b).

D. Descent Cost Mapping \( \varphi \) for Data Grouping

We will next introduce mapping \( \varphi \) as an operation that is used to find new grouping pattern \( \mathbf{u}_{m+1} \) from \( \mathbf{u}_m \), such that

\[
D(\{\mathbf{x}_i\}_{i=0}^{T-1}, \prod_{q=0}^{Q-1} C(q)[k_q] \mid \mathbf{u}_{m+1})
\]

\[
\leq D(\{\mathbf{x}_i\}_{i=0}^{T-1}, \prod_{q=0}^{Q-1} C(q)[k_q] \mid \mathbf{u}_m).
\]

That is, mapping \( \varphi \) generates new grouping pattern \( \mathbf{u}_{m+1} \)

\[
\varphi(\mathbf{u}_m, \prod_{q=0}^{Q-1} C(q)[k_q]) = \mathbf{u}_{m+1}.
\]

If we use Fig. 3 for explanation, this \( \varphi \) is an operation to modify \( \mathbf{u}_m \) of Fig. 3(a) to generate \( \mathbf{u}_{m+1} \) of Fig. 3(b) so that the total error can be reduced.

Special cases are optimal mapping \( \varphi^{opt} \), which finds the best grouping, and identical mapping \( \varphi^0 \) which just passes the previous grouping pattern to the next step. Between these two extreme mappings, there are many descent cost mappings for the grouping. Thus, we can define the set of mappings for the grouping by

\[
\Phi = \{\varphi^{opt}\} \cup \{\text{descent cost mappings for grouping} \} \cup \{\varphi^0\}.
\]

Clearly, set \( \Phi \) is finite.
E. Descent Cost Mapping $\psi_p$ for Standard Patterns

Descent cost mapping $\psi_p$ finds $\prod_{q=0}^{Q-1} C^q[k_q + \delta_{pq}]$ which achieves

$$D\left( \left\{ x_i \right\}_{i=0}^{T-1} \prod_{q=0}^{Q-1} C^q[k_q + \delta_{pq}] \mid u_{m+1} \right) \leq D\left( \left\{ x_i \right\}_{i=0}^{T-1} \prod_{q=0}^{Q-1} C^q[k_q] \mid u_{m+1} \right).$$

Here, $\delta_{pq}$ is Kronecker’s delta, which is the unity if $p = q$ holds. Otherwise, it is zero. Thus, mapping $\psi_p$ for updating the standard pattern generates

$$\psi_p \left( \prod_{q=0}^{Q-1} C^q[k_q], u_{m+1} \right) = \prod_{q=0}^{Q-1} C^q[k_q + \delta_{pq}].$$

Thus, mapping $\psi_p$ shows that the $p$th component of a standard pattern in Fig. 3(c) is updated to Fig. 3(d) so that the total error can be reduced.

An important special case of $\psi_p$ is mapping $\psi_{opt}$, which finds a generalized centroid (which is defined below). First, we will prepare a larger partition from (4)

$$\left\{ A_{np}(u_{m+1}) \right\}_{n=0}^{N_p-1} = \bigcup_{q=0}^{Q-1} P_{n_0, \ldots, n_{Q-1}}(u_{m+1}).$$

Thus, partition $A_{np}(u_{m+1})$ is a set of triangles in Fig. 3(b) where the winners have the same components.

Given this partition, let $s_{np}^{(p)} \in R^{LK_p}$ be such that

$$s_{np}^{(p)} = \arg \min_{c_n \in [0,1]} \sum_{v_j(u_{m+1}) \in A_{np}(u_{m+1})} d_w(v_j(u_{m+1}), c_n). \quad (5)$$

Here, “arg min” is an argument which achieves the minimum value. Element $s_{np}^{(p)}$ is the generalized centroid for partition $A_{np}(u_{m+1})$. This is the $\psi_{opt}$ operation. In Fig. 3(d), $\psi_{opt}$ is an operation that can find the best $p$th component of each standard pattern. The index $p = 1, 2, 3$ corresponds to a component of (Red, Green, Blue).

An extreme opposite of $\psi_{opt}$ is $\psi_0^0$, which is the identity mapping. There are many descent cost mappings whose performances lie between $\psi_{opt}$ and $\psi_0^0$. Define the following set of descent cost mappings:

$$\Psi_p = \{ \psi_{opt} \} \bigcup \{ \text{descent cost mappings for standard pattern update} \} \bigcup \{ \psi_0^0 \}.$$  

When it is difficult to find optimal mapping $\psi_{opt}$, we can apply a series of suboptimal mappings. Some examples are the steepest descent and the Newton-Raphson methods [3].

F. Mapping Scheduler

A mapping scheduler, say $A$, picks up descent cost mappings from $\Phi$ and $\Psi_q$, $(q = 0, \ldots, Q-1)$, and inserts $\theta$ in order to form a mapping sequence. Fig. 4 illustrates an example of such scheduling. This illustration corresponds to $\psi_0 \circ \theta \circ \psi_0 \circ \psi_0 \circ \psi_0 \circ \psi_0 \circ \psi_0 \circ \psi_0 \circ \psi_0 \circ \psi_0 \circ \psi_0 \circ \psi_0 \circ \psi_0 \circ \psi_0$.

If the scheduling is autonomous, i.e., independent of external intelligence, the entire learning algorithm is unsupervised. This is the case of plain self-organization without a supervisor. Convergence of the learning algorithm is defined by the following property: Both the grouping pattern $u_m$ and the standard pattern set $\prod_{q=0}^{Q-1} C^q[k_q]$ become fixed after infinite visits on each $\Phi$, $\{ \theta \}$, and $\Psi_q$, $(q = 0, \ldots, Q-1)$, as directed by the scheduler. Variety of mapping schedules may lead to different local optima.

III. LEARNING ALGORITHMS WITH MULTIPLE DESCENT COST COMPETITION: BATCH MODE

Multiple descent cost competition can operate in two modes: the batch learning and the successive learning modes. The batch mode gives an exact update mapping $\psi_{opt}$ for the standard patterns with a precise stopping rule. Therefore, we will start from the batch mode. The successive mode will be given in the next section.

In the batch mode, all input data is fed together. First, we will explain the basic batch mode where a weight vector feature map is not considered. This clearly explains the multiple descent cost property per se. The weight vector feature map will be introduced in a later section.

A. Basic Batch Mode

In the batch mode, the whole set of data $\{ x_i \}_{i=0}^{T-1}$ is fed into the learning mechanism for every update. In the basic mode, there is only a grouping feature map. No weight vector feature map is incorporated. We will begin by using an abstract description of this algorithm.

1) Batch Learning Algorithm (Basic Mode):

- Mapping scheduler
  Mapping scheduler $A$ chooses elements from $\{ \varphi_{opt}, \varphi_0 \}$, $\{ \theta \}$, and $\{ \psi_{opt}, \psi_0 \}$. Mappings $\varphi_{opt}$, $\theta$, and $\psi_{opt}$, $(q = 0, \ldots, Q-1)$, are selected infinitely often (actually, “finnitely often” is enough as long as the selection is impartial; see Proposition 3.1 and Appendix A).
• Initial state $(k_0 = \cdots = k_{Q-1} = 0, m = 0)$
  At the start of the learning, the following items are given:
    † training data $(x_i)_{i=0}^{T-1}$;
    † an initial grouping pattern $u_0 \in U$ for $(v_j(u_0))_{j=0}^{J-1}$;
    † the initial standard pattern set $\prod_{q=0}^{Q-1} C^{(q)}[0]$;
    † the initial cost
  \[ D[k_0, \cdots, k_{Q-1}; m] = D[0, \cdots, 0; 0] = D[old] = \infty. \]
  Infinity can be replaced by a large number;
    † performance threshold $\varepsilon \geq 0$.

• Grouping $(m = m + 1)$. Scheduler $A$ selects $\varphi \in \{ \varphi^{opt}, \varphi^0 \}$. If $\varphi = \varphi^0$, then go to “halt judge.” Otherwise, mapping $\theta \circ \varphi^{opt}$ is applied to $(v_j(u_{m}))_{j=0}^{J-1}$ using $\prod_{q=0}^{Q-1} C^{(q)}[k_q]$ in order to form a new set $(v_j(u_{m+1}))_{j=0}^{J-1}$. Here, $D[new]$ is computed
  \[ D[new] = D[k_0, \cdots, k_{Q-1}, m + 1]. \]

• Halt judge.
  Check to see if scheduler $A$ has completed the selection of all mappings $\psi^{opt}_q$, $(q = 0, \cdots, Q-1)$, and $\varphi^{opt}$ since the previous halt judge. If “no,” then go to “standard pattern update.” If “yes,” then check to see that the following condition has been satisfied:
  \[ (D[old] - D[new])/D[new] \leq \varepsilon. \]
  If the above inequality holds, then adopt $(\prod_{q=0}^{Q-1} C^{(q)}[k_q], u_{m+1})$ as the designed $(\prod_{q=0}^{Q-1} C^{(q)}, u)$ and halt the iteration. Otherwise, replace $D[old]$ by $D[new]$ and go to “standard pattern update.”

• Standard pattern update.
  Scheduler $A$ picks up $p \in \{0, \cdots, Q - 1\}$ and then $\psi_p \in \{ \psi^{opt}_p, \psi^0_p \}$. If $\psi_p = \psi^0_p$, then go to “grouping.” Otherwise compute mapping $\psi^{opt}_p(\prod_{q=0}^{Q-1} C^{(q)}[k_q], u_{m+1})$. Set $k_p = k_p + 1$. Then, go to “grouping.”

The above algorithm can be understood using Fig. 3 as follows. The training data of one still image [such as Fig. 3(a)] is fed into the leaning algorithm. Initial grouping pattern $u_0$ is the set of regular triangles of Fig. 3(a). Then, grouping $\varphi$ (which was chosen by the scheduler $A$) is applied to generate a new grouping pattern $(u_{m+1})$ in Fig. 3(b) for a reduced error. Scheduler $A$ picks up $p$ so that the new standard pattern set of Fig. 3(d) gives a further reduced error. Existing batch modes [6], [10], [11] correspond to special cases for scheduler $A$.

The scheduler may occasionally choose $\varphi^0$ or $\psi^0$ so that a concentration in the grouping or the weight vector updates can occur. This is based on the following scheduling:

\[ A = \{ \varphi^{opt} \rightarrow \theta \rightarrow \{ \psi^{opt}_q \}_{q=0}^{Q-1} \rightarrow \varphi^{opt} \rightarrow \theta \rightarrow \{ \psi^{opt}_q \}_{q=0} \rightarrow \cdots \} \]

realizes the fastest multiple descent cost competition. However, this scheduling may be more likely to be trapped at a local minima. Thus, $\varphi^0$ and $\psi^0_q$, $(q = 0, \cdots, Q - 1)$, are sometimes inserted in order to alleviate this effect. An example is to skip optimal grouping $\varphi^{opt} \circ \theta \rightarrow \{ \psi^{opt}_q \}_{q=0}^{Q-1} \rightarrow \varphi^0 \rightarrow \theta \rightarrow \{ \psi^{opt}_q \}_{q=0}^{Q-1} \rightarrow \varphi^{opt} \circ \theta \rightarrow \cdots \}.

This corresponds to a nonlinear programming spacer [3]. However, there is not yet the assurance of converging to a global minimum.

The convergence of this basic batch mode can be stated as follows.

Proposition 3.1 (Basic Batch Mode Convergence): For data set $(x_i)_{i=0}^{T-1}$ and $\varepsilon \geq 0$, the basic batch mode converges at a finite number of halt judges. That is, both the grouping pattern and the weight vector set converge.

Proof is given in Appendix A.

B. Block Additive Costs

The minimization of (3), or equivalently, (5) is a joint operation. That is, the whole $\prod_{q=0}^{Q-1} C^{(q)}[k_q]$ needs to be searched for. There are many occasions where this minimization can be achieved by independent operations with respect to each $C^{(q)}[k_q]$. An important example is for block additive cost

\[ d_G(x, w) = \sum_{q=0}^{Q-1} d_{K_q}(v^{(q)}, w^{(q)}) \]

where

\[ v^{(q)} \in R^{K_q} \text{ and } w^{(q)} \in R^{K_q}. \]

Then, an individual update of $\psi_q$, $(q = 0, \cdots, Q - 1)$, reduces total cost $D$. Since $\psi$ and $\psi_q$ become commutative, $\prod_{q=0}^{Q-1} \psi_q$ can be processed in parallel. Thus, the following algorithm can be obtained.

1) Parallel Update Algorithm for Block Additive Cost: This version is the same as the batch learning algorithm for the basic mode, except for the following items.

• Grouping.
  Every grouping pattern is expressed by $u_{m+Q}$.

• Standard pattern update.
  Scheduler $A$ chooses $\psi_q \in \{ \psi^{opt}_q, \psi^0_q \}$ for $q = 0, \cdots, Q - 1$. Compute the following minimization which can be performed in parallel: $(\prod_{q=0}^{Q-1} \psi_q) \circ (\prod_{q=0}^{Q-1} C^{(q)}[k_q], u_{m+Q})$.
  Also, update $k_q$ by $k_q + 1$, $(q = 0, \cdots, Q - 1)$.

This parallel update corresponds to the generation of the standard pattern set in Fig. 3(d) by computing a component of (Red, Green, Blue).

C. Typical Scheduling

An effective special case of autonomous scheduling adopts “bang-bang switching” of $\varphi^{opt}$ and $\varphi^0$. The following description is based on block additive cost. Note that block additivity itself is not a requirement for bang-bang switching.

Bang-Bang Switching and Parallel Update of $\psi_q$

Step 1 (Initialization): Training data set $(x_i)_{i=0}^{T-1}$, and design thresholds $\varepsilon \geq 0$, $\delta \geq 0$ are given. The initial states
are specified by \( u_0 \in \mathcal{U} \) and \( \prod_{q=0}^{Q-1} C^{(q)}[0] \). The initial values of \( D_{\varphi}[\text{new}], D_{\psi}[\text{old}], \) and \( D_{\psi}[\text{new}] \) are prepared so that they satisfy

\[
D_{\varphi}[\text{new}] = \infty, \quad D_{\psi}[\text{old}] > (1+\delta)D_{\psi}[\text{new}].
\]

Here, the suffixes \( \varphi \) and \( \psi \) stand for the measurements of total cost (2) at the phases of \( \varphi \) and \( \psi \). Loop count indexes are \( m = 0 \) and \( k_q = 0, \) \( q = 0, \cdots, Q-1. \)

Step 2 (Grouping): Check to see if the following inequality holds:

\[
(D_{\varphi}[\text{old}] - D_{\psi}[\text{new}])/D_{\psi}[\text{new}] < \delta.
\]

If “no,” then go to Step 4. If “yes,” then replace \( D_{\varphi}[\text{old}] \) by \( D_{\varphi}[\text{new}] \). Apply \( \theta \circ \varphi_{\text{opt}} \) to \( \prod_{q=0}^{Q-1} C^{(q)}[k_q], u_m \) in order to obtain \( u_{m+1} \) and \( D_{\psi}[\text{new}] \). Replace \( D_{\psi}[\text{new}] \) by \( D_{\psi}[\text{new}] \), and go to Step 3.

Step 3 (Halt Iteration): Check to see if the following inequality holds:

\[
(D_{\varphi}[\text{old}] - D_{\psi}[\text{new}])/D_{\psi}[\text{new}] < \varepsilon.
\]

If “no,” then go to Step 4. If “yes,” then adopt \( \prod_{q=0}^{Q-1} C^{(q)}[k_q], u_{m+1} \), and halt the iteration.

Step 4 (Standard Pattern Update): Replace \( D_{\varphi}[\text{old}] \) by \( D_{\varphi}[\text{new}] \). Apply \( \prod_{q=0}^{Q-1} \psi_{\text{opt}}^{(q)} \) to obtain \( \prod_{q=0}^{Q-1} C^{(q)}[k_q+1] \) and \( D_{\psi}[\text{new}] \). Update indexes \( m = m + Q, k_q = k_q + 1, \) \( q = 0, \cdots, Q-1. \) Then, go back to Step 2.

D. Weight Vector Feature Map by Cooperative Neurons

In traditional batch mode methods, each neuron learns its weight vector (standard pattern) without relying on any other neurons. That is, there are no cooperative neurons. However, a neuron may send signals back to others when it wins the competition. This is a schema of winner-take-quota which includes winner-take-all as a special case [2], [8], [9]. In our multiple descent cost case, the following partner update of (7) is inserted right after the standard pattern update.

1) Batch Learning Algorithm (Winner-Take-Quota):

• Partner update (winner-take-quota)

Let \( C^{(q)}[c_{\text{opt}}[k_q]] \) be a subset of \( C^{(q)}[k_q] \) decided by winner \( c_{\text{opt}}[k_q] \). Then, a weight vector feature map can be formed by updating the elements of \( b_{n_{\psi}}^{(q)}[k_q] \in O^{(q)}(c_{\text{opt}}^{(q)}[k_q]) \)

\[
b_{n_{\psi}}^{(q)}[k_q + 1] = \frac{b_{n_{\psi}}^{(q)}[k_q] + \varepsilon_{n_{\psi}}^{(q)}(c_{\text{opt}}^{(q)}[k_q])}{b_{n_{\psi}}^{(q)}[k_q]}
\]

for all \( c_{\psi}^{(q)}, q = 0, \cdots, Q-1. \). Here, \( \varepsilon_{n_{\psi}}^{(q)} > 0 \) is selected to decrease to zero for large iteration count \( m. \)

This learning operation means that nonwinning standard patterns (weight vectors) in Fig. 3(c) are updated if they are specified as partners of the winner.

IV. SUCCESSIVE MODE

The batch mode in Section III gave a firm theoretical foundation for multiple descent cost competition. However, this batch mode strongly depends on the computational accessibility of partition centroids. This is a severe limitation to its applicability to real-world problems. On the other hand, successive modes (which successively use a piece of training data) have the following advantages: Difficult centroid computation for a complex cost can be replaced by descent cost operations such as the steepest descent or by similar methods. Besides, various classes of problems can be treated as regularization problems [8]. In those cases, the centroid method is often inapplicable. Thus, a successive mode for multiple descent cost computation needs to be developed. Note that “successive mode” is not a synonym of “serial computation.” Parallel and distributed processing are very useful. Experiments on digital images and animation in Sections V and VI will use the successive mode that appears in this section.

A. Descent Cost Operations for the Successive Mode

In the successive mode, the descent cost for the grouping is done locally at each iteration. Following this phase, a gradient descent (with respect to a standard pattern) is performed instead of using the centroid. First, we will discuss the strategy of partially optimal grouping.

Each grouping, \( \varphi \in \Phi \), picks up a new pattern, \( u \in \mathcal{U} \), in order to generate \( \{v_j(u)\} \subseteq \mathcal{V} \). This process is specified by (at most) \( T \) correspondences from \( \{i = 0, \cdots, T-1\} \) onto \( \{j = 0, \cdots, J-1\} \). In the most practical cases, it is necessary to specify only a portion of these kinds of correspondences. Let \( M(T) \) be the cardinality of this correspondence. Usually, \( M(T) \ll T. \) The rest of the \( T-M(T) \) correspondences are automatically supplied. Then, grouping pattern \( u \in \mathcal{U} \) is completely specified by the correspondence of the \( M(T) \) elements in \( \{x_i\}_{i=0}^{T-1}. \) An example of \( M(T) \) from Fig. 3(b) is a necessary number of specifications in order to locate triangular vertices. Thus, descent cost grouping \( \varphi \in \Phi \) can be interpreted as a positional assignment on better \( M(T) \) elements. In the successive mode, only a part of \( \{x_i\}_{i=0}^{T-1} \), i.e., a part of \( \{v_j(u)\} \subseteq \mathcal{V} \), is given for learning at each time. Therefore, global grouping \( \psi_{\text{opt}}^{(q)} \) is replaced by iterative applications of descent cost mappings from \( \Phi \). It is worth noting that some positions of the \( M(T) \) elements may be decided independently without violating the descent cost property. Thus, the set \( \Phi \) can be divided into nonoverlapping subsets

\[
\Phi = \bigcup_{\ell=0}^{L(M)-1} \Phi_{\ell}, \quad \Phi_{\ell} \cap \Phi_m = \emptyset, \quad \ell \neq m
\]

where each \( \Phi_{\ell} \) is a set of commutative mappings. That is, either \( \varphi_i \in \Phi_{\ell} \) or \( \varphi_j \in \Phi_{\ell} \) can be applied first. Then, the \( \theta \circ \psi_{\text{opt}}^{(q)} \) application can be replaced by repeated applications of \( \theta \circ \psi_{\ell} \), \( \ell = 0, \cdots, L-1 \). Here, \( \Phi_{\ell} \) is a parallel application of the whole elements \( \varphi \in \Phi_{\ell} \). Fig. 3(b) and (b) explain that the vertices are classified into three types: \( \Phi_0 = \{\text{filled squares}\} \), \( \Phi_1 = \{\text{filled circles}\} \) and \( \Phi_2 = \{\text{empty circles}\} \). For instance, the positions of the empty circles in \( \Phi_2 \) can be optimized without looking at any other empty circles in \( \Phi_2 \). The vertices of \( \Phi_0 \) and \( \Phi_1 \) are fixed during this phase.

Next, we will discuss approximations of \( \psi_{\text{opt}}^{(q)} \), \( q = 0, \cdots, Q-1. \). Instead of finding centroids, we will use the gradients of the cost \(-\varepsilon\partial(d_{\psi})/\partial c_{\psi}^{(q)} \). Here, \( d_{\psi} \) is a distortion
of (1), and $c_{vj}^{(q)}$ is a winner that is selected from $\{c_{vn}^{(q)}\}_{n=0}^{N_v}$.
This is the steepest descent method that is used in the following description of the successive algorithm. There are many improvements concerning the speed of convergence. Such topics are found in nonlinear programming monographs [3].

B. Successive Learning Algorithm

For the learning algorithm of the multiple descent cost competition, the following progressive optimization is repeated.

- Applications of partial optimization $\Phi_t$, whose elements may be operated in parallel, $(t = 0, \ldots, L - 1)$.
- Gradient descent for weight vector update $\psi_{q}$, $(q = 0, \ldots, Q - 1)$.

Thus, scheduler $A$ directs the following mapping sequence:

$$
\{ \theta \circ \Phi_{t} \rightarrow \{ \psi_{q} \}_{q=0}^{Q-1} \} \rightarrow \cdots \rightarrow \{ \theta \circ \Phi_{L-1} \rightarrow \{ \psi_{q} \}_{q=0}^{Q-1} \}
$$

$$
\rightarrow \{ \theta \circ \Phi_{0} \rightarrow \{ \psi_{q} \}_{q=0}^{Q-1} \} \rightarrow \cdots
$$

This process is described by the following successive version of the multiple descent cost competition:

Successive Learning Algorithm

Step 1 (Initialization; $t=0$): Training data $\{x_i\}_{i=0}^{T-1}$, initial grouping pattern $u_t \in U$, and an initial standard pattern $\prod_{l=0}^{Q-1} c_{vl}^{(l)}[t]$ are given for $t = 0$. That is, a grouped vector set $\{v_{j}^{(u_{t})}\}_{j=0}^{Q-1}$ is also given for $t = 0$. The cost between $\{v_{j}^{(u_{t})}\}_{j=0}^{Q-1}$ and $\prod_{l=0}^{Q-1} c_{vl}^{(l)}[t]$ is block-additive. A stop condition is specified (setting the maximum number of iteration, say, MaxIter, is practical). The learning rate on winner $\varepsilon_{\text{win}}(t)$ and on partners $\varepsilon_{\text{ptn}}(t)$ are specified.

Step 2 (Halt Check): Check to see if the termination condition is met ($t \geq \text{MaxIter} \ast L$). If “yes,” then adopt $u_t$ and $\prod_{l=0}^{Q-1} c_{vl}^{(l)}[t]$ as the designed results. If “no,” then go to Step 3.

Step 3 (Grouping; $t \leftarrow t+1$): Let $l = t$ (mod $L$). Apply all descent cost grouping $\varphi_n \in \Phi_t$ (parallel partial optimization).

Each $\varphi_n$ generates $B_n = \{v_{j}^{(u_{n+1})}\}$. At this phase, each $v_{j} \in B_n$ finds the optimal standard pattern $\prod_{l=0}^{Q-1} c_{vl}^{(l)}[t]$ by the competition.

Step 4-1 (Standard Pattern Update for the Winner): Each $c_{vj}^{(q)}[t]$ is updated by

$$
c_{vj}^{(q)}[t+1] = c_{vj}^{(q)}[t] - \varepsilon_{\text{win}}(t) \frac{\partial d_{v}}{\partial c_{vj}^{(q)}[t]} \quad v_{j} \in B_n, q = 0, \ldots, Q - 1.
$$

Step 4-2 (Standard Pattern Update For Partners): For each $b_{vj}^{(q)}[t]$, which is a partner specified by winner $c_{vl}^{(q)}[t]$, the following update can be made:

$$
b_{vj}^{(q)}[t+1] = b_{vj}^{(q)}[t] - \varepsilon_{\text{ptn}}(t) \frac{\partial d_{v}}{\partial b_{vj}^{(q)}[t]}, \quad q = 0, \ldots, Q - 1.
$$

Step 5 (Parameter Update): Update $\varepsilon_{\text{win}}(t)$ and $\varepsilon_{\text{ptn}}(t)$, and go to Step 2.

We have two comments about the above algorithm: An example of $B_n$ from Fig. 3(b) is the set of triangles whose shape is affected by a movement of one vertex. The simplest control rules in regards to $\varepsilon_{\text{win}}(t)$ and $\varepsilon_{\text{ptn}}(t)$ decrease them monotonically. However, these learning parameters may increase as long as the final value approaches zero [8].

V. IMAGE COMPRESSION

In this section, the successive mode of the multiple descent cost competition is applied to image compression. An important issue here is to understand how a grouping feature map is obtained. The grouping feature map will be utilized for animation coding in Section VI.

A. Space Warping, Bilinear Patches, and Cost

Source data $\{x_i\}_{i=0}^{T-1}$ is a set of pixels. Each pixel $x_i$ is given by $(R, G, B)$ components $(Q = 3)$. If the image is monochromatic, only one component is given $(Q = 1)$. Set $G$ of admissible groupings is a family of triangles. Set $G_0$ for a standard pattern is a right-angled isoceles triangle. Fig. 5(a) and (b) illustrates virtual pixels $p$ and $q$ in $G_0$ and $G$. The positions of the virtual pixels are not necessarily on a physical integer mesh. Therefore, their intensities need to be interpolated using some of the elements of the set $\{x_i\}_{i=0}^{T-1}$. There are two situations for this kind of virtual pixels. In Fig. 5(c), virtual pixel $p$ has four surrounding points. Virtual pixel $q$ in Fig. 5(e) has only three surrounding points. For the four surrounding points of Fig. 5(c), virtual pixel $p = (s \ast t) \in [0, 1]^2$ has the intensity of each color component, as expressed by

$$
b(s, t) = (1 - s)(1 - t)b(0, 0) + t(1 - s)b(0, 1) + s(1 - t)b(1, 0) + stb(1, 1).
$$

This intensity is placed on position $p = (p_x, p_y)$ in Fig. 5(d), as expressed by

$$
p_x = c_{x}s + b_x t + (d_x - b_x - c_{x})st
$$

$$
p_y = c_{y}s + b_y t + (d_y - b_y - c_{y})st.
$$

For the three surrounding points of Fig. 5(e), virtual pixel $q = (s \ast t) \in [0, 1]^2$ (actually, only in the diagonal upper half of $[0, 1]^2$) has the intensity of each color component, as expressed by

$$
b(s, t) = (1 - s - t)b(0, 0) + t(1 - s)b(0, 1) + sb(1, 0).
$$

This intensity is placed on position $q = (q_x, q_y)$ in Fig. 5(f), as expressed by

$$
q_x = c_{x}s + b_x t
$$

$$
q_y = c_{y}s + b_y t.
$$

The above bilinear interpolation and bilinear patches (8)–(11) are the embodiment of the space warping $w$ in (1). Therefore, there is a way to compare different triangular patches. Therefore, we can define the total cost

$$
D = \sum_{v_j} \sum_{\text{all physical pixels in } v_j} ||\text{OriginalPixel - InterpolatedPixel}||^2
$$

$$
= \sum_{v_j} ||v_j - w(c_{vj}, v_j)||^2.
$$

(12)
B. Descent Cost Mappings $\phi_i$ and $\psi_q$

A descent cost mapping $\phi_i \in \Phi_i$, $(l = 0, \ldots, L - 1)$, generates a new grouping feature map, $u_{\text{new}} \in \mathcal{U}$ from $u_{\text{old}} \in \mathcal{U}$. Every element of $\mathcal{U}$ is a set of triangles. Thus, each $\phi_i$ finds a better position in a vertex. This is illustrated in Fig. 6. In this figure, a vertex at the center may stay or may move to one of the eight directions whose specified edges pass through the center. The candidate points divide the edges into a ratio of 1:2. Among the above nine possible positions, the least cost position, [as measured using (12)], is selected. In Fig. 6, the central vertex is moved to a position that is specified by another filled circle.

For the descent cost update of the standard pattern, the gradient descent method can be directly applied to (6). Then, one obtains

$$-\epsilon_{\text{win}} \frac{\partial (d_{\phi})}{\partial c_{\phi}^{(i)}[\text{old}]} = \epsilon_{\text{win}} [\bar{w}^{(q)}(v_j) - c_{\phi}^{(q)}[\text{old}]]$$

$$v_j \in B_n; \quad q = 0, \ldots, Q - 1 \quad (13)$$

and

$$-\epsilon_{\text{ptn}} \frac{\partial (d_{\phi})}{\partial b_{\phi}^{(0)}[\text{old}]} = \epsilon_{\text{ptn}} [\bar{w}^{(q)}(v_j) - b_{\phi}^{(q)}[\text{old}]]$$

$$v_j \in B_n; \quad q = 0, \ldots, Q - 1. \quad (14)$$

Here, $\bar{w}^{(q)}$ is the bilinear interpolation for $v_j \in \mathcal{G}$, taking standard pattern set $C^{(q)} \subset \mathcal{G}_0$ into account. Expressions (13) and (14) are used in the successive algorithm.
C. Image Version of the Successive Mode

The original successive mode is also applicable to various data entities. In this section, the successive mode of the multiple descent cost competitive learning is interpreted using the preparation in Section V-A for images. Speech processing is given in Appendix C.

Successive Learning Algorithm for Images

Step 1 (Initialization; \(t = 0\)): The following data and the initial states are given.
\[
\{x_i\}_{i=0}^{T-1} \cdots (R, G, B) \text{ pixels of image (} Q = 3 \text{) is given.}
\]
\(T\) corresponds to its size, (e.g., \(512 \times 512\)). Usually, the intensity of each component is from zero to 255 (one byte).

If the image is monochromatic, \(x_i\) is a scalar \((Q = 1)\). \(u_0 \cdots\)

Initial grouping pattern such as Fig. 3(a).

\[
\{v_j(u_0)\}_{j=0}^{Q-1} \cdots \text{ Set of patches (set of regions) such as Fig. 3(a),}
\]
\[
\prod_{q=0}^{Q-1} c^{(q)}(t) \cdots \text{ Set of initial standard patterns such as Fig. 3(c).}
\]

Partners \cdots Each element of the standard pattern set of Fig. 3(c) may specify other specific elements as partners.

Cost \cdots Block additive cost of (6).

Initial values and update rules for learning parameters \(\varepsilon_{\text{win}}(t)\) and \(\varepsilon_{\text{ptu}}(t)\). Both approach zero for a large \(t\). This need not be monotonic [8].

Step 2 (Halt Check): Halt check is attempted here.

Halt check \cdots If \(t < t_{\text{max}}\), then set \(t = t + 1\) and go to Step 3. If \(t \geq t_{\text{max}}\), then stop the iteration. A grouping feature map [such as in Fig. 3(b)] and a set of standard patterns that are filled by designed intensity [such as in Fig. 3(d)] can be obtained. Here, a feature map of \(\prod_{t=0}^{Q-1} C^{(q)}(t)\) can also be obtained.

Step 3 (Grouping Feature Map; \(t = t + 1\)): Here, parallel partial optimization for grouping is tried.

Partial optimization \cdots Compute \(t = (t \mod L)\). All \(\varphi_n \in \Phi_{t}\) are applied in order to select better vertex positions using the strategy in Fig. 6. Fig. 3(a) and (b) illustrate the optimization when \(L = 3\). Vertices in the same group are marked by filled squares, filled circles, and unfilled circles. The competition that decides a winning standard pattern is computed using (3) and the bilinear patches in (8)-(11). \(B_n\) is a set of triangles \(v_j(u_{t+1})\) whose shape is affected by a movable vertex for \(\varphi_n\).

Step 4-1 (Standard Pattern Update for the Winner): Here the winner is updated.

Update of standard patterns \cdots \(c^{(q)}(v_j(t+1)\) is updated to \(c^{(q)}(v_j(t+1)\) using \(v_j(u_{t+1}) \in B_n\) and (13), \((q = 0, \cdots, Q - 1)\).

Step 4-2 (Standard Pattern Update for Partners): Here partners are updated.

Update of partners \cdots \(b_{v_j}(t+1)\) by using (14), \((q = 0, \cdots, Q - 1)\).

Step 5 (Parameter Update): Here learning parameters are updated.

Update of learning parameters \cdots \(\varepsilon_{\text{win}}(t+1)\) and \(\varepsilon_{\text{ptu}}(t+1)\) are computed.

As was observed in [8], bad initial standard patterns lead to inferior local optimality. We added rewards and punishment which is a simplified version of (15) of [8]. Standard patterns that win too much are given handicaps [8].

D. Image Compression Experiments

We completed the preparations for multiple descent cost competitive learning in order to handle digital images:

- descent cost operations for images in Sections V-A and V-B;
- multiple descent cost competitive learning for the images in Section V-C, which were derived from the basic successive mode algorithm in Section IV-B.

As a preparation step to realize the information integration in Fig. 2, a still image was data-compressed by multiple descent cost competition. Here, it is necessary to point out the following in advance: The important goal is to obtain a grouping feature map.

Source digital images were obtained from the standard image database (SIDBA) and by a scanner. Table I describes the learning conditions for joint image compression and grouping feature map generation.

Fig. 7(a) and (b) shows the original image and a data-compressed one for a monochromatic situation. Fig. 7(c) is learned grouping feature map of \(u\), which is a set of triangles. Fig. 7(d) is standard patterns set \(C\). There is no weight vector feature map for \(C\) here. Table II summarizes the bit allocation for this experiment.

The following reproduction quality (as measured by peak-to-peak SNR) was obtained:

\[
\text{SNR}_{PP} = 10 \log_{10} 255^2 \sum \frac{\text{SourcePixel}_i}{\text{ReproducedPixel}_i}^2 = 33.0 \text{ dB}
\]

at

\[
R = 184.5/512 \times 512 = 0.704 \text{ bit/pixel.}
\]

The compression ratio is

\[
S = 2097152/184546 = 11.4.
\]

At this point, it is necessary to comment on bit allocations (i) and (ii) in Table II. Bit allocation (i) is used to describe
TABLE I
LEARNING CONDITIONS FOR JOINT IMAGE COMPRESSION AND GROUPING FEATURE MAP GENERATION

<table>
<thead>
<tr>
<th>Learning conditions</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source image size</td>
<td>$512 \times 512$ pixels;</td>
</tr>
<tr>
<td></td>
<td>monochromatic and (R, G, B) full color</td>
</tr>
<tr>
<td>Number of vertices for ${v(u)}_{\varepsilon=2}^{\infty}$</td>
<td>$65 \times 65 + 64 \times 64 = 8321$</td>
</tr>
<tr>
<td>Size of a standard pattern</td>
<td>$3 \times 3$ (right-angled isosceles triangle)</td>
</tr>
<tr>
<td>Learning parameter</td>
<td>$\varepsilon(0) = 0.2$;</td>
</tr>
<tr>
<td></td>
<td>$0.8$ is multiplied at every $\lceil t/8321 \rceil = 0$</td>
</tr>
<tr>
<td>Maximum number of data supply</td>
<td>5 times per vertex; $8,321 \times 5 = 41,605$</td>
</tr>
</tbody>
</table>

standard pattern set $C$, as illustrated in Fig. 7(d). However, this information can be specified off-line. That is, there is no need to specify $C$ for each source image, since set $C$ is universal to unlearned images. Here, the bit for (i) is zero.

Full color images that are described by (R, G, B) components are data-compressed using the product of standard pattern set $C \times C \times C$. The grouping feature map can be obtained from the gray-scale equivalent

$$\text{Gray} = 0.3 \times \text{Red} + 0.59 \times \text{Green} + 0.11 \times \text{Blue}.$$ 

Experiments were made on an image (including a cable car in San Francisco) as seen in Fig. 8. We give only a data-compressed image, Fig. 8(a), and a grouping feature map Fig. 8(b). This scene will be used in the next section for virtual movie generation. Table III describes the bit allocation in this case.

The SNR$_{PP}$ is 34.2 dB. The bit rate is

$$R = 378082/(512 \times 512 \times 3) = 0.481 \text{ bit/pixel}.$$ 

The compression ratio is

$$S = 6291456/378082 = 16.6.$$ 

Comparing the bit rate of Tables II and III, the bit rate of the full-color version is considerably smaller. This is because the bit allocation of item (ii) is around 45% of the total in Table II and 22% in Table III. Thus, higher compression can be achieved by reducing the bits needed for item (ii). Further reduction of this side-information can be achieved by restricting the movable vertices region for the grouping feature map. Tables II and III have a full region for the movable vertices (83 170 bits for $512 \times 512$ size). If all of the vertices are predefined (as in ordinary vector quantization), the side-information is a zero bit. Thus, the bit rate and the compression ratio for the color image version is

$$0.481 \geq R \geq 294912/(512 \times 512 \times 3) = 0.375 \text{ bit/pixel}$$

$$16.6 \leq S \leq 6291456/294912 = 21.3.$$ 

For the purpose of mere data compression, zero side-information is the best. However, our main goal is to use the grouping feature map for animation coding in Section VI. The above mentioned choice of this region for movable vertices will be clear from the experiments in Section VI.

VI. VIRTUAL MOVIE GENERATION

The grouping feature maps in Figs. 7(c) and 8(b) pick up considerable information that is contained in source images. Similar mesh feature maps can be obtained from traditional self-organizing maps [2], [9], [12]. However, there is a crucial difference between the multiple descent cost competition’s grouping feature map and these kinds of feature maps. Each region of the grouping feature map has accompanying standard patterns. This is the property that enabled the restoration of source images, such as in Figs. 7(b) and 8(a). Observing this source restoration process, one finds that both the mesh and the standard patterns can change their shapes. If we introduce a time axis to this transformation, a series of frames for virtual movies can be generated.

A. Movement of a Vertex Specified by External Intelligence

If we want to modify compressed images, it becomes necessary to move vertices. Fig. 9(a) and (b) illustrates this type of a process. In Fig. 9(a), vertex $p$ is requested to move to new position $q$. New position $q$ may be located in a distant region. This is quite a different situation from the optimization in Fig. 6. Thus, the vertex movements in this section (as well as in later ones) are independent of learning optimization. However, it is necessary that a resulting new pattern be an admissible grouping feature map; that is, $u_{\text{new}}$ is a member of $U$.

The resulting $u_{\text{new}}$ need not be unique. The requirements are as follows.

1) vertex $p$ is forced to move to position $q$;
2) $u_{\text{new}} \in U$.

We want to obtain $u_{\text{new}}$ by using a single-shot computation. Fig. 10 illustrates an algorithm that satisfies these constraints. Here, $p$ is a vertex that is requested to move to position $q$. Vector $s$ is a position which extends segment $pq$ by the ratio of $1 : \alpha$. Then, every vertex $r_{\text{old}}$ around position $p$ is moved to new position $r_{\text{new}}$, according to the following equations:

$$\begin{align}
|r_{\text{old}} - p| &\leq (1 + \alpha)|q - p|,
|r_{\text{old}} - q| &\leq (1 + \alpha)|p - q|,
|u_{\text{new}} - q| &\leq (1 + \alpha)|q - p|, \\
r_{\text{new}} &\equiv r_{\text{old}} + \frac{|r_{\text{old}} - q|}{|q - p|}(q - p). \tag{15}
\end{align}$$

Here, adjustment parameter $\alpha$ has the role of suppressing excessive vector movement ($\alpha = 0.1$ is used in the following experiments).
Fig. 7. Data compression for a monochromatic image. (a) Original image. (b) Data-compressed image. (c) Grouping feature map. (d) Standard pattern set.

<table>
<thead>
<tr>
<th>Bits for original image</th>
<th>Bits for compressed image</th>
</tr>
</thead>
<tbody>
<tr>
<td>$521 \times 512 \times 8$</td>
<td>(i) description of $\mathcal{C}$: $64 \times 6 \times 8 = 3,072$ bits</td>
</tr>
<tr>
<td>$= 2,097,152$ bits</td>
<td>(ii) assignment of standard patterns: $64 \times 64 \times 4 \times 6 = 98,304$ bits</td>
</tr>
<tr>
<td></td>
<td>(iii) vertices: ${(64 + 1) \times (64 + 1) + 64 \times 64 - 4} \times 5 \times 2 = 83,170$ bits</td>
</tr>
<tr>
<td>2,097,152 bits</td>
<td>184,546 bits</td>
</tr>
</tbody>
</table>
TABLE III

<table>
<thead>
<tr>
<th>Bits for original image</th>
<th>Bits for compressed image</th>
</tr>
</thead>
<tbody>
<tr>
<td>(512 × 512 × 8) × 3</td>
<td>(i) description of C: 64 × 6 × 8 = 3,072 bits</td>
</tr>
<tr>
<td>= 6,291,456 bits</td>
<td>(needed only off-line)</td>
</tr>
<tr>
<td></td>
<td>(ii) assignment of standard patterns:</td>
</tr>
<tr>
<td></td>
<td>(64 × 64 × 4 × 6) × 3 = 294,912 bits</td>
</tr>
<tr>
<td></td>
<td>(iii) vertices:</td>
</tr>
<tr>
<td></td>
<td>{(64 + 1) × (64 + 1) + 64 × 64 − 4} × 5 × 2 = 83,170 bits</td>
</tr>
<tr>
<td>6,291,456 bits</td>
<td>378,082 bits (besides 3,072 bits off-line)</td>
</tr>
</tbody>
</table>

B. Virtual Image Generation Using a Grouping Feature Map

Here, we will generate virtual or imaginary images by moving the vertices of the grouping feature map. From this point on, the source images are already data-compressed. We chose Fig. 7 for experiments about emotional feature generation. Fig. 11 is an imaginary image in which the person keeps his/her mouth closed. This image was generated from Fig. 7(c) and (d) using the methods described in Section VI-A. Note that the original image in Fig. 7(a) was not used. Therefore, our approach in regards to virtual digital movies is effective.

C. Virtual Digital Movie

Between the self-organized grouping feature map and the transformed one using (15), it is possible to generate movie frames by temporal interpolation. Let \( z(t) \) be this kind of interpolated vertex at time \( t \in [0, 1] \). Then

\[
z(t) = (1 - t)z(0) + tz(1). \tag{16}
\]

Fig. 12 illustrates a plan for generating a triangle at \( t = \tau \) from two terminals at \( t = 0 \) and \( t = 1 \). In this type of metamorphic triangle, the same element in standard pattern set \( C \) is always patched. This is possible since every element in the standard pattern set can change its triangular shape. During this process, the intensity of each pixel is also interpolated by (8)–(11).

For the generation of each frame, (16) is applied to all the vertices in the two grouping feature maps of \( t = 0 \) and \( t = 1 \). Fig. 13 illustrates the process of generating a series of frames. It is important to notice here that a snap image of \( 0 < t < 1 \) looks like a real scene. In other words, there is no cross-fading. This is examined in Figs. 14(a)–(f) (numbering is from the top-left to the bottom-right). Fig. 14(a) and 14(d) correspond to \( t = 0 \). These are data-compressed versions by multiple descent cost competitive learning. Fig. 14(c) and 14(f) are at \( t = 1 \), i.e., the transformed image using a vertex specification of (15). Fig. 14(b) and 14(e) are the \( t = 0.5 \) images. Cross-fading does not exist. Since our purpose is to create a digital movie, we prepared intermediate images of 30 frames/s. Thus, the bit rate and the compression ratio are as follows since the pinned vertex specification needs a negligible amount of information relative to the compressed image. For monochromatic, the rate and the compression ratio are

\[
\begin{align*}
R & \approx 0.704/30 = 0.023 \text{ bit/pixel}, \\
S & \approx 11.4 \times 30 = 342.
\end{align*} \tag{17}
\]

For the full color

\[
\begin{align*}
R & \approx 0.481/30 = 0.016 \text{ bit/pixel}, \\
S & \approx 16.6 \times 30 = 498.
\end{align*} \tag{18}
\]

Note that numbers \( R \) and \( S \) can be further reduced by using the specification of the movable vertex regions. All of these kinds of strategies can be rephrased by “describe only changed parts.”

Our demonstration of a virtual movie was generated using the grouping feature maps of Figs. 7(c) and 8(b). This movie proceeds as follows.

The cable car moves forward → the cable car is changed into a girl’s face → she closes her left eye → she blinks both eyes → she closes and opens her mouth repeatedly. This digital movie can be played back by animation software in computers based on MPEGx. Thus, compression by MPEGx is used further, besides (17) and (18).

VII. CONCLUDING REMARKS

Multiple descent cost competitive learning was presented. There were two main versions: the batch and the successive modes. Both modes generate a grouping feature map, a weight vector set (standard pattern set), and a weight vector feature map. In this sense, these two modes are equivalent, i.e., unified. They can be interleaved in order to make one algorithm. In this paper, however, the successive mode was used extensively because of its applicability to intelligent information processing. In our first example, digital images were data-compressed. The novelty here is that a grouping feature map (together with a standard pattern set) can reproduce good quality source images.

The standard pattern set in Fig. 7(d) was used both for the monochromatic and for the \((\mathbf{R}, \mathbf{G}, \mathbf{B})\) full color images. For the full color, \( N_{\text{RGB}} = ||C \times C \times C|| = 64 \times 64 \times 64 \). The identification of each element in \( C \times C \times C \) requires a considerable amount of information [see (ii) of Table III]. Thus, it is interesting to have a standard pattern set for the full color. Fig. 13, (given in Appendix B), is an example. The size is only \( N_{\text{F}} = ||C_{\text{F}}|| = 64 \), yet the reproduced image contains a good amount of source information (25.0 dB). Bit allocation for this case if given in Table IV. In the preparation of \( C_{\text{F}} \) for more than 30 dB, larger \( N_{\text{F}} \) and a larger amount of source data are necessary. That issue is best left for future papers.

An important accomplishment (following data compression) was the generation of virtual images using the grouping
feature map’s transformation. The original image itself can be metamorphosed using a mesh modification. This process leads to the generation of imaginary movies from data-compressed images. This type of image processing is unprecedented.

The purpose of multiple descent cost learning is not limited to the 2-D virtual movie generation that is described in the text. Before explaining this, we will schematically describe the process of virtual movie generation as follows.

Two-dimensional original $\rightarrow$ 2-D data-compressed $\rightarrow$ 2-D x frames $\rightarrow$ 2-D digital movie.

Then, it is possible to obtain the following process.

2-D original $\rightarrow$ 2-D data-compressed $\rightarrow$ three-dimensional (3-D) data-compressed $\rightarrow$ 3-D x frames $\rightarrow$ 3-D digital movie.

This is possible by specifying the z-axis (height) of the vertices of the grouping feature map. An initial attempt is given in [7].

As a final comment, we again emphasize that multiple descent cost competitive learning is applicable to other signal sources. Appendix C gives evidence for this.

APPENDIX A

Proof of Proposition 3.1:

Step 1: We will show that total cost $D$ is a multiple descent cost for $\varphi$ and $\psi_q$, $(q = 0, \cdots, Q - 1)$. First, one obtains

$$D\left(\{x_i\}_{i=0}^{T-1}, \prod_{q=0}^{Q-1} C(q)[k_q] \mid u_m\right) \geq D\left(\{x_i\}_{i=0}^{T-1}, \prod_{q=0}^{Q-1} C(q)[k_q] \mid u_{m+1}\right)$$

by applying $\varphi$. Also, by applying $\psi_{pq}$, one obtains

$$D\left(\{x_i\}_{i=0}^{T-1}, \prod_{q=0}^{Q-1} C(q)[k_q] \mid u_{m+1}\right) \geq D\left(\{x_i\}_{i=0}^{T-1}, \prod_{q=0}^{Q-1} C(q)[k_q + \delta_{pq}] \mid u_{m+1}\right).$$

Thus

$$D[old] \geq D[new]$$

holds for given scheduler $A$.

**Step 2:** Using the above inequalities, the convergence of

$$\left\{\prod_{q=0}^{Q-1} C(q)|k_q\}, u_m\right\} \rightarrow \left\{\prod_{q=0}^{Q-1} C(q), u\right\}$$

$k_q \rightarrow \infty, (q = 0, \cdots, Q - 1), m \rightarrow \infty$

can be shown by the following substeps. Here, speeds of $k_q$, $(q = 0, \cdots, Q - 1)$, and $m$ that approach infinity are directed by scheduler $A$.

**Step 2.1:** First, consider

$$\Phi = \{\varphi^{opt}\}, \Psi = \{\psi_q^{opt}\}_{q=0}^{Q-1}.$$

The learning algorithm schedules the halt check infinitely often. Then, one obtains

$$D\left(\{x_i\}_{i=0}^{T-1}, \prod_{q=0}^{Q-1} C(q)[k_q] \mid u_m\right) \geq D_{\min} \geq 0$$

$k_q \rightarrow \infty, (q = 0, \cdots, Q - 1), m \rightarrow \infty$.

Here, $D_{\min}$ is the minimum value that is obtained by the descent cost learning (it need not be the global minimum). This minimum is achieved since the learning algorithm operates on a finite set. Then, there exists a convergent subsequence

$$\left(\prod_{q=0}^{Q-1} C(q)[k_q(i)] \mid u_{m(i)}\right)_{i=0}^{\infty}.$$

Let $\left(\prod_{q=0}^{Q-1} C(q), u\right)$ be its accumulation point. This must be a fixed point of the learning algorithm. Suppose the contrary, i.e., that it is not a fixed point. Then

$$\left(\prod_{q=0}^{Q-1} C(q), u\right) \neq \left(\Psi \prod_{q=0}^{Q-1} C(q), \Phi(u)\right).$$

Thus

$$D\left(\{x_i\}_{i=0}^{T-1}, \prod_{q=0}^{Q-1} C(q) \mid u\right) \geq D\left(\{x_i\}_{i=0}^{T-1}, \Psi \prod_{q=0}^{Q-1} C(q)[k_q] \mid \Phi(u)\right)$$

holds. In the case of the equality, the learning algorithm does not use either $\Phi$ or $\Psi$ any longer. Thus, the strict inequality must hold. This, however, contradicts

$$D\left(\{x_i\}_{i=0}^{T-1}, \prod_{q=0}^{Q-1} C(q)[k_q] \mid u\right) = D_{\min}.$$

Therefore, $\left(\prod_{q=0}^{Q-1} C(q), u\right)$ is a fixed point.
Step 2-2: For general $\Phi$ and $\Psi_q$, $(q = 0, \ldots, Q - 1)$, the same property holds since each element is scheduled to appear infinitely often.

Step 3: Since all mappings operate on finite set $\{x_i\}_{i=0}^{T-1}$, the algorithm converges after a finite number of halt checks.

APPENDIX B

Full Color Standard Pattern Set: It is a well-known property of $Q = 1$ that it gives the best compression ratio. This is because the bits needed to specify the indexes of each standard pattern are less than the product cases of $Q \geq 2$. Thus, preparing a set of full color pattern is a theoretical concern.

Fig. 15 is an example of a full color standard pattern set with $N_F = 64$. As is discussed in Section VII, this set gives a quality of 25.0 dB at $R = 192690/(512 \times 512 \times 3) = 0.245$ bit/pixel, i.e., $S = 6.291456/192690 = 32.7$. Note that the separated color planes of $Q = 3$ use $N_{RGB} = 64 \times 64 \times 64$. 

---

Fig. 8. Product standard pattern set applied to a different image. (a) Data-compressed image (see Acknowledgment to the original figure). (b) Grouping feature map.

Fig. 9. Modification of a feature map by vertex movement. (a) Specification of vertex movement. (b) Warped grouping pattern.

Fig. 10. Adjustment of surrounding vertex positions.
Thus, $N_F$ needs to be larger. Here, large training images are required. Further results on the full color standard pattern set will be given in a separate paper.

**APPENDIX C**

*Gradient Descent for Time Series:* Time series are 1-D signals. Speech, seismic waves, and currency exchange rates fall into this class. There are many types of costs (distortion measures) in order to compare two time series. Fundamental measures compare two spectra of the time series. The following is one example [1] from which many basic distortion measures can be derived. Terminologies of speech processing are used in the following explanation:

$$d(f_v, f_c) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left\{ \frac{f_v(\lambda)}{f_c(\lambda)} - \log \frac{f_v(\lambda)}{f_c(\lambda)} - 1 \right\} d\lambda.$$  

Here, $f_v(\lambda)$ and $f_c(\lambda)$ are the spectra of an input and a reference speech, respectively. By using linear prediction theory, e.g., [4], one obtains

$$d(f_v, f_c) = \frac{a_c^T R_v \cdot a_c}{\sigma_c^2} - \log \frac{\sigma_u^2}{\sigma_c^2} - 1. \quad (C.1)$$

Here

$$a_c^T = (a_c(1), \ldots, a_c(m))$$

is a vector of reference autoregressive coefficients. $R_v$ is the correlation matrix that is obtained from input speech

$$R_v = \begin{bmatrix} r_0 & \cdots & r_m \\
\vdots & \ddots & \vdots \\
r_m & \cdots & r_0 \end{bmatrix}.$$
σ_e and σ_c are gain constants for the mth order autoregressive model of f_v and f_c. Then, one obtains
\[
\frac{\partial}{\partial (\log \sigma_e^2)} \frac{d(f_v, f_c)}{\sigma_c^2} = \frac{a_e^2 R_e a_c}{\sigma_c^2} - 1 \tag{C.2}
\]
and
\[
\frac{\partial}{\partial a_c} \frac{d(f_v, f_c)}{\sigma_c^2} = -2 \frac{R_e a_c}{\sigma_c^2}. \tag{C.3}
\]
It is possible to express (B.1)–(B.3) by using only gain constants and partial autocorrelation coefficients [5].

ACKNOWLEDGMENT

The author is grateful to Y. Kurosawa, M. Tan, S. Saito, and N. Takeda for their assistance and software support. Special thanks are due to Bo-Tree Productions and T. Tracy for the original figure that corresponds to the data-compressed version in Fig. 8(a). Usage is limited only to academic purposes.

REFERENCES


Yasu Matsuyama (S'77–M'78–SM’92–F’98) received the B.Eng., M.Eng. and Dr. Eng. degrees in 1969, 1971, and 1974, respectively, all from Waseda University, Japan. He received the Ph.D. degree from Stanford University, CA, in 1978. Formerly, he was a Professor and Division Chairperson of the doctor course at Ibaraki University. In 1994, he was with the National Personnel Authority as a Cochairperson of the governmental personnel selection. Since 1996, he has been a Professor at Waseda University. His current research interests include computational and communication mechanisms, symbols/subsymbol learning algorithms, multimedia information processing, and their electronic implementations.

Matsuyama received awards from the Institute of Electronics, Information, and Communication Engineers (IEICE) and the Telecommunication Foundation. He was a Councillor of the IEICE Tokyo Chapter from 1995 to 1996, and is an editorial member of the Japanese Artificial Intelligence Society from 1997.
自己組織化と外部知性との結合
—架空のコンピュータHALの生誕によって—

松山 泰男
1 平城学理工学部電気電子情報工学科

1. はじめに

「1997年」は、架空の意味ではあるが、強力なコンピュータが生まれた年となっており、そして今後はその「任務始め」の年となっている。さて、そのマシンとは、1968年に出版されたA. C. ClarkeのSF小説「2001年宇宙の旅」に登場するHAL 9000のことである。HALは、「Heuristic ALgorithm」という2つの対立した概念にありと称するという構造によって生まれてきた。この物語の中で、HALは自己をより人を対等

に意志の決め手を、さらに木星への有人宇宙飛行において乗員の生命維持装置を司るという何とかも幻の役割を担っている。ところが、身上に存在したわずかなバグのために暴走を起こし、人間に生存危機をもたらすという大災難を演じてしまうのである。

HALは現在においてももちろん実体をなしていないコンピュータである。しかしながら、同じ197年に、Deep Blueが人間のチェス・チャンピオンを破ったことから想定するに、その部類の機能はすでに実現されている。そして、この「1997年以降」を意識

して、現在の最先端コンピュータとHALとの比較が、学術書や学会論文において実にじめじめとなされている。たとえば、D. G. Storkは、A. C. ClarkeやM. Minskyへのインタビュー、そして自身を含む多くの議論の稿に基づいて、スーパーコンピューティング、人工知能、コンセクション、信頼性、そして感性情報処理の最前線を話す、HALの能力との比

較を行っている。また、コンセクションマシンの開発に携わったD. Waltzは、ICNN'97の招待講演の上で、HALへの道はまだ遠いものであると認識した上で、「論理処理だけに基づくシステムは、"hopelessly

brittle" である」と述べている。そして、「AIとコン

セクション、あるいは関連する諸分野がそれぞれの特長を提供し合い、構造が不明確な状況でも作動できる情報処理が実現されるべきだ」と説いている。

さて、不明確な状況での情報処理とは、どのような場合に登場するのであろうか。個別技術については、たとえば感性情報処理やゲームに見受けられるように、かなり成熟している。ところが、何か1つ足りない。それは高次な個別技術をきわめて多数寄せ集めたときに、その設定数の多さのために、外部からの指定だけでは統合動作にもっていくことができないことである。HALの多様な能力は、そのすべてが外部から命令されるものではないはずである。すなわちHALは、自身の機能の自己組織化を行い、まぎらわせかけたはずである。そこで、本稿では、HALへの道をたどろうとする。そして、解決されない大事な分野である自己組織化と情報処理に主眼を置くことにする。用意している内容は次の通りである。まず第2章において、学習と自

己組織化についての導入を行い、いくつかの手法に与

えることにする。そして、ここでの手法を第3章にお

ける外部知性との結合を発展させ、感性情報処理に

おける実例を示してみることにする。次の第4章は、

やや高めに立ったものとなっている。これは、学習や

自己組織化といった分野が、理論の意味でも深さを追

求できる分野であることを説明するために用意したもの

である。そして、最後の第5章においては、コネク

ションズの学習に直接対応しようとするコンピュータの問

題点について、いくつかの指摘を行うことにする。

2. 学習と自己組織化

ここでいう「学習」とは、計算機(コンピュータ)が、入力に基づいてある目的にかかわった情報処理形態を実現していくことを意味している。そして「自己組織化」とは、その学習の過程において、計算機の内部状態が何らかの自律性をもって形成されていく
ことを指すものとする。したがって、一般的な自己組織化は目に見える表現になっているとは限らないのであるが、本稿では次元又は3次元に可視化できる場合に紙数を割ることにする。そして、このような自己組織化と可視化が一体となった手法の代表がこの後で登場する自己組織化特徴マップ（self-organizing feature map）である。可視化と一体になっている場合の例については、第8章で触れることにする。

2.1 学習と自己組織化

コネクションズムに基づく学習においては、多数個の小さな素子からなる結合系が内部状態（結合重み）を更新していく。したがって、自己組織化とは、その結合重みが変化していて、特有な相互関係が生じることを意味するものとなる。図1-1はこのことを説明する基本的なモデルである。同様に、_x_と_y_は入出力ベルクトルを表すものとする。一方、_y_と結びついている素子パラメータを比較するための通信を意味するものである。

上のような自己組織化は、次々に与えられる入力データを学習素子全体が保持するデータ群を、_w_と、_m_ = 1, ..., _M_ で、似たものに相当している。そして、ここで大事なことは、_w_ はその保持しているデータ群の内部的な構造を持ち込むが可能なのかであったり、自己組織化とは、まさにこのことを指しているのである。このように、学習による自己組織化は、データの近似を基本構造として有している。したがって、この部分は似たデータ群を評価関数（コスト関数）とみなした最適化問題として定式化できることがある。

2.2 特徴マップと競合学習：可視的な自己組織化

自己組織化の過程を、数式に基づき、かつ視覚的に表現しようという試みは、D. G. WillshawとC. von der Malsburg63による回路のマップに関する研究に始まり、T. Kohonen64によって自己組織化特徴マップ65へと引き継がれて、ロボットの経路設定、機能合最適化、データベース分類などに広く応用されている。この特徴マップは、前述のような、それ自身が可視化されている自己組織化の代表例である。このとき、競合学習とはこのような特徴マップを生成させる最も有効な手法となり、その高度化は特徴マップの応用範囲を広げるための重要な技術となっている。そこで、競合学習と特徴マップとの関連について調べてみよう。

63 特徴マップと同等の構造は、それより以前に、スタンフォード大学の情報処理グループが、ベルクトル量化化に関連して発表していた。Kohonenは、自己組織化という意味で остаしをしたと解釈される。
64 よって、_w_ は、入力ベクトルの重みを結合重みとして用いる。
である。そして\( Q(x, w_m) \)は重みベクトル\( w_m \)が勝者となるときに1ととなり、そうでないときは0となる関数であり、これが出力\( y_m \)である。このとき、調和競合学習においては、与えられた入力データ\( x \)に対して、(1)式を最小化する重みベクトル\( w_m^{(0)} \)を見つける。

\[
W_{m}^{(n)} = W_{m}^{(n-1)} + \Delta W_{m}^{(n)}
\]

によりコスト関数を減少させる。

上の問題は、互いに矛盾する可能性がある目的関数\( f \)と\( g \)を、競合学習により折り合い付けて、すなわち調和的に減少させるという多目的最適化問題となっている。このとき、\( h_f \)はデータのラベルを表すものであり、(1)式は単なる近似度ではなくデータへのラベルや教師信号を受け付け可能なことを表している。このような多目的最適化は、いろいろな問題を解くのに有効な定式化である。ところが、\( f \)と\( g \)が協調する場合には、消去する目的が最も適当に避ける問題に適用できる。2.4 調和競合学習による特徴マップの生成

2.2節で述べたように、勝者のパターンにも次のような学習を行わせると、特徴マップが生成される。

\[
W_{N(m)}^{(n)} = W_{N(m)}^{(n-1)} + \Delta W_{N(m)}^{(n)}
\]

ここに\( N(m) \)は、勝者\( w_m \)が指定するパターンの番号である。68、このような特徴マップは、重みベクトル同士のトポロジーを指定し、その上で学習により実際の空間配置を得るものである。したがって、正確には「重みベクトル特徴マップと呼ぶべきである。多目的最適化に基づく調和競合学習とその重みベクトル特徴マップは、いっそう広範な応用問題に適用できる。69

しかしながら、この種の特徴マップは、データ空間の形状性の一部を抽出しただけになっている。そのために、この特徴マップのみを用いて原データの近似の再生やパターン認識を行うとすると、失敗に終わることが多い。そこで、次節においては、原データに固有なパターンを別的方式で抽出する特徴マップの自己組織化をしてみることにする。

2.5 グループ化特徴マップの生成

ここでは具体的な例をもって、「グルーピング特徴マップ」とよばれるもう1つの特徴マップを提案する。

図-2(a)は、\( N \times N \)ピクセルからなるデジタル画像であるとする。1つのピクセルは、色調の意味で、8

68 パターン指定の例としては、図-1で要素Aを要素Bだけにかつ図表作成的なものである。この指定は、ユニコルド空間における巡回マップ問題や、いろいろな制約条件を満たした最適化問題に対するものである。

69 これにより、この特徴マップは、高次元のデータの低次元表現を可能にし、学習が容易になる。
3.1 ラベルによる自己組織化の誘導
ここでは、入力データにあらかじめカテゴリーを表すラベルが与えられている場合を考える。この場合、もしこのラベルと自己組織化の様子が合わせなければ、通常の勝者は別の重みベクトルが競合学習の勝者として選ばれる。これは、(1)式におけるが大きさをとるか、あるいは (2) 式においてを用いることを意味する。学習ベクトル量化62)は、このような形の調和競合学習の特別な形となっている。

教師信号をラベルにより与える方法は、外部知性が自己組織化の様子を制御する方法としては最も単純なものである。しかし、これによって、このように部分的に教師信号を与えるか、もしくは単純な自己組織化に比べて驚くほどの性能向上をもたらす。

3.2 外部知性による自己組織化の操作
ここでは、外部知性が学習系における自己組織化を利用する場合を、感性情報処理を例にあげて調べてみる。

図3は画素のグループ化特徴マップを例にとった結果である。図3(a)は図2(a)と同様に学習の結果として得られたグループ化特徴マップであるとする。そして、外部知性はこのグループ化特徴マップに対して、頂点の位置を別の位置に変動するように指定しているとする。この指定は、グループ化特徴マップの正則性(三角形の保存)を保つとする。図3(b)のような結果を得たとする。このような教師情報は、自己組織化のどの段階においても与えることができるのであるが、ここでは、生成されたグループ化特徴マップを、外部知性からの指定に基づいて変形することを考えてみよう。これは実は、上のようなメッセージパターンを仲介にして仮想的な静止画像を作り、さらには仮想的な動画像を作ることに相当することである。

図4において、r=0.0は自己組織化の結果生成されたグループ化特徴マップにテクスチャ(たとえば図2(b)の重みベクトル)を埋め込んだ再生画像である。一方、図4でr=1.0は、図3(b)のように、外部知性からの指定に従って変形がなされたグループ化特徴マップを用いた生成画像であり、女性は片目を閉じている。また、r=0.5は時間的に補間を施してできたグループ化特徴マップを用いた生成画像であり、片目を閉じる途中である。このように、任意の時刻におけるグループ化特徴マップに重みベクトル(色のテクスチャ)を割り当てることができ、任意のフレーム数の圧縮画像を作成できる。したがって、以上のようにして生成された一連のフレームをモニタ上で時間表示してみれば仮想的な動画像が得られることになる。すなわち、このような手順を情報圧縮として考えると、静止画像の意図で、0.0の画像は、0.481 bit/pixelで34.2 dBとなっている。しかしながら、これを動画像圧縮の意味で考えると、頂点の移動情報は一枚の画像全体に比べてきわめて小さいので、さらにフレームレートで割ることになり、レートは0.481/30=0.016 bit/pixelとなる。これは、外部情報が「意味あるとい」とは意図の部分のみを指定していることに起因している。なお、実際に動画像として表示する場合には、さらにMPEGの圧縮率が得られることを付け加えておく。

4. 確率的学習：さらなる理論の発展
コネクショニズムにおける学習や自己組織化は、実に興味深い確率構造を反映したものとなっていて、理論的深さが存在する。ここではそのことにについて調べてみることにする。

4.1 トレーニングデータと確率
コネクショニズムにおける学習や自己組織化においては、通常、大量のトレーニングデータが与えられる。たとえば、図1においても、入力1つ1つのサンプルベクトルxとして供給される。また、教師あり学習の代表となっている連続逆伝播法についても同じことがいえる。すなわち、トレーニングデータは、密度が不均一な空間からたまたま現れたサンプルなのである。ところが、第2章の各部分で提示した学習と自己組織化の手法においては確率構造は特に意識されておらず、いずれも「統計モデルを仮定しないアルゴリ
EMアルゴリズムとその拡張

EMアルゴリズム (Expectation-Maximization algorithm) は、その名称が示すように、確率構造のよさあるいは確率分布表す平均値をとり、それを最大化する手法を提案法と称しており、その応用範囲はきわめて広い、そして多くの学習法はこの場合の特徴となっている。ところが、コンテクスト、ノズル状学習法は信頼性アルゴリズムそのものに最適な影響を表している。たとえば、WEM (α-Weighted EM algorithm) と呼ばれるものがそれであり、次のようなアルゴリズムとなっています。

WEM: E-step 次のような期待値
\[ Q^{(m)}_{X|Y}(y|\psi) = \int p(x,y|\psi) \left( \alpha \frac{p(x,y|\psi)}{p(x,y|\theta)} + (1-\alpha) \right) dx - 1 \]

WEM: M-step 次のような最大化を行う。
\[ \psi^* = \arg \max Q^{(m)}_{X|Y}(y|\psi) \]

WEM: U-step 収束判定を行う。未収束なら、\( \psi^* \)を再計算しE-stepに戻る。

このアルゴリズムにおいて、\( X \)は完全データ、そして\( Y \)は実際に観測できるデータ、すなわち不完全データである。そして、\( \psi \)は確率値の関数形を表すパラメータである。\( \alpha \)は、\( \alpha \)-ダイバージェンス\(^{97}\)のパラメータである。通常のEMアルゴリズムは、\( \alpha = 1 \)の場合に対応している。この\( \alpha \)の値は学習の速度や局所最適解の影響を与える、関連した議論から、確率重みに付いたフィッシャー情報量\(^{98}\)の一般化が得られる。

WEMはもう1つ大差性を有している。すなわちWEMを用いると、順方向と逆方向が対になって、たたブロックが得られるので、これをストリックアレイ\(^{99}\)とつなぐことができる。これは、分割統治 (divide-and-conquer) 基本とモデリングと密接に関連している。

5. 「Wah教授からの宿題」とその対極

本書では、架空のコンピュータHALから話を起こし、HALへの道の必要条件の1つと目されている学習
と自己組織化、そして外部知識との結合に話を進めた。

また、アルゴリズムの一般化および高度化の観点から、確率構造に基づき学習理論についても論述した。
そこで、もう一度HALを引き合いに出して、学習、自己組織化、コンピュータについて次のようなことを考えている。

まず第1の項目は、アーキテクチャに伴う変更のようである。本稿で述べた学習の自己組織化はコンテクスト、ノズル状学習法に基づくもので、それを情報処理として利用する場合には、類似の小さな、そしてきわめて多数のプロセッサ群からなるMIMD構造が対応するアーキテクチャになる。ところが、ユーザ側 (プログラマ) は、複雑度に起因する制約のため、各プロセッサに共に個別の指示を出すことはできない。このような場合、パックグラウンドに何らかの自己組織化の情報処理を設定し、ユーザは、ほんの少しずつに指示を与えることになる。これは実現性がありそうである。さて、問題は次の段階である。イリノイ大学BのB. Wah教授は、以前に、Thinking Machines社が会社更生を申請したことに関連して、私に次のような問い合わせをした。すなわち、万が1万から1000万位の数のプロセッサをもつ超並列コンピュータを考えよう。これは、原理的にはもはや可能になっている。そしてこのようなマシンが欲しい。そうすると、このようなコンピュータを実用ベースで必要とする分野を皆で考えてみなければならない\(^{111}\)。その通りである。よい適用分野の発見は、何にも増して進歩を助長させるのである。

次に、そのような巨大なコンピュータではなくて、逆にその対極として、最も身近なものについて考えてみよう。それはVLSIのさらなる進歩\(^{111}\)に模索した装着型コンピュータ (wearable computer) である。学

---

\(^{97}\) パラメータ \( \alpha \) を有し、\( \alpha = 1 \) のときにKullback-Leibler数となるような情報量、さらに特例で wreath bang \( \alpha \) は Shannon\(^{98}\) とエントロピーである。

\(^{98}\) 負の対数尤度を確率のパラメータベクトルで2段階微分し、さらにその期待値をとことで行う。

\(^{99}\) systolic array は、互いに交差する入力出力をもつブロックからなる情報処理構造。

\(^{101}\) 同様に、HALはイリノイ大学のあるアーキテクチャに生まれたことになっている\(^{101}\)。これは、事実上は無数の並列マシン、ILLIAC 4の生誕である。文献2には、日本のメーカーも含め、現在までにHACKが生誕しに来ることの伝えられるされている。

\(^{111}\) ただし、各プロセッサに1つの静止子を置き出すといった機構の例題は、数も答えからに除いてお預けたい」ということである。
習や自己組織化は常に力を監視することによって成り立つ。人間の感性情報処理はちょうどこの形になっている。ここにも適用分野を見つけるのはある。

謝辞 本稿の準備段階で貴重なコメントをいただいた担当委員と読者の方に深く感謝いたします。そして、製品設計を活用させていただいた各会社や諸企業に深甚の謝を表します。また、参考文献については従来の文献に従ったため、多くの優れた文献を削除せざるを得ませんでした。ここにその旨を記してお詫びし、大方のご寛容をうかがいます。

参考文献

（平成9年7月23日受付）

松山 泰男（正会員）
WEM Algorithms and Probabilistic Learning

YASUO MATSUYAMA†, ‡
† Department of Electrical, Electronics and Computer Engineering, Waseda University, Tokyo, 169-8555 Japan.
TEL/FAX +81-3-5286-3180
E-mail: yasu@wizard.elec.waseda.ac.jp
‡ Sympat Committee of the RWCP, Japan.

Abstract

A new class of Expectation and Maximization algorithm is presented and applied to probabilistic learning. This algorithm can be derived from the non-negativity of the $\alpha$-divergence and Bayesian computation. The design parameter $\alpha$ specifies a prior probability weight for the learning. Accordingly, this algorithm is called the $\alpha$-Weighted EM algorithm (WEM, $\alpha$-EM). The traditional EM algorithm (log-EM) corresponds to special case of $\alpha = -1$. Besides the WEM, a practically more useful version, the W-GEM (Weighted GEM), which gradually updates parameters is presented. Then, this algorithm is applied to learning in mixture-of-probability neural networks. In the discussion of update equations, extensions of basic statistical theories such as Fisher’s efficient score, his information matrix and Cramér-Rao’s inequality are given. Therein, a concept of an aptitude number is defined in terms of the number $\alpha$. Numerical experiments on such networks show that the WEM (G-WEM) outperforms the log-EM on the learning speed for appropriate aptitude numbers.

This paper unvels another new method. It is found that each WEM structure can be used as a building block to generate a learning systolic array. Thus, it becomes possible to create functionally distributed systems by appending such blocks as monitors.

1 Introduction

Computing an expectation and maximization is a set of powerful tools in Bayesian statistics theory including computational learning and pattern recognition. Dempster, Laird and Rubin (abbreviated by DLR) [4] collected expectation and maximization examples from diverse areas. Therein, they built a unified theory and coined the name “EM algorithm.” Since then, this algorithm has been successfully applied to many fields [16], [19]. One of the most creative enrichment to this field from computational learning is Jordan and Jacobs’ mixture-of-expert neural networks [6].

The core part of the EM algorithm is a repeated computation of the following two steps.

[E-step] Computing the expectation of the log-likelihood using tentative complete data.
[M-step] Computing an argument that maximizes this expectation.

Thus, this algorithm heavily depends on the logarithm and the non-negativity of the relative entropy, i.e., the Kullback-Leibler divergence. But, this paper claims the following fact. There is a generalized class for such a logarithm and non-negativity pair [10], [11], [12], [13], [14]. This comes from the divergence of order $\alpha$ [2], [3], [5], [8], [20]. It is worthy to note that the history of this information divergence traces back to early days of information theory [20]. In this paper, we first derive a general EM algorithm based upon the above $\alpha$-divergence. A key idea is to use a likelihood ratio in terms of a generalized logarithm. The obtained algorithm is called the $\alpha$-Weighted EM algorithm (WEM, $\alpha$-EM) by reflecting the probability weight in terms of the design parameter $\alpha$.

Following to the formal derivation of the WEM algorithm, a way to widen applicability of the WEM is given. It is a partial update of structural parameters, and is called the W-GEM (Weighted Generalized-EM) which is a further extension of the DLR’s GEM [4]. Besides the extensions of the log-EM and GEM, we extend the ECM [18] and ECME [9] to the WEM case.

Next, computational efficiency is checked for neural networks of mixture probability learning. For this purpose, extensions of existing statistical theories such as the Fisher’s efficient score and his measure of information as well as the Cramér-Rao’s bound are made. Numerical experiments show that the plain EM (i.e., log-EM), which is the case of $\alpha = -1$, can be speeded up. That is, appropriately chosen $\alpha$ gives faster and non-oscillating trends of learning. Thus, the WEM algorithm is shown to be a truly meritorious generalization.

This paper gives further contributions concerning to the expectation and maximization algorithm. The WEM algorithm per se has a cyclic structure whose input and output can be interchanged. This structure can be regarded as one building block to construct a larger system. One of such an organized structure is a learning systolic array. This array accepts such building blocks as additional monitors. At the end of this paper, discussions on making functionally distributed systems with such monitors are given.

2 General Divergence Measures

2.1 Rényi’s $\alpha$-Divergence and Plain $\alpha$-Divergence

A generalized information divergence, which is derived from a set of weaker axioms, is presented by Rényi in early days of information theory [20]. His measure of information is called the Rényi’s $\alpha$-divergence. But, it is better to replace his $\alpha$ by $(1 + \alpha)/2$ and normalize the total amount in order to obtain simple relationships to its descendants. Then, this version of the Rényi’s $\alpha$-divergence between two probability densities $p(x)$ and
Let
\[ Q_{X|Y}^{(α)}(ψ|φ) = E_{PX,Y,ψ} \left[ L_{X}^{(α)}(ψ|φ) \right]. \] (8)

On the α-divergence \( D_P^{(α)}(φ||ψ) \), we use the following conventions:
\( ϕ \leftrightarrow P_{X|Y,ϕ}(x|y,ϕ), \quad ψ \leftrightarrow P_{X|Y,ψ}(x|y,ψ). \)

Then, one obtains the following equation (9) by computing \( D_P^{(α)}(φ||ψ) \) with the separation of conditional probabilities.
\[ \frac{4}{1 - α^2} \left\{ \frac{p_{X|Y}(ψ)}{p_{X|Y}(ψ)} \right\} \frac{1 + α}{2} \left\{ \frac{p_{X|Y}(ψ)}{p_{X|Y}(ψ)} \right\} \frac{1 + α}{2} \int dx \frac{d}{dx} \left[ \frac{p_{X|Y}(ψ)}{p_{X|Y}(ψ)} \right]. \]

Let
\[ S_{X|Y}^{(α)}(ψ|φ) \equiv \int X \left\{ \frac{p_{X|Y}(ψ)}{p_{X|Y}(ψ)} \right\} \frac{1 + α}{2} \frac{d}{dx} \left[ \frac{p_{X|Y}(ψ)}{p_{X|Y}(ψ)} \right] dx \]
\[ E_{PX,Y,ψ} \left[ \frac{p_{X|Y}(ψ)}{p_{X|Y}(ψ)} \right] \frac{1 + α}{2} \frac{d}{dx} \left[ \frac{p_{X|Y}(ψ)}{p_{X|Y}(ψ)} \right]. \]

Then,
\[ Q_{X|Y}^{(α)}(ψ|φ) = \frac{2}{1 + α} \{ S_{X|Y}^{(α)}(ψ|φ) - 1 \}. \] (11)

Therefore, one obtains from (9) that
\[ L_{Y}^{(α)}(ψ|φ) = \frac{Q_{X|Y}^{(α)}(ψ|φ)}{1 - α} \left\{ \frac{p_{X|Y}(ψ)}{p_{X|Y}(ψ)} \right\} \frac{1 + α}{2} \frac{d}{dx} \left[ \frac{p_{X|Y}(ψ)}{p_{X|Y}(ψ)} \right] \] (12)

This is the basic equality which is directly related to the WEM algorithm below. The left-hand side is the α-log likelihood ratio of incomplete data (7). The right-hand side includes the complete data expectation of the α-log likelihood ratio of two complete data probabilities (8). Then, the following algorithms are obtained.

[**WEM-I**] The WEM algorithm is a series of applications of the E-step and the M-step followed by the U-step for \( α \leq 1 \)

**E-step**
Compute \( Q_{X|Y}^{(α)}(ψ|φ) \).

**M-step**
Compute \( ψ^* = \arg \max_ψ Q_{X|Y}^{(α)}(ψ|φ) \).

**U-step**
Replace \( ϕ \) by \( ψ^* \) and go back to E-step until convergence is achieved.

Individual discussions on \( α \) lead to the following version.

[**WEM-II**] The WEM algorithm is classified into the following cases depending on the parameter \( α \).

**E-step**
Compute \( S_{X|Y}^{(α)}(ψ|φ) \).

**M-step**
Compute \( ψ^* \) by the following methods.

1. \( α < -1 \):
\[ ψ^* = \arg \min_ψ S_{X|Y}^{(α)}(ψ|φ). \]
2. \( α = -1 \):
\[ ψ^* = \arg \max_ψ E_{PX,Y,ψ} \left[ \log p_X|ψ \right]. \]
3. \(-1 < α ≤ 1 \):
\[ ψ^* = \arg \max_ψ S_{X|Y}^{(α)}(ψ|φ). \]
4. \( α > 1 \):
Avoid this case.

[**U-step**] Replace \( ϕ \) by \( ψ^* \) and go back to E-step until convergence is achieved.

Note that
\[ Q_{X|Y}^{(α)}(ψ|φ) = Q_{X|Y}^{(α)}(ψ|ψ) = 0. \] (13)

There are two comments on the above algorithms. First, the convergence can be claimed in \( L_{Y}^{(α)}(ψ|φ) \) because of (7), WEM’s non-decreasing property and non-negativity of \( \max_ψ Q_{X|Y}^{(α)}(ψ|φ) \). But, this convergence is not necessarily on each parameter [21]. Secondly, for \( α < -1 \) of the WEM-II, the M-step is not the maximization but the minimization.

### 3.2 W-GEM, W-ECM and W-ECME

For moderate size of problems, it is usually rare to obtain exact “arg max” for the WEM (even for the EM). Therefore, we need a GEM version [4] for the WEM. We call this version the W-GEM. The algorithm is as follows.

[W-GEM]

**M-step:**
Choose \( ψ^+ \) such that
\[ Q_{X|Y}^{(α)}(ψ^+|φ) ≥ Q_{X|Y}^{(α)}(ψ|φ) = 0. \] (14)

The most popular strategy to establish Equation (14) is the Newton-Raphson method or the generalized least squares. This is possible only if the Hessian matrix of \( Q_{X|Y}^{(α)}(ψ|φ) \) is positive definite for convergence. Therefore, it is necessary to check the stability of the W-GEM which depends on the number \( α \).

If the parameter set \( ψ \) is expressed by a pre-defined partition such that
\[ ψ = \bigcup_{s=1}^{S} \Psi_s, \]
then the M-step of the W-GEM can be divided into the following substeps. This is called the W-ECM, which is an \( α \)-weighted generalization of the ECM (Expectation and Conditional Maximization [18]).

[W-ECM]

**CM-step at k :**
Let \( ϕ = (ϕ_1,\ldots,ϕ_S)^T \) and \( ψ = (ψ_1,\ldots,ψ_S)^T \).

Sub-step at \( k + (s/S) \):
Given \( ϕ = (ϕ_1^+,\ldots,ϕ_{s-1}^+,ϕ_s,ϕ_{s+1},\ldots,ϕ_S)^T \), choose \( ψ_{s}^+ \) such that
\[ Q_{X|Y}^{(α)}(ϕ_1^+,\ldots,ϕ_{s-1}^+,ψ_s^+,ϕ_{s+1},\ldots,ϕ_S)^T \]
\[ ≥ Q_{X|Y}^{(α)}(ϕ_1^+,\ldots,ϕ_{s-1}^+,ψ_s,ϕ_{s+1},\ldots,ϕ_S) \]
\[ D_R^{(\alpha)}(p||q) = -\frac{4}{1-\alpha^2} \log \left[ \int \frac{p(q/p)}{p} \frac{1+\alpha}{2} dx \right] \geq 0. \]  

(1)

A plain \( \alpha \)-divergence appeared later [2], [3], [5], [8] has the same core part:

\[ D_P^{(\alpha)}(p||q) = \frac{4}{1-\alpha^2} \left\{ 1 - \int \frac{p(q/p)}{p} \frac{1+\alpha}{2} dx \right\} \geq 0. \]  

(2)

There is a direct relationship such that

\[ D_R^{(\alpha)}(p||q) = \frac{4}{1-\alpha^2} \log \left\{ 1 - (1 - \alpha^2) \frac{D_P^{(\alpha)}(p||q)}{4} \right\}. \]

Therefore, the Rényi’s \( \alpha \)-divergence \( D_R^{(\alpha)}(p||q) \) and the plain \( \alpha \)-divergence \( D_P^{(\alpha)}(p||q) \) are equivalent in the sense of optimization due to the monotonicity of the logarithm. Our idea actually started from \( D_P^{(\alpha)}(p||q) \). In the following sections, however, \( D_P^{(\alpha)}(p||q) \) is used in the derivation of the WEM algorithm. This is due to its simplicity of the form.

2.2 More General Divergence and Extended Logarithm.

Csiszár [3] presented a general divergence measure (its equivalent form had been given in Rényi’s work [20]):

\[ D_C(p||q) = \int f(x) p(q/x) dx = \int g(x) q(p/x) dx, \]

where \( f \) and \( g \) are twice differentiable convex functions with \( f(1) = g(1) = 0 \). Then, the plain \( \alpha \)-divergence \( D_P^{(\alpha)}(p||q) \) is the case of

\[ f(r) = \frac{4}{1-\alpha^2}(r - r^{\frac{1+\alpha}{\alpha}}) \]  

(3)

with \( r = q/p \). If \( \alpha = -1 \), then \( D_P^{(\alpha)}(p||q) \) is reduced to the well-known Kullback-Leibler divergence:

\[ K(p||q) = D_P^{(-1)}(p||q) = \int p \log(p/q) dx. \]

The case of \( \alpha = 0 \) is reduced to

\[ D_P^{(0)}(p||q) = 2 \int \sqrt{p} - \sqrt{q}^2 dx, \]

which is also well-known as the Hellinger’s distance (e.g., [2]). Note that both \( D_R^{(\alpha)}(p||q) \) and \( D_P^{(\alpha)}(p||q) \) are skew symmetric with respect to \( \pm \alpha \), i.e.,

\[ D^{(\alpha)}(p||q) = D^{(-\alpha)}(q||p). \]

This relationship at \( \alpha = \pm 1 \) is especially well-known.

Considering the above \( f(r) \) of Equation (3) as well as the Kullback-Leibler divergence, one finds that \( \log r \) does not correspond to this \( f(r) \) itself. But, the logarithm rather corresponds to

\[ \frac{2}{1+\alpha} \left\{ r^{\frac{1+\alpha}{2}} + c(r, \alpha) \right\}. \]

The term \( c(r, \alpha) \) is not unique as long as it satisfy a necessary condition: The case of “\( \alpha \to -1 \)” becomes the logarithm in the limit. Note that \( c(r, \alpha) \) can not be zero. We select \( c(r, \alpha) = -1 \) by reflecting computations on probability measures. Then, the quantity

\[ L^{(\alpha)}(r) = \frac{2}{1+\alpha} (r^{\frac{1+\alpha}{2}} - 1) \]

is regarded as an extension of \( \log r \). We call this the \( \alpha \)-logarithm. The case of \( \alpha = -1 \) in the limit is \( \log r \). Note that the constant “\(-1\)” in Equation (4) is often cancelled out when \( L^{(\alpha)}(r) \) is differentiated by a contained parameter or when it is compared with other \( L^{(\alpha)}(\theta) \).

3 Extended EM Algorithms Based Upon the \( \alpha \)-Divergence

3.1 Derivation of the WEM Algorithm

In this section, an extended EM algorithm is derived using the \( \alpha \)-logarithm (4) and the nonnegativity of the plain \( \alpha \)-divergence (2). “Plain” is often omitted in the rest of the text. Bayesian probability computation on conditioning is also used in this derivation.

We define data, probability densities, parameters and likelihood ratios. Vector values therein are often denoted by non-bold letters. This is to avoid suffices in bold letters so that equations may look neat. In some specific cases where confusions may occur, however, bold fonts will be used for vectors.

Let \( p_{Y|\varphi}(y|\varphi) \) be the probability density of observed vector data \( y \). Here, the vector \( \varphi \) denotes structure defining the probability densities. Its simplest case is a set of parameters. The set \( \mathcal{Y} = \{ y \} \) is considered to be an incomplete data set. Let a vector \( x \) be complete data which contains unknown parts at observation. Then, the observed incomplete data is expressed by

\[ \mathcal{X}(y) = \{ x | x(y) = y \} \subset \mathcal{X}, \]

where \( y(x) \) is any incomplete observation of \( x \). The simplest case is that \( \varphi \) and \( \psi \) are parameter vectors. Then,

\[ p_{X|\psi}(x|\psi) = \int_{\mathcal{X}(y)} p_{X|Y,\psi}(x|y,\psi) dx \]

is the relationship describing the observation [4]. Then, the conditional probability is given by

\[ p_{X|Y,\psi}(x|y,\psi) = \frac{p_{X|Y}(x|y,\psi)}{p_{Y|\psi}(y|\psi)}. \]  

(5)

Let

\[ R_X^{(\alpha)}(\psi|\varphi) = \left\{ \frac{p_{X|Y}(x|y)}{p_{X|Y}(x|\psi)} \right\}^{\frac{1}{1+\alpha}}. \]

Let

\[ L_X^{(\alpha)}(\psi|\varphi) = L^{(\alpha)} \left( \frac{p_{X|Y}(x|y)}{p_{X|Y}(x|\psi)} \right) = \frac{2}{1+\alpha} \left( R_X^{(\alpha)}(\psi|\varphi) - 1 \right). \]  

(6)

Define corresponding quantities with respect to \( Y \) by

\[ R_Y^{(\alpha)}(\psi|\varphi) = \left\{ \frac{p_{Y|X}(y|x)}{p_{Y|\psi}(y|\psi)} \right\}^{\frac{1}{1+\alpha}}. \]

Let

\[ L_Y^{(\alpha)}(\psi|\varphi) = L^{(\alpha)} \left( \frac{p_{Y|X}(y|x)}{p_{Y|\psi}(y|\psi)} \right) = \frac{2}{1+\alpha} \left( R_Y^{(\alpha)}(\psi|\varphi) - 1 \right). \]  

(7)
(i) Exponential expectation:
\[ E_{\exp(L^{(\alpha)})} \left[ \frac{\partial L^{(\alpha)}}{\partial \varphi} \right] = -E_{\exp(L^{(\alpha)})} \left[ \frac{\partial^2 L^{(\alpha)}}{\partial \varphi \partial \varphi^T} \right] \]
\[ \overset{\text{def}}{=} M^{(\alpha)}_{\exp}(\varphi), \quad (17) \]
Here, the expectation is taken with respect to
\[ \exp\{L^{(\alpha)}(p)\} = \exp\{\frac{2}{1+\alpha}(p^{\frac{1+\alpha}{2}} - 1)\}, \]
where \( p = p_Y|\varphi \). The same relationship holds for
\[ \exp\{\bar{L}^{(\alpha)}(p)\} = \exp\{\frac{2}{1+\alpha}p^{\frac{1+\alpha}{2}}\}. \]
(ii) Plain expectation:
\[ E_p \left[ \frac{1-\alpha}{2} p^{-1/(1+\alpha)} \left( \frac{\partial L^{(\alpha)}}{\partial \varphi} \right) \left( \frac{\partial L^{(\alpha)}}{\partial \varphi^T} \right) \right] = -E_p \left[ p^{-\frac{1+\alpha}{2}} \left( \frac{\partial^2 L^{(\alpha)}}{\partial \varphi \partial \varphi^T} \right) \right] \overset{\text{def}}{=} M^{(\alpha)}(\varphi). \quad (18) \]
Here, \( E_p = E \), i.e., the expectation is taken with respect to the probability density \( p \). It is important to note that both cases are reduced to the Fisher’s information matrix when \( \alpha = -1 \). \( M^{(\alpha)}(\varphi) \) is relatively simple to compute. But, \( M^{(\alpha)}_{\exp}(\varphi) \) is complicated since an exponential of the probability density itself is the base of the expectation. The second one, \( M^{(\alpha)}(\varphi) \), appears later in the discussion of speedup.

Next, we consider relationship of \( \alpha \) to the Cramér-Rao inequality which gives the bound on parameter estimation. Let \( \hat{h}(\varphi) \) be an unknown vector function of \( \varphi \) which specifies a statistical model \( p_X|\varphi(\varphi|\varphi) \). Let \( \hat{h}(X) \) be an unbiased estimate for \( h(\varphi) \). Let
\[ V_\varphi(\hat{h}(X)) = [C_\varphi(\hat{h}(X), \hat{h}(X))] \]
be a covariance matrix. Let \( \Omega(\varphi) = \partial h(\varphi) / \partial \varphi^T \). Then, the following relationship of \( \alpha \) to the Cramér-Rao bound is obtained.

**[\( \alpha \)-Version of the Matrix Cramér-Rao Inequality]**

The covariance matrix \( V_\varphi(\hat{h}(X)) \) satisfies the following matrix inequality.
\[ V_\varphi(\hat{h}(X)) \geq \frac{1-\alpha}{2} \Omega(\varphi) \left[ M^{(\alpha)}(\varphi) \right]^{-1} \Omega(\varphi)^T, \]
\[ = \Omega(\varphi) \left[ M^{(-1)}(\varphi) \right]^{-1} \Omega(\varphi)^T \quad (19) \]
Here, the matrix version of “\( \geq \)” stands for positive semi-definiteness of the difference of two terms. The above equation (19) means that usage of \( \alpha \)-logarithm does not deteriorate nor improve the logarithmic case. Therefore, utilization of \( \alpha \) should be found in speed of maximization, i.e., quickness to learn. Thus, one obtains a ratio such that
\[ m^{(\alpha)} \overset{\text{def}}{=} M^{(\alpha)}(\varphi) \left[ M^{(-1)}(\varphi) \right]^{-1} = \frac{1-\alpha}{2}. \quad (20) \]
This number \( m^{(\alpha)} \) reflects the prior weighting on the usage of information in the source data. Thus, \( m^{(\alpha)} \) can be called the aptitude number \([10], [11], [12], [13]\]. If \( \alpha = -1 \), i.e., \( M^{(\alpha)}(\varphi) = M^{(-1)}(\varphi) \), then this is the case of the logarithm and \( m^{(\alpha)} = 1 \). On the contrary, one obtains \( m^{(\alpha)} = 0 \) if \( \alpha = 1 \). For \( \alpha > 1 \), \( m^{(\alpha)} \) becomes negative. Then, the system uses source data information towards reverse direction.

It is possible to define a complemental value of the aptitude number in the following way.
\[ \tilde{m}^{(\alpha)} = 1 - m^{(\alpha)} = \frac{1+\alpha}{2} \]
This is called the co-aptitude number \([14]\), which will appear in weighting on probabilities.

**4.2 The \( \alpha \)-Efficient Score and the \( \alpha \)-Information Matrix Related to Learning Methods**

(a) Gradient ascent learning
A simple method to find
\[ \psi^* = \arg \max_{\psi \in \Phi} L_Y^{(\alpha)}(\psi|\varphi) \]
is an application of series of gradient ascent operations. This is based on the relationship
\[ L_Y^{(\alpha)}(\varphi_{k+1}) = L_Y^{(\alpha)}(\varphi_k) + \Delta L_Y^{(\alpha)}(\varphi_k). \]
Then, parameter update at the \( k \)-th iteration is
\[ \varphi_{k+1} = \varphi_k + \Delta^{(\alpha)}(\varphi_k) \]
with
\[ \Delta^{(\alpha)}(\varphi_k) = \rho \left[ \frac{\partial L_Y^{(\alpha)}(\varphi_k)}{\partial \varphi} \right]_{\varphi = \varphi_k} \]
where \( \rho \) is a small positive constant called the learning rate. Then, one obtains
\[ \Delta^{(\alpha)}(\varphi) = \rho \left( p_{Y|\varphi}(y|\varphi) \right)^{-\frac{1+\alpha}{2}} \frac{\partial}{\partial \varphi} p_{Y|\varphi}(y|\varphi). \]
The case of \( \alpha = -1 \) is the traditional log-version:
\[ \Delta^{(-1)}(\varphi) = \rho p_{Y|\varphi}(y|\varphi)^{-1} \frac{\partial}{\partial \varphi} p_{Y|\varphi}(y|\varphi) = \rho \frac{\partial \log(\varphi)}{\partial \varphi}. \]
Thus, one obtains
\[ \Delta^{(\alpha)}(\varphi) = \left[ p_{Y|\varphi}(y|\varphi) \right]^{\frac{1+\alpha}{2}}, \]
that is
\[ \Delta^{(\alpha)}(\varphi) = \rho \left[ p_{Y|\varphi}(y|\varphi) \right]^{\frac{1+\alpha}{2}} \Delta^{(-1)}(\varphi) \quad (21) \]
where a probability weighting by the co-aptitude number \( \tilde{m}^{(\alpha)} \) appears. This method is used in Section 8.2.

(b) Newton-Raphson method
Let the \( k \)-th iteration value of the extended logarithm of \( p_{Y|\varphi}(y|\varphi) \) be
\[ L_{\varphi|\varphi}^{(\alpha)}(\varphi_k) = \frac{2}{1+\alpha} \left\{ p_{Y|\varphi}(y|\varphi_k) \right\}^{\frac{1+\alpha}{2}} - 1 \}. \]
The Newton-Raphson method using \( L_Y^{(\alpha)}(\varphi_k) \) and its Hessian matrix is
\[ \varphi_{k+1} = \varphi_k - \left[ \frac{\partial^2 L_Y^{(\alpha)}(\varphi_k)}{\partial \varphi \partial \varphi^T} \right]^{-1} \frac{\partial L_Y^{(\alpha)}(\varphi_k)}{\partial \varphi}. \quad (22) \]
for $s = 1, \ldots, S$.
Subupdate-step:
Replace $\varphi_s$ by $\psi^+_s$.

Note that the ordering $s$ is arbitrary as long as all $\psi^+_s$ are obtained and updated. The W-ECM is used for supervised and unsupervised learning algorithms in Sections 5.2 and 7.2.

There is a further sophistication on the maximization step. In the maximization step, $\psi^+_s$ can be chosen so that $L_Y^{(\alpha)}(\psi|\varphi)$ is directly maximized. This is the ECME (Expectation and Conditional Maximization Either [9]; a technical flaw is corrected in [17]). Our $\alpha$-version, say the W-ECME, is as follows.

[W-ECME]
CME-step for $Q^{(\alpha)}$:
Repeat the following substeps for $s = 1, \ldots, T$, $(T < S)$.
Substep at $k + (s/S)$:
Given $\varphi = (\varphi^+_1, \ldots, \varphi^+_{s-1}, \varphi_s, \varphi_{s+1}, \ldots, \varphi_S)^T$, choose $\psi^+_s$ such that
\[
Q_{X|Y}^{(\alpha)}(\varphi^+_1, \ldots, \varphi^+_{s-1}, \psi^+_s, \varphi_{s+1}, \ldots, \varphi_S) \geq Q_{X|Y}^{(\alpha)}(\varphi^+_1, \ldots, \varphi^+_{s-1}, \varphi_s, \varphi_{s+1}, \ldots, \varphi_S).
\]

Subupdate-step:
Replace $\varphi_s$ by $\psi^+_s$.
CME-step for $L_Y^{(\alpha)}$:
At the substep $k + (T + 1)/S$, \[
\varphi = (\varphi^+_1, \ldots, \varphi^+_{T+1}, \varphi_{T+2}, \ldots, \varphi_S)^T
\]
is given. Then, obtain $\psi^+_{T+1}$ such that
\[
L_Y^{(\alpha)}(\psi^+|\varphi) \geq L_Y^{(\alpha)}(\varphi|\varphi)
\]
where
\[
\psi^+ = (\varphi^+_1, \ldots, \varphi^+_{T+1}, \varphi_{T+2}, \ldots, \varphi_S)^T.
\]

Then, replace $\varphi$ by $\psi^+$.

Note that it may be possible to update multiple parameters at the CME for $L_Y^{(\alpha)}$, if parameters' independence is guaranteed.

3.3 Change of Conditioning, Cyclic WEM and P-WEM

In Section 3.1, the $\alpha$-divergence between $p_{X|Y,\varphi}$ and $p_{X|Y,\varphi_{\xi}}$ was computed in the world of $p_{X|Y,\varphi}$. The same holds by replacing the conditioning. That is, we can compute the $\alpha$-divergence between $p_{Y|X,\zeta}$ and $p_{Y|X,\zeta}$ in the world of $p_{Y|X,\zeta}$ [10], [11], [12], [13]. Then, for $Q_{X|Y}^{(\alpha)}(\xi|\zeta)$ and $S_{X|Y}^{(\alpha)}(\xi|\zeta)$, equivalent versions of the WEM's can be obtained. Then, the pair $\{Q_{X|Y}^{(\alpha)}(\psi|\varphi), Q_{X|Y}^{(\alpha)}(\xi|\zeta)\}$, or equivalently, $\{S_{X|Y}^{(\alpha)}(\psi|\varphi), S_{X|Y}^{(\alpha)}(\xi|\zeta)\}$ generates a cyclic WEM. Special cases of this pair are given by $(\xi, \zeta) = (\psi, \psi)$ or $(\varphi, \psi)$. The cyclic nature of the WEM algorithm can be observed more clearly in such cases. We note that different philosophies related to this structure are given in [1], [22] for the case of $\alpha = -1$ (i.e., the Kullback-Leibler case) and in [23] for non-Kullback case. But, it is important to point out here that there exists a more sophisticated structure called the systolic and monitoring WEM's [10], [11], [12], [13]. The cyclic WEM is just a building block for such systolic and monitoring WEM's. Such discussions are given in Section 9.

There is one more different version on the conditioning in the WEM algorithm. Bayes equations and (9) give
\[
\frac{2}{1+\alpha}_{\text{WEM}} \left\{ \begin{array}{l}
p_{Y|X}(\psi|\varphi) \\
p_{Y|X}(\varphi|\psi)
\end{array} \right\}^{\frac{1+\alpha}{\alpha}} - 1
\]
\[
= \frac{2}{1+\alpha} \int_{X|Y} p_{X|Y,\varphi}(x|y, \varphi) \left\{ \begin{array}{l}
p_{Y|X}(\psi|\varphi) \\
p_{Y|X}(\varphi|\psi)
\end{array} \right\}^{\frac{1+\alpha}{\alpha}} - 1 dx
\]
\[
+ \frac{1-\alpha}{2} \left\{ \begin{array}{l}
p_{Y|X}(\psi|\varphi) \\
p_{Y|X}(\varphi|\psi)
\end{array} \right\}^{\frac{1+\alpha}{\alpha}} D_{P}^{(\alpha)}(\varphi||\psi)
\]
(15)
when $\psi$ and $\varphi$ are considered as random variables. Note that such a case can be found in problems of hyperparameters. Thus, one obtains
\[
S^{(\alpha)}_{\varphi}(\psi|\varphi) = E_{p_{X|Y,\psi}} \left\{ \begin{array}{l}
p_{Y|X}(\psi|\varphi) \\
p_{Y|X}(\varphi|\psi)
\end{array} \right\}^{\frac{1+\alpha}{\alpha}} - 1
\]
and
\[
Q^{(\alpha)}_{\varphi}(\psi|\varphi) = \frac{2}{1+\alpha} \left\{ S^{(\alpha)}_{\psi}(\psi|\varphi) - 1 \right\}
\]
for the WEM algorithm on parameter estimation. Equivalent statements to the WEM-I and WEM-II for this parameter estimation are obtained by using the above $Q^{(\alpha)}_{\varphi}(\psi|\varphi)$ and $S^{(\alpha)}_{\psi}(\psi|\varphi)$. This is called the P-WEM.

4 Various Statistical Measures Related to $\alpha$-Logarithm and WEM

4.1 $\alpha$-Versions of Important Statistical Measures

First, we consider the $\alpha$-log likelihood for the incomplete data.
\[
L_Y^{(\alpha)}(\psi) = \frac{2}{1+\alpha} \left\{ p_{Y|\psi}(y|\psi) \right\}^{\frac{1+\alpha}{\alpha}} - 1
\]
Then, an $\alpha$-efficient score can be defined from the following equation.
\[
L_Y^{(\alpha)}(\psi) = L_Y^{(\alpha)}(\varphi) + \Delta L_Y^{(\alpha)}(\varphi).
\]
That is, the $\alpha$-efficient score is
\[
\frac{\partial L_Y^{(\alpha)}(\varphi)}{\partial \varphi} = p_{Y|\varphi}(y|\varphi) \psi_{\alpha} \frac{2}{1+\alpha} \frac{\partial}{\partial \varphi} \frac{p_{Y|\varphi}(y|\varphi)}{p_{Y|\psi}(y|\psi)}
\]
(16)
Here, $\psi_{\alpha}(\psi)$ is the logarithm of $p_{Y|\varphi}(y|\varphi)$. The number $\partial \psi_{\alpha}(\psi)/\partial \varphi$ is the original Fisher’s efficient score. Hereafter, $L_Y^{(\alpha)}(\varphi)$ is denoted simply by $L^{(\alpha)}$ or $L$, and $p_{Y|\varphi}$ by $p$ depending on emphases.

We can define various extensions of the Fisher’s information matrix for the $\alpha$-version. We list up the following two versions.

— 264 —
Note that "Ψ" stands for all parameters in $g_i$ as well as $p_{Y|X, \psi_i}$ for all $i$. It is also possible to discuss the case where the network has multiple layers [6], [7].

Next, we compute the $\alpha$-log likelihood ratio for the structure of

$$C = \{I, E\} = \{(X, Y), (X, Y)\}.$$  

The probability density for the complete data is as follows.

$$p_C(Y, Z|X, \Psi) = \prod_{t=1}^{N} p(y^{(t)}|x^{(t)}, \psi)$$

$$= \prod_{t=1}^{N} \prod_{j=1}^{K} \{g_j(x^{(t)}, \psi_j)p(y^{(t)}|x^{(t)}, \psi_j)\} z_j^{(t)}.$$  

Here, $p_{Y|Z|X, \Psi}$ is simply denoted by $p$ to avoid complicated notations. This should be safe since variables are clearly specified. Similarly simplified notations will be used for other probabilities. Next, we obtain the incomplete data probability:

$$p_C(Y, Z|X, \Psi) = \int Z \ p_C(Y, Z|X, \Psi) dZ$$

$$= \prod_{t=1}^{N} p(y^{(t)}|x^{(t)}, \psi).$$

Then, the $\alpha$-version of the complete data likelihood ratio is

$$L_C^{(\alpha)}(Y, Z|X, \Psi, \Phi) = \frac{2}{\alpha + 1} \left( \frac{p_C(Y, Z|X, \Psi)}{p_C(Y, Z|X, \Psi)} \right)^{\frac{1+\alpha}{2}} - 1,$$

and the $\alpha$-version of the complete data likelihood ratio is

$$L_C^{(\alpha)}(Y|X, \Psi, \Phi) = \frac{2}{\alpha + 1} \left( \frac{p_C(Y|X, \Psi, \Phi)}{p_C(Y|X, \Psi, \Phi)} \right)^{\frac{1+\alpha}{2}} - 1.$$  

(29)

(30)

Then, the expectation of the complete data-likelihood ratio is

$$Q_{X|Y, \Psi}^{(\alpha)}(\Psi|\Phi) = \mathbb{E}_{p_C(Y, Z|X, \Psi)} \left[ L_C^{(\alpha)}(Y, Z|X, \Psi, \Phi) \right]$$

$$= \frac{2}{\alpha + 1} \left[ \mathbb{E}_{p_C(Y|X, \Psi, \Phi)} (\frac{1+\alpha}{2}) - 1 \right],$$

(31)

where

$$S_{X|Y, \Psi}^{(\alpha)} = \prod_{t=1}^{N} W_t^{(\alpha)}, \quad W_t^{(\alpha)} = \sum_{j=1}^{K} h_j^{(\alpha)}(t),$$

and

$$h_j^{(\alpha)}(t) = \left\{ \frac{g_j(x^{(t)}, \psi_j)p(y^{(t)}|x^{(t)}, \psi_j)}{g_j(x^{(t)}, \psi_j)p(y^{(t)}|x^{(t)}, \psi_j)} \right\}^{\frac{1+\alpha}{2}} h_j^{(t)}.$$  

(32)

Here, $g_j(x^{(t)}, \psi_j)$ is the gating weight of the $j$-th expert module under the assumption that the structural parameter is $\varphi_0$. $h_j^{(t)}$ is its posterior probability

$$h_j^{(t)} = \frac{g_j(x^{(t)}, \psi_0)p(y^{(t)}|x^{(t)}, \psi_0)}{\sum_{i=1}^{K} g_i(x^{(t)}, \psi_0)p(y^{(t)}|x^{(t)}, \psi_i)}.$$  

5.2 W-GEM (W-ECM) by Newton-Raphson and Least Squares

First, we define a normalized value $h_j^{(\alpha)}(t)$.

$$\tilde{h}_j^{(\alpha)}(t) = h_j^{(\alpha)}(t) / W_t^{(\alpha)}, \quad \sum_{j=1}^{K} \tilde{h}_j^{(\alpha)}(t) = 1.$$  

Then, the Newton-Raphson method gives an update value of the $k$-th component for $\psi_0 = (\psi_0, \ldots, \psi_{K})^T$:

$$\psi^{\text{new}}_k = \psi^{\text{old}}_k + \rho \left( R^{(\alpha)}_{\psi_k} \right)^{-1} \xi_k^{(\alpha)}, \quad (k = 1, \ldots, K),$$

(33)

where $0 < \rho \leq 1$ is a learning rate. Other quantities are

$$e_k^{(\alpha)} = S^{(\alpha)} \sum_{t=1}^{N} \left[ \tilde{h}_k^{(\alpha)}(t) - g_k(x^{(t)}) \right] x^{(t)} x^{(t)T}, \quad (k = 1, \ldots, K),$$

and

$$R^{(\alpha)}_{\psi_k} = \sum_{t=1}^{N} g_k(t) \left\{ 1 - g_k(t) \right\} x^{(t)} x^{(t)T} - \frac{1+\alpha}{2} P_1 - \frac{1+\alpha}{2} P_2,$$

(34)

where

$$P_1 = \sum_{t=1}^{N} \tilde{h}_k^{(\alpha)}(t) \left\{ 1 - \tilde{h}_k^{(\alpha)}(t) \right\} x^{(t)} x^{(t)T}$$

(35)

and

$$P_2 = \left[ \sum_{t=1}^{N} \tilde{h}_k^{(\alpha)}(t) - g_k(t) \right] x^{(t)} x^{(t)T}.$$  

The parameter $\psi_j$ for Gaussian probability is updated by computing $\partial Q^{(\alpha)} / \partial \psi_j = 0$. From this equation, one obtains

$$\psi_j = \{ R^{(\alpha)}_{\psi_j} \}^{-1} c_j^{(\alpha)},$$

where

$$c_j^{(\alpha)} = \sum_{t=1}^{N} \tilde{h}_j^{(\alpha)}(t) x^{(t)} \psi_j^{-1} x^{(t)}$$

and

$$R^{(\alpha)}_{\psi_j} = \sum_{t=1}^{N} \tilde{h}_j^{(\alpha)}(t) x^{(t)} \psi_j^{-1} x^{(t)}.$$  

(37)

Similarly, one obtains the update of the variance:

$$\Sigma_{\psi_j} = \left[ \sum_{t=1}^{N} \tilde{h}_j^{(\alpha)}(t) (y^{(t)} - f_j)(y^{(t)} - f_j)^T \right] / \sum_{t=1}^{N} \tilde{h}_j^{(\alpha)}(t).$$

Then, we have the following update steps.

[Step $n+1/3$]

Given $\varphi_0^{[n]}$, $\varphi_j^{[n]}$, $\Sigma_{\psi_j}^{[n]}$, $(k = 1, \ldots, K)$, $j = 1, \ldots, K)$, compute $\varphi_0^{[n+1/3]} = \varphi_0^{[n]}$ and set $\Sigma_{\psi_j}^{[n+1/3]} = \Sigma_{\psi_j}^{[n]}$, $(j = 1, \ldots, K)$.

[Step $n+2/3$]

Given $\varphi_0^{[n+1/3]}$, $\varphi_j^{[n+1/3]}$, $\Sigma_{\psi_j}^{[n+1/3]}$, $(k = 1, \ldots, K)$, $j = 1, \ldots, K)$, compute $\varphi_j^{[n+2/3]} = \psi_j^{[n]}$, $(k = 1, \ldots, K)$, $j = 1, \ldots, K)$. Then, set $\varphi_0^{[n+2/3]} = \varphi_0^{[n+1/3]}$ and set $\Sigma_{\psi_j}^{[n+2/3]} = \Sigma_{\psi_j}^{[n+1/3]}$, $(j = 1, \ldots, K)$.  

[Step $n+1$]

Given $\varphi_0^{[n+2/3]}$, $\varphi_j^{[n+2/3]}$, $\Sigma_{\psi_j}^{[n+2/3]}$, $(k = 1, \ldots, K)$, $j = 1, \ldots, K)$, compute $\Sigma_{\psi_j}^{[n+1]} = \Sigma_{\psi_j}^{[n]}$. Then, set $\varphi_0^{[n+1]} = \varphi_0^{[n+2/3]}$ and set $\varphi_j^{[n+1]} = \varphi_j^{[n+2/3]}$, $(j = 1, \ldots, K)$.  

Note that there can be two cycles: $\psi_j^{[n]} \rightarrow \psi_j^{[n]} \rightarrow \Sigma_{\psi_j}^{[n]}$ and $\psi_j^{[n]} \rightarrow \psi_j^{[n]} \rightarrow \Sigma_{\psi_j}^{[n]}$. Either works well.
Here, $P_k^{(o)}$ is a notation to emphasize the iteration count $k$. Thus, the Hessian matrix changes the direction of vector $\varphi_k$ depending on the parameter $\alpha$. This property will contribute to speedup of learning, whose example will be given in Sections 8.3 and 8.4.

Next, we estimate appropriate parameter range by taking an expectation of the Hessian matrix. If the expectation is taken to the Hessian matrix itself, one obtains

$$\varphi_{k+1} = \varphi_k - \left[ E \left\{ \frac{\partial^2 L^{(o)}}{\partial \varphi \partial \varphi^T} \right\} \right]^{-1} \frac{\partial L^{(o)}}{\partial \varphi}. \quad (23)$$

But, in order to compare with the case $\alpha = -1$, we modify Equation (22) by using

$$\frac{\partial L^{(o)}}{\partial \varphi} = p_k^{1+\alpha} \frac{\partial f_k}{\partial \varphi}.$$

Then, we have

$$\varphi_{k+1} = \varphi_k - \left[ \frac{\partial^2 L^{(o)}}{\partial \varphi \partial \varphi^T} \right]^{-1} \frac{1+\alpha}{p_k} \frac{\partial f_k^2}{\partial \varphi} = \varphi_k - \left[ \frac{1+\alpha}{p_k} \left( \frac{\partial^2 L^{(o)}}{\partial \varphi \partial \varphi^T} \right) \right]^{-1} \frac{\partial f_k^2}{\partial \varphi}.$$

Therefore, the expectation version to compare directly with the logarithmic case is

$$\varphi_{k+1} = \varphi_k - \left[ \frac{1+\alpha}{p_k} \left( \frac{\partial^2 L^{(o)}}{\partial \varphi \partial \varphi^T} \right) \right]^{-1} \frac{\partial f_k^2}{\partial \varphi} = \varphi_k + \left[ M_k^{(o)}(\varphi) \right]^{-1} \frac{\partial f_k^2}{\partial \varphi}, \quad \text{(24)}$$

Therefore, speedup of learning is obtained for $\alpha > -1$. It is important to emphasize that what we use corresponds to Equations (22) and (23) where vector directions are definitely changed from the case of $\alpha = -1$ depending on the number $\alpha$. Equation (24) was derived to obtain a reasoning that there is a merit in the range of $\alpha > -1$.

5 Learning for Neural Networks on Mixture Densities

5.1 $Q^{(\alpha)}$ for Neural Networks of Expert Mixtures

In the neural network via mixture of experts [6], [7], the random variable $X^{(t)}$ stands for an input, $Y^{(t)}$ is a teacher signal, and $Z^{(t)}$ indicates a path in the hierarchy. The value of $Z^{(t)}$ is either zero or one. The superfix $t$ indicates one of $N$ data. For the expectation and maximization structure, the incomplete information given by measurements is $I = \{X^{(t)}, Y^{(t)}\}$, where the lower case characters stand for actually observed values. By denoting $Z = \{z^{(t)}\}$, the complete information is expressed by $C = \{I, Z\}$. Thus, WEM-I and the neural networks here have the correspondence in Table 5.1.

<table>
<thead>
<tr>
<th>data</th>
<th>WEM-I</th>
<th>mixture-of-experts NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>incomplete $I$</td>
<td>$Y$</td>
<td>$({X^{(t)}}, {Y^{(t)}})$</td>
</tr>
<tr>
<td>missing $Z$</td>
<td>implicit</td>
<td>$({Z^{(t)}})$</td>
</tr>
<tr>
<td>complete $C$</td>
<td>$X$</td>
<td>$({X^{(t)}}, {Y^{(t)}}, {Z^{(t)}})$</td>
</tr>
</tbody>
</table>

Fig. 1 illustrates a mixture-of-experts neural network with one layer [6], [7].

![Fig. 1 Neural network of expert mixtures.](image)

The mixture probability is

$$p_{Y|X, \psi}(y^{(t)}|x^{(t)}, \psi) = \sum_{i=1}^{K} p_{Z|X, \psi}(z^{(t)} = i|x^{(t)}, \psi_0)p_{Y|X, \psi}(y^{(t)}|x^{(t)}, \psi_i)$$

$$= \sum_{i=1}^{K} g_i(x^{(t)}|\psi)p_{Y|X, \psi}(y^{(t)}|x^{(t)}, \psi_i), \quad \text{(25)}$$

with

$$p_{Y|X, \psi}(y^{(t)}|x^{(t)}, \psi_i) = \frac{1}{(2\pi)^{n/2}V_i^{-1/2}} \times \exp\left[-\frac{1}{2} (y^{(t)} - \mu_i^{(t)})^T \Sigma_i^{-1} (y^{(t)} - \mu_i^{(t)}) \right], \quad \text{(26)}$$

where

$$\mu_i^{(t)} = f_i(x^{(t)}, \psi_i) \quad \text{(27)}$$

with

$$f_j(x^{(t)}, \psi_j) = X^{(t)} T \psi_j$$

and

$$X^{(t)} T = \left[ \text{diag}(x^{(t)} T) | I \right].$$

The gating probability $g_i$ for the first layer is

$$g_i(x^{(t)}|\psi_i) = p_{Z|X, \psi}(z^{(t)} = i|x^{(t)}, \psi_0) = \frac{e^{x^T e}}{\sum_{m=1}^{K} e^{x^T e}}, \quad \text{(28)}$$

Expectation of the network’s output is

$$\mu^{(t)} = \sum_{i=1}^{K} g_i \mu_i^{(t)}.$$
7.1 Clustering

The EM-algorithm has been applied also to unsupervised learning such as clustering [7], [24]. Here, we use the WEM on this clustering problem. In this case, the teacher signal $Y^{(t)}$ is absent. Therefore, mixture coefficients $\gamma_j$ appear instead of $g_j(x^{(t)}, \psi_j)$, $j = 1, \ldots, K$.

Since $\sum_{j=1}^K \gamma_j = 1$, one obtains using a Lagrange multiplier that

$$\gamma_j = \frac{\sum_{i=1}^N h_j^{(t)}/N}{\sum_{i=1}^N h_j^{(t)}}.$$

The probability model is as follows.

$$p(x^{(t)}|\psi_j) = \exp\left\{-\frac{1}{2} (x^{(t)} - \mu_j)^T \Sigma^{-1}_{\psi_j} (x^{(t)} - \mu_j)\right\} / \{2\pi\}^{d/2}/m|\Sigma_{\psi_j}|^{1/2} \}

Then, the maximization function is as follows.

$$Q^{(n)}(\psi|\Phi) = \frac{2}{1+\alpha} \left[ \prod_{i=1}^N \sum_{j=1}^K h_j^{(t)} \left\{ \frac{\gamma_j p(x^{(t)}|\psi)}{\sum_j \gamma_j p(x^{(t)}|\psi_j)} \right\}^{1+\alpha} - 1 \right].$$

Here,

$$h_j^{(t)} = \frac{\gamma_j p(x^{(t)}|\psi_j)}{\sum_j \gamma_j p(x^{(t)}|\psi_j)}.$$

7.2 W-GEM (W-ECM) for Clustering

By using the above $Q^{(n)}(\psi|\Phi)$, we have a set of update equations. They are used as follows.

**[Step n+1/3]**

Given $\gamma_j^{[n]}$, $\mu_j^{[n]}$, $\Sigma_j^{[n]}$, $(k = 1, \ldots, K; j = 1, \ldots, K)$, compute $\gamma_j^{[n+1/3]}$. Then, set $\mu_j^{[n+1/3]} = \mu_j^{[n]}$ and set $\Sigma_j^{[n+1/3]} = \Sigma_j^{[n]}$, $(j = 1, \ldots, K)$.

**[Step n+2/3]**

Given $\gamma_j^{[n+1/3]}$, $\mu_j^{[n+1/3]}$, $\Sigma_j^{[n+1/3]}$, $(k = 1, \ldots, K; j = 1, \ldots, K)$, compute $\mu_j^{[n+2/3]}$, $(k = 1, \ldots, K; j = 1, \ldots, K)$. Then, set $\gamma_j^{[n+2/3]} = \gamma_j^{[n+1/3]}$ and set $\Sigma_j^{[n+2/3]} = \Sigma_j^{[n+1/3]}$, $(j = 1, \ldots, K)$.

**[Step n+1]**

Given $\gamma_j^{[n+2/3]}$, $\mu_j^{[n+2/3]}$, $\Sigma_j^{[n+2/3]}$, $(k = 1, \ldots, K; j = 1, \ldots, K)$, compute $\Sigma_j^{[n+1]}$. Then, set $\gamma_j^{[n+1]} = \gamma_j^{[n+2/3]}$ and set $\mu_j^{[n+1]} = \mu_j^{[n+2/3]}$, $(j = 1, \ldots, K)$.

Note that there are two cycles. One is $\gamma_j \rightarrow \psi_j \rightarrow \Sigma_j$. The other is $\psi_j \rightarrow \gamma_j \rightarrow \Sigma_j$. Either works properly.

8 Experiments on Supervised and Unsupervised Learning

8.1 Source Data and Their Estimation

We prepare the following problem of source data estimation in order to identify effects of the probability weighting by $\alpha$.

**[Test Data]**

Generate raw data by

$$y_n = |x_n| + \varepsilon_n \quad \text{with} \quad \varepsilon_n \in \mathcal{N}(0, (0.05)^2)$$

for $x_n \in [-1.0, 1.0]$. Then, normalize $y_n$ so that max $y_n = 1$ and min $y_n = 0$. Rename the resulting data to be $\{x^{(t)}\}$. This is the input data set. Teacher data set is the case of $\varepsilon_n = 0$. The expectation of the output is

$$\mu^{(t)} = \mu_1^{(t)} + \mu_2^{(t)}$$

with

$$\mu_i^{(t)} = u_{i1} x^{(t)} + u_{i2}, \quad i = 1, 2,$$

and

$$g_i^{(t)} = \frac{e^{v_{i1} x^{(t)}}}{e^{v_{i1} x^{(t)}} + e^{v_{i2} x^{(t)}}}, \quad i = 1, 2.$$

**[Problem]**

Estimate true data generation mechanism by using $\mu^{(t)}$. That is, estimate the optimal parameter set

$$\psi^* = \varphi^* = \left[ \{u_{i,j}^{*}\}_{i,j=1}^2 \right], \left[ \{v_{i}^*\}_{i=1}^2 \right].$$

**[Experiment]**

Estimation errors in the course of training were measured by the root mean square error (RMSE). This is because $Q^{(n)}$ differs depending on $\alpha$ and does not appear in the case of the gradient ascent.

8.2 Gradient Ascent Learning

Gradient ascent learning updates parameters by Equation (21) with the replacement of $Y \rightarrow (X, Y)$; c.f., Table 5.1. Note that the variance can be exactly optimized in learning steps.

Fig. 2. RMSE for gradient ascent learning with various $\alpha$.

Fig. 2 is a learning curve for different $\alpha$. We can observe that $\alpha = -0.98$, $m^{(0)} = 0.99$, is slow and oscillates. Surprisingly enough is the trend of $\alpha = -1$, $m^{(0)} = 1.0$, which is the case of the traditional logarithm. Convergence is still slow and oscillates. On the other hand, the case of $\alpha = -1.02$, $m^{(0)} = 1.01$, is fast and no oscillation occurs around the convergence.
6 Convergence Properties

6.1 Positive Definiteness of Update Matrices

(i) Positive definiteness of $R_{g\alpha}$:

By observing Equations (34)~(36), one obtains the following fact: There exists $\beta^* > 0$ such that $R_{g\alpha}$ is positive definite with probability one for $\alpha < -1 + \beta^*$ as $N$ tends to infinity. The same holds for $\{R_{g\alpha}\}^{-1}$.

(ii) Positive definiteness of $R_{\psi\psi}$:

The matrix $R_{\psi\psi}$ is positive definite with probability one as $N$ tends to infinity because of the form (37).

(iii) Positive definiteness of $\Sigma_{\psi\psi}$ and related matrices:

From Equation (38), one obtains $\Sigma_{\psi\psi}$ is positive definite with probability one as $N$ tends to infinity. Using Equation (38) again, one obtains

$$\Sigma_{\psi\psi}^{[\text{new}]} - \Sigma_{\psi\psi}^{[\text{old}]} = \frac{2}{\sum_{i=1}^{N} \lambda_i^{(\alpha)}(t)} \times \Sigma_{\psi\psi}^{[\text{old}]} \frac{\partial Q^{(\alpha)}}{\partial \psi_j} \Sigma_{\psi\psi}^{[\text{old}]}.$$

Next, we consider a column-piled vector $v[\cdot]$ of $\Sigma_{\psi\psi}^{[\text{new}]} - \Sigma_{\psi\psi}^{[\text{old}]}$. Then, one obtains

$$v[\Sigma_{\psi\psi}^{[\text{new}]}] - v[\Sigma_{\psi\psi}^{[\text{old}]}] = R_{\Sigma_j}^{(\alpha)} \frac{\partial Q^{(\alpha)}}{\partial \Sigma_j}.$$ 

Here,

$$R_{\Sigma_j}^{(\alpha)} = \sum_{i=1}^{N} \frac{2}{\lambda_i^{(\alpha)}(t)} \{\Sigma_{\psi\psi}^{[\text{old}]} \otimes \Sigma_{\psi\psi}^{[\text{old}]}\}.$$ 

The operator `$\otimes$' stands for the Kronecker product. Thus, one obtains the following claim: The matrix $R_{\Sigma_j}^{(\alpha)}$ is positive definite with probability one as $N$ tends to infinity.

(iv) Positive definiteness of over-all matrix:

Next, observe that a big parameter vector is as follows:

$$\Psi = \text{col} \{\psi_0, \psi_1, \ldots, \psi_K, v[\Sigma_{\psi\psi}], \ldots, v[\Sigma_{\psi\psi}]\}.$$ 

Define a big matrix

$$H^{(\alpha)}(\Psi_n|\Psi_{n-1}) = \text{diag}\{R_{g\alpha}^{(\alpha)}\}^{-1}, \ldots, \{R_{g\alpha}^{(\alpha)}\}^{-1},$$

$$\{R_{\psi\psi}^{(\alpha)}\}^{-1}, \ldots, \{R_{\psi\psi}^{(\alpha)}\}^{-1}, R_{\Sigma_j}^{(\alpha)} \ldots, R_{\Sigma_j}^{(\alpha)}.$$ 

Then, the matrix $H^{(\alpha)}(\Psi_n|\Psi_{n-1})$ is positive definite for $\alpha < -1 + \beta^*$, $\beta^* > 0$ with probability one as $N$ tends to infinity.

6.2 Differentiation of $Q^{(\alpha)}$ With Respect to Parameter Vectors and Matrices

(i) Differentiation with respect to $\psi_0$:

Since

$$\frac{\partial Q^{(\alpha)}}{\partial \psi_0} = e_0^{(\alpha)} [\text{old}],$$

one obtains

$$\psi_0^{[\text{new}]} - \psi_0^{[\text{old}]} = \rho R_{g\alpha}^{(\alpha)} \frac{\partial Q^{(\alpha)}}{\partial \psi_0}.$$ 

(ii) Differentiation with respect to $\psi_j$:

Since

$$\frac{\partial Q^{(\alpha)}}{\partial \psi_j} = e_j^{(\alpha)} - R_{\psi\psi}^{(\alpha)} \psi_j^{[\text{old}]},$$

one obtains

$$\psi_j^{[\text{new}]} - \psi_j^{[\text{old}]} = R_{\psi\psi}^{(\alpha)} \frac{\partial Q^{(\alpha)}}{\partial \psi_j}.$$ 

(iii) Differentiation with respect to $\Sigma_{\psi\psi}$:

We had the relationship

$$v[\Sigma_{\psi\psi}^{(\alpha)}] - v[\Sigma_{\psi\psi}^{(\alpha)}] = R_{\Sigma_j}^{(\alpha)} \frac{\partial Q^{(\alpha)}}{\partial \Sigma_{\psi\psi}}.$$ 

in the previous section.

6.3 Proof of W-GEM (W-ECM) property

Here, we prove that the algorithm given in Section 5.2 has the W-GEM property.

[Proposition 1]

There exists a number $\beta^* > 0$ such that the algorithm of Section 5.2 is W-GEM (W-ECM) for $\alpha < -1 + \beta^* \leq 1$ with probability one as $N$ tends to infinity.

[Proof]

We have obtained that $H^{(\alpha)}(\Psi|\Phi)$ is positive definite with respect to $\Psi$ for $\alpha < -1 + \beta^* \leq 1$ with probability one as $N$ tends to infinity. One obtains that

$$\Delta \Psi_n = H^{(\alpha)}(\Psi_n|\Psi_{n-1}) \nabla Q^{(\alpha)}(\Psi_n|\Psi_{n-1}),$$

where

$$\nabla Q^{(\alpha)}(\Psi_n|\Psi_{n-1}) = \nabla Q^{(\alpha)}(\Psi_n|\Psi_{n-1})|_{\Psi = \Psi_n}.$$ 

Then

$$\nabla Q^{(\alpha)}(\Psi_n|\Psi_{n-1})^T \Delta \Psi_n = \nabla Q^{(\alpha)}(\Psi_n|\Psi_{n-1})^T H^{(\alpha)}(\Psi_n|\Psi_{n-1}) \nabla Q^{(\alpha)}(\Psi_n|\Psi_{n-1}) \geq 0.$$ 

The equality holds if and only if $\nabla Q^{(\alpha)}(\Psi_n|\Psi_{n-1}) = 0$. Thus, the vector $\Delta \Psi_n = \Psi_{n+1} - \Psi_n$ keeps the direction of increasing the performance function $Q^{(\alpha)}(\Psi_n|\Psi_{n-1})$. Note that one obtains that

$$Q^{(\alpha)}(\Psi_{n+1}|\Psi_n) \geq 0 = Q^{(\alpha)}(\Psi_n|\Psi_n).$$

This means that the maximization algorithm using $H^{(\alpha)}(\Psi_n|\Psi_{n-1})$ is a W-GEM. Hence, this algorithm converges to a stationary point for $\alpha < -1 + \beta^* \leq 1$, $(0 < \beta^* \leq 2)$ with probability one as $N$ tends to infinity.
layer can fork. This can be regarded as attaching a monitor such as in Fig. 6. In this case, even/odd clocking is generalized to mutual exclusion mechanism. That is, multiple directions of the streams avoid to be active at the same time in the same sub-world. There can be a learning scheduler, say $\mathcal{A}$, that orders the activity of modules. The scheduler $\mathcal{A}$ is proper if it requests every subsystem to work infinitely often.

![Fig. 6 Monitoring WEM structure.](image)

### 10 Concluding Remarks

On the convergence and speed, we can summarize our observations as follows.

(i) We proved the convergence of the WEM's. If we put additional assumptions, it is possible to prove the convergence to a local maximum. However, it is necessary to assume convexity around the stationary point. But, this assumption is impractical since the stationary point per se is unknown. This situation has been the same as logarithmic cases.

(ii) Since we are given only finite data, “convergence with probability one as $N$ tends to infinity” is almost always met.

(iii) A condition

\[ Q(\alpha)[n] < \epsilon \]

for a small $\epsilon$ is a possible stopping rule. This is because the Hessian matrix becomes always singular at a stationary point. This is because the stationary point is defined by the singularity of the Hessian.

(iv) For gradient ascent learning, the case of $\alpha < -1$ gives superior convergence to the the logarithmic case ($\alpha = -1$).

(v) For the WEM’s, $\alpha > -1$ is superior to the logarithmic case ($\alpha = -1$) in the sense of iteration count. Usually, computation of $Q(\alpha)$ by $\alpha^x$ takes more CPU time than $Q^{(-1)}$ using log x. The ratio of two CPU times depends on both software refinement and hardware. So far, we give the following rule of thumb:

\[
\frac{\text{computation by general } \alpha}{\text{computation by logarithm}} = 1.6.
\]

Thus, the case of Fig. 3 definitely supports the superiority of the WEM (W-GEM) in every sense.

(vi) The speedup of the WEM (W-GEM) is due to the change of vector directions by the Hessian matrix.

### Acknowledgements

The author is indebted to Professor K. Akitzuki for his encouragement to submit this paper. The author wishes to thank anonymous referees for their comments to improve this paper. The author is grateful to his

---

**References**


region. The reason is that the probability weighting
by \[ \prod_t p_Y|X_t, \rho(\gamma(t)|\alpha) \] emphasizes penalty
on outliers. That is, penalties on low probability data
are emphasized. On the other hand, high probability
data where the convergence is easy are less emphasized.
Thus, a good proportion of both penalties lead to a fast
and non-oscillating convergence.

8.3 Estimation of Source Data Mechanism via
Supervised Mixture of Experts
In this section, we check to see if speedup property
predicted by the theoretical discussions of Section 4.2
can be observed or not.

![Graph](image)

Fig. 3. RMSE depending on \( \alpha \) with \( \rho = 0.1 \).

Fig. 3 shows learning curves for various \( \alpha \). Contrary
to the gradient ascent, W-GEM is faster if \( \alpha \) is greater
than \(-1\). This was predicted in Section 4.2 since the
inverse of the Hessian matrix is used. The case of
\( \alpha = -1 \) is the traditional log-EM. If \( \alpha \) is increased by
\( \beta = 0.2 \), i.e., \( \alpha = -0.8 \), the convergence is significantly
increased. This learning curve is drawn by a solid line.
Note that \( \alpha = -0.8 \) is close to the limit of the stable
region. On the other hand, decreasing \( \alpha \) from \(-1 \) gives
slower convergence. This can be observed by the dotted
or dashed learning curves. It is worth noting that the
case of \( \alpha = -0.8 \) outperforms the log case (\( \alpha = -1 \))
not only by the iteration count but also by actual com-
putation time. Further discussions on this is given in
Section 10.

8.4 Unsupervised Learning: Clustering
The algorithm of Section 7.2 was applied to clustering of
Gaussian mixtures. Fig. 4 illustrates the convergence
curve for various parameters. We can observe again
that cases of \( \alpha > -1.0 \) show faster convergence. In
this case, the speedup is less than the supervised ver-
sion. This is because the mixture coefficients \( \gamma_j \) can be

![Graph](image)

Fig. 4. Convergence of clustering depending on \( \alpha \).

9. Systolic Monitoring WEM

9.1 Systolic Layer of WEM
In Section 3.3, we considered the cyclic WEM. But,
this structure is too monolithic to model sophisti-
cated systems. One of such a target is functional
modeling of brains which should have afferent and
efferent paths. Then, it is useful to have block
connection structures. The building block of the
WEM has multiple input/output pairs passing each
other. Therefore, block connections can make a sys-
tolic layer. Fig. 5 illustrates such an example.

![Diagram](image)

Fig. 5 Systolic layer of WEM blocks.

In this figure, processing of \( X_{i-1} \) to generate \( X_i \) in the
sub-world \( \Omega_i \) becomes active at an even clock. On the
other hand, processing of \( Y_{i-1} \) to generate \( Y_i \) in the sub-
world \( \Omega_i \) becomes active at an odd clock. A restricted
version of this structure is obtained by the following
specialization:

\[ \mu_i = \nu_i, \quad Y_i = X_{i-1}, \quad X_N = Y_0 = \lambda \]

and

\[ p(x_N|x_{N-1}, \mu_N) = p(y_0|y_1, \nu_1) = \lambda. \]

Here, \( X_i \) and \( Y_i \) are random outputs. \( \lambda \) and \( \lambda \) stand
for null outputs and null structures, respectively.

9.2 Asynchronous Monitoring WEM
The learning systolic array of Fig. 5 is a basis for the
next sophistication. The main stream of the systolic
Harmonic Competition: A Self-Organizing Multiple Criteria Optimization

Yasuo Matsuyama, Senior Member, IEEE

Abstract—Harmonic competition is a learning strategy based upon winner-take-all or winner-take-quota with respect to a composite of heterogeneous subcosts. This learning is unsupervised and organizes itself. The subcosts may conflict with each other. Thus, the total learning system realizes a self-organizing multiple criteria optimization. The subcosts are combined additively and multiplicatively using adjusting parameters. For such a total cost, a general successive learning algorithm is derived first. Then, specific problems in the Euclidian space are addressed. Vector quantization with various constraints and traveling salesperson problems are selected as test problems. The former is a typical class of problems where the number of neurons is less than that of the data. The latter is an opposite case. Duality exists in these two classes. In both cases, the combination parameters of the subcosts show wide dynamic ranges in the course of learning. It is possible, however, to decide the parameter control from the structure of the total cost. This method finds a preferred solution from the Pareto optimal set of the multiple object optimization. Controlled mutations motivated by genetic algorithms are proved to be effective in finding near-optimal solutions. All results show significance of the additional constraints and the effectiveness of the dynamic parameter control.

I. INTRODUCTION

COMPETITION is an agent selection mechanism based upon a fitness measure. Given an input, each agent computes the fitness of its state to the input. Upon completion of this stage, every agent broadcasts its own figure of fitness to the others. Then, each agent compares its own fitness with the ones received. An agent which has a better fitness than any of the received values can claim to be the winner.

If the agent state is equivalent to an input weight vector, or if the above structure is well-expressed by a directed graph, the total system is called a competitive artificial neural network. Each agent is called an artificial neuron with competition. “Artificial” is mostly omitted if there is no possibility of mistaking them for wet-ware neurons.

Learning by competition is usually winner-take-all. That is, only the winner obtains the right to produce an output and to learn. Learning means a modification of the state vector so that the fitness to the current input is increased. If there are neurons which cooperate with the winner, these comrades can also learn. This is called winner-take-quota. Hereafter, “state vector,” “weight vector,” and “neuron” are interchangeably used.

The measure of fitness is often called the cost or error. In this case, the winner selection is based upon a minimization. If the cost is an error in the input approximation by weight vectors, the learning is equivalent to a phase of clustering or data compression. If we try to treat more sophisticated or real-world problems, it is necessary to use a composite of heterogeneous subcosts. The most important subcost, the main cost, is for the data approximation. The rest of the subcosts are for various constraints. The competition phase computes the values of fitness for all subcosts. The harmonic competition is a winner selection mechanism taking all heterogeneous subcosts into account.

Incorporation of the aforementioned fitting subcosts in the total error measure has enabled the expansion of the problem class to be solvable by learning. There are several studies treating such additional subcosts [3], [9], [15]–[19], [23], [25]. In these cases, different subcosts are additively combined with the main cost. Since the main cost and subcosts are quite different in nature, combination parameters are essential to adjust the subcosts’ dynamic ranges. For instance, the simplest form of such a total cost includes one adjustment parameter, $\lambda$, such that $d = f + \lambda \bar{g}$. Here, $\bar{f}$ is an average cost for the approximation. $\bar{g}$ is a constraint. Except for a few studies, the parameter $\lambda$ per se is fixed throughout the learning. This is meaningful, however, only if a carefully selected value of this parameter is given a priori. Since fixing the parameter is part of the design phase, not specifying this number forces ill-conditioned learning. Therefore, designers of the learning system can not set this parameter until many repeated trials are performed: It is necessary to find a reasonable method for dynamically adjusting such a parameter.

Among the above references, [16]–[18] and [23] report the importance of the dynamic control of the subcost. The strategies presented therein, however, were limited to specific problems. It is desirable to find a more universal method. This is one of the main issues in this paper after a formal derivation of general harmonic competition. It is also worth emphasizing that a strategy of controlled mutation of the weight vectors is theoretically derived from the harmonic competition. Thus, the organization of this paper is as follows: First, general derivation of harmonic competition and learning is given. Then, two different classes of problems are studied as benchmarks. These classes show a beautiful duality. The first is the case where the number of neurons, $M$, is smaller than that of the data, say $N$. Data compression by divergence-constrained vector quantization (DVQ; all for equiprobability, equierror, and joint equiprobability/error) is such a case. From


B. Weight Update for the Euclidean Cost

Here, we compute the case where the term $f_n$ of (3) is Euclidan. Note that the class with additive subcosts is already significantly wide. Thus, this case is discussed first. Multiplicative penalties will be discussed later jointly with the extended vehicle routing problems.

Consider the case where $\lambda_{nk} \equiv \lambda$ and $h_{nk} \equiv 1$; i.e., $K = L = 1$ and the combination parameters are independent of $n$. Then

$$D_n = \sum_{m=0}^{M-1} (\|x_n - w_m\|^2 + \lambda g_n) Q(x_n, w_m).$$

Here, $\| \cdot \|$ is the Euclidean metric. Then, one obtains from (4) that

$$\tilde{d} = \frac{1}{N} \sum_{n=0}^{N-1} D_n$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} \|x_n - w_{m(n)}\|^2 + \lambda \frac{1}{N} \sum_{n=0}^{N-1} \left\{ \sum_{m=0}^{M-1} g_n Q(x_n, w_m) \right\}$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} \|x_n - w_{m(n)}\|^2 + \lambda \frac{1}{N} \sum_{n=0}^{N-1} [g]_{m(n)}$$

$$\equiv \tilde{f} + \lambda \tilde{g} \quad (10)$$

where $[g]_{m(n)}$ means that the winner $w_{m(n)}$ is identified in $g_n$. Then, one obtains from (7) and (8) that

$$\Delta w_{m(n)}^{(t+1)} = \epsilon(t) \left( x_n - w_{m(n)}^{(t)} \right) - \frac{\epsilon(t) \lambda(t) \partial [g]_{m(n)}}{2 \partial w_{m(n)}^{(t)}}. \quad (11)$$

Here, the superscript $t$ for $\epsilon(t)$ and $\lambda(t)$ specifies the values at the $t$th data input. Thus, the harmonic competitive learning is a combination of actions described by (5), (6), (9), and (11). The derivation of the winner $w_{m(n)}^{(t)}$ in (10) requires detailed specifications of $\tilde{g}$. Therefore, this part is discussed in Sections III and IV depending upon the form of $\tilde{g}$.

The control of the parameters $\epsilon(t)$ and $\lambda(t)$ also depends on the class of problems. The learning parameter $\epsilon(t)$ can be controlled by

$$\epsilon(t+1) = \epsilon(t) + \Delta \epsilon(f(t), f^{(t-1)}, \tilde{g}(t), \tilde{g}^{(t-1)}, t).$$

This form includes the case of predefined control. We note here that $\epsilon(t)$ need not monotonically decrease to zero. In some regularization problems, a monotone increase with saturation is even desirable.

In any problem, the subcost parameter $\lambda(t)$ can be fixed provided an appropriate value is known a priori (static rule). This figure, however, can never be given in advance. Thus, the problem itself is ill conditioned. One naive method is to perform repeated experiments changing this value. Since the main cost and the subcost are heterogeneous in nature, many experiments are required to cover the wide dynamic range of $\lambda$. Another method is to use a dynamic rule

$$\lambda(t+1) = \lambda(t) + \Delta \lambda(f(t), f^{(t-1)}, \tilde{g}(t), \tilde{g}^{(t-1)}, t).$$

One will find that this is a method to pick up a preferred solution from the Pareto optimal set using the harmonic competition dynamics.

Starting from Section III, specific strategies of the harmonic competition are given. Treated problems and their significances are as follows:

- The case where the number of neurons, $M$, is smaller than that of the data, $N$, (data compression): Divergence is selected as the subcost for the minimization. Effective competition biases and strategies are obtained in the following cases:
  a) Divergence-constrained vector quantization for equiprobability.
  b) Divergence-constrained vector quantization for equierror.
  c) Divergence-constrained vector quantization for joint equiprobability and equierror.

- The case of $M > N$ (regularization).
  a) Euclidean traveling salesperson problem.
  b) Extended vehicle routing problem (an example of three subcosts with multiplicative constraints).

The duality of the above two cases is an important subject to be observed throughout the text.

III. DATA COMPRESSION WITH CONSTRAINTS

Data compression is the case in which the number of neurons, $M$, is less than that of the data, $N$. In this case,

$$Q(x_n, w_m) = 1$$

means that the input data $x_n$ is approximated by the weight vector $w_m$. Then, each input is expressed by log$_2 M$ bits achieving data compression. Since $x_n$ is a vector, such a case is called vector quantization. Since the input vectors are fed into the learning mechanism one by one, the strategy is called a successive mode. On the other hand, there is a learning method which uses the whole training data set repeatedly [11]. This is called a batch mode. Both successive and batch modes of their plain versions suffer from traps at undesirable local minima. We present methods which use the divergence (Kullback–Leibler number) to make up this deficiency. The divergence can be effective either for output probability equalization (equiprobability) or for output error equalization (equierror).

A. Equiprobability Vector Quantization

The equiprobability constraint is incorporated as a subcost in the case where the usage of the weight vectors $w_m$ $(m = 0, \ldots, M - 1)$, is requested to be uniform. This means that the output entropy is maximized, or equivalently, the output divergence is minimized. Since the divergence includes a target probability, nonuniform distributions can be the design object. In the experiments, only uniform distribution is treated since there is no specific requirement for nonuniformity in applications so far.

Let $p_m$ be the probability that the weight vector $w_m$ is selected. Then, $p_m$ is the expectation of $Q$: $p_m = E[Q(x_n, w_m)]$. Denote $\{p_m\}_{m=0}^{M-1} \equiv P$. Let $Q = \{q_m\}_{m=0}^{M-1}$
the competition bias, a strategy of weight mutation is obtained. The second is the opposite case: \( M \) is larger than \( N \). A Euclidian traveling salesman problem (TSP) and extended vehicle routing problems (EVRP) with sets of real-world data are such cases. The EVRP is a case containing all types of constraints. Through general discussions and experiments with the above classes, the following can be claimed:

- Harmonic competition is an eligible strategy to find a preferred solution to the Pareto optimal set (noninferior solution set) [12, pp. 331–332] of the multiple criteria optimization. That is, this learning mechanism picks up a trade-off between the main cost and subcosts.
- The parameter \( \lambda \) has quite a wide dynamic range. Setting this parameter requires some sort of prior knowledge. Therefore, the subcosts should be controlled so that their ratio to the main cost changes slowly. This adjustment covers a wide dynamic range of the parameter \( \lambda \). It gives a better, or at least, a comparable performance to the static method which relies upon a priori information.
- The weight mutation can be derived from the competition bias. This method helps to avoid bad local minima.
- The rule presented is applicable to the case of multiple constraints. A wide variety of problems can be cast in the learning algorithm presented.

II. COST WITH PENALTIES

As was introduced in Section I, harmonic competition is a phase of finding a learning neuron for the optimization of the total cost. The total cost is made up of two parts: the main cost and subcosts. The main cost has the role of measuring the degree of data approximation. The subcosts are selected according to the nature of the problem to be solved. Ubiquitous applications use only the main cost. In what follows, however, we present a class of costs whose usage will widely expand the class of problems solvable by competitive learning. The harmonic cost, i.e., the total cost, is the key concept of the following discussions. The terminology harmonic competition refers to competitive learning of the harmonic cost.

A. Combination of Subcosts

Let \( \{x_n\}_{n=0}^{N-1} \) be a set of input vectors to the learning network which contains a set of weight vectors \( \{w_m\}_{m=0}^{M-1} \). Let \( Q(x, w) \) be a mapping such that

\[
Q(x, w) = \begin{cases} 1, & \text{if the weight vector } w \text{ is assigned to the input vector } x, \\ 0, & \text{otherwise}. \end{cases}
\]

The average cost of the approximation by the weight vectors is then

\[
\bar{f} = \frac{1}{N} \sum_{n=0}^{N-1} f(x_n, w_m) Q(x_n, w_m) \overset{\text{def}}{=} \frac{1}{N} \sum_{n=0}^{N-1} f_n Q(x_n, w_m). \tag{2}
\]

This is the main cost. Let \( g_{nk}((x_{i_j})_{j=0}^{K-1}, (w_{j_j})_{j=0}^{M-1}), \quad (k = 0, \ldots, K - 1), \quad (i_j = 0, \ldots, L - 1) \) be constraints added to \( f_n \). Let \( h_{nt} ((x_{i_j})_{j=0}^{N-1}, (w_{j_j})_{j=0}^{M-1}), \quad (\ell = 0, \ldots, L - 1) \) be multiplicative penalties. The cost of the mapping \( Q(x_n, w_m) \)
is then

\[
D_n = \sum_{m=0}^{M-1} \left( f_n + \sum_{k=0}^{K-1} \lambda_{nk} g_{nk} \right) \left( \prod_{\ell=0}^{L-1} h_{nt} \right) Q(x_n, w_m) \overset{\text{def}}{=} N d_n. \tag{3}
\]

Here, \( x_n = x^{(t)} \) is the actual input at the \( t \)th data supply of the learning phase. That is, the superscript \( t \) is the number of iterations treated as time count. On the other hand, \( \tau = \lceil t/N \rceil \) is called the sweep. The parameters \( \lambda_{nk}, (k = 0, \ldots, K - 1) \), combine the subcosts. These parameters appear when a convex multiple criteria optimization is transformed into a cost function approach. The average of the total cost, i.e., the harmonic cost, is then

\[
\bar{d} = \frac{1}{N} \sum_{n=0}^{N-1} d_n = \frac{1}{N} \sum_{n=0}^{N-1} D_n. \tag{4}
\]

The first step of harmonic competition is to find a minimization element for \( d_n \) with respect to a successively given input data \( x^{(t)} = x_n \)

\[
w^{(t)}_{m(n)} = \arg \min_{0 \leq m < M} d_n = \arg \min_{0 \leq m < M} D_n. \tag{5}
\]

Ties are broken appropriately. The above \( w^{(t)}_{m(n)} \) is called the winner which is treated as the case where \( Q(x_n, w^{(t)}_{m(n)}) = 1 \). The neural weight vector is updated for learning by

\[
w^{(t+1)}_{m(n)} = w^{(t)}_{m(n)} + \Delta w^{(t)}_{m(n)} \tag{6}
\]

where the modification term is

\[
\Delta w^{(t)}_{m(n)} = -\frac{\varepsilon^{(t)} N}{2} \frac{\partial d_n}{\partial w^{(t)}_{m(n)}} = -\frac{\varepsilon^{(t)} N}{2} \frac{\partial D_n}{\partial w^{(t)}_{m(n)}}, \tag{7}
\]

\( \varepsilon^{(t)} \) is a learning parameter. The derivation in (7) is

\[
N \frac{\partial d_n}{\partial w^{(t)}_{m(n)}} = \frac{\partial D_n}{\partial w^{(t)}_{m(n)}} = \left( \frac{\partial f_n}{\partial w^{(t)}_{m(n)}} + \sum_{k=0}^{K-1} \lambda_{nk} \frac{\partial g_{nk}}{\partial w^{(t)}_{m(n)}} \right) \left( \prod_{\ell=0}^{L-1} h_{nt} \right) + \left( f_n + \sum_{h=0}^{L-1} \lambda_{nh} g_{nh} \right) \left( \frac{\partial}{\partial w^{(t)}_{m(n)}} \left( \prod_{\ell=0}^{L-1} h_{nt} \right) \right). \tag{8}
\]

If we consider a cooperative neighborhood, \( N^{(t)}_{m(n)} \), of the winner \( w^{(t)}_{m(n)} \), then the following update is also applied:

\[
w^{(t+1)}_{N^{(t)}_{m(n)}} = w^{(t)}_{N^{(t)}_{m(n)}} + \alpha^{(t)}_{N^{(t)}_{m(n)}} \Delta w^{(t)}_{m(n)}. \tag{9}
\]

Here, the parameter \( \alpha^{(t)}_{N^{(t)}_{m(n)}} \) specifies a degree of cooperation, possibly affected by the multiplicative handicap \( h_{nt} \) [15], [17].

The above learning strategy has the following effective interpretation: The harmonic competition by (5) and (6) indicates only the most appropriate agent with respect to the input is eligible to learn. This means that the learning is undertaken by the agent which suffices to change the total system only the least. In this sense, the harmonic competition is an embodiment of the minimum learning principle.
That is
\[ g_{m(n)} = \frac{\|x_n - w_{m(n)}\|^2}{f} \log \frac{r_{m(n)}}{q_{m(n)}}. \]

Then, the harmonic competition for the equierror is
\[ w^{(t)}_{m(n)} = \arg \min_{0 \leq m < M} \left\{ \left( 1 + \frac{\lambda}{\eta} \log \frac{r_m}{q_m} \right) \|x_n - w^{(t)}_m\|^2 \right\}. \quad (15) \]

Thus, the log-conscience acts as a multiplicative penalty. This corresponds to a shunting inhibition [13]. For the weight update, (6), (9), and (14) are used.

The equierror harmonic competition is described as follows.

**Equierror Harmonic Competition:** Replace Step 2) of the equiprobability harmonic competition as follows.

Step 2) (data feeding; increment \( t \))
A data \( x_n \) is given. The harmonic competition then selects the winner \( w^{(t)}_{m(n)} \) using (15).

In the sweep-based update of Step 5), \( p_m \) is replaced by \( r_m \).

### C. Vector Quantization Jointly with Equiprobability and Equierror

The constraint on both equiprobability and equierror is expected to show stronger effects on the exiting from bad local minima.

Let \( u_m \) be a normalized subtotal of joint probability and errors with respect to usage of the weight vector \( w_m \). That is
\[ u_m = \frac{p_m r_m}{\sum_{\ell=0}^{M-1} p_{m(\ell)}} = \frac{\eta_m}{\eta}. \]

In this case, \( \sum_m u_m = 1 \). Let
\[ \bar{g} = D(Q \| Q) = \sum_{m=0}^{M-1} u_m \log \frac{u_m}{q_m}. \]

Then, the total cost is
\[ d = f + \lambda \bar{g} \]
\[ = \frac{1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \|x_n - w_m\|^2 Q(x_n, w_m) \]
\[ + \lambda \sum_{m=0}^{M-1} u_m \log \frac{u_m}{q_m} \]
\[ = \frac{1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \left\{ \left( 1 + \frac{\lambda p_m}{\eta} \log \frac{u_m}{q_m} \right) \|x_n - w_m\|^2 \right\} Q(x_n, w_m) \]
\[ = \frac{1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \left\{ \left( 1 + \frac{\lambda p_m}{\eta} \log \frac{u_m}{q_m} \right) \|x_n - w_m\|^2 \right\} Q(x_n, w_m). \]

That is
\[ g_{m(n)} = \frac{p_m \|x_n - w_m\|^2}{\bar{g}} \log \frac{u_m}{q_m}. \]

This is again a multiplicative penalty. Thus, the harmonic competition for the joint equiprobability/error is
\[ w^{(t)}_{m(n)} = \arg \min_{0 \leq m < M} \left\{ \left( 1 + \frac{\lambda p_m}{\eta} \log \frac{u_m}{q_m} \right) \|x_n - w^{(t)}_m\|^2 \right\}. \quad (16) \]

The joint equiprobability/error competitive learning is described as follows.

---

**Fig. 1.** Generation of training data. (a) Data set A. (b) Data set B. (c)-(f) Four Gaussian clusters used for the generation of data set A.

**Joint Equiprobability/Error Harmonic Competition:** Replace Step 2) of the equiprobability harmonic competition as follows.

Step 2) (data feeding; increment \( t \)).
A data \( x_n \) is given. Harmonic competition then selects the winner \( w^{(t)}_{m(n)} \) using (16).

In the sweep-based update of Step 5), \( p_m \) is replaced by \( u_m \).

**D. Experiments of Equiprobability, Equierror, and Joint Equiprobability/Error**

In all of the following vector quantization experiments, the updating of cooperative neurons [Step 4)] was not performed. This is because our target was the minimization of the approximation error taking the exit from bad local minima into account. In the experiments of Section IV, however, updating of the cooperating neurons [Step 4)] will be identified as an important phase of learning.

1) **Two Sets of Training Data:** First, we prepare two sets of training data illustrated in Fig. 1(a) and (b). Set A is a simple mixture of four Gaussian clusters in a unit square.
be a set of desirable output probabilities; e.g., \( q_m = 1/M \) for the uniform case. The divergence is then
\[
\bar{d}(P\|Q) = \sum_{m=0}^{M-1} p_m \log \frac{p_m}{q_m}.
\]

Here, \( \log \) is the natural logarithm. The average cost (10) for equiprobability harmonic competition is then
\[
d = \bar{d} + \lambda \bar{g} = \frac{1}{N} \sum_{n=0}^{N-1} \left( \frac{1}{M} \sum_{m=0}^{M-1} \|x_n - w_{m(n)}\|^2 + \lambda \sum_{m=0}^{M-1} p_m \log \frac{p_m}{q_m} \right).
\]

That is
\[
d = \frac{1}{N} \sum_{n=0}^{N-1} \left( \frac{1}{M} \sum_{m=0}^{M-1} \|x_n - w_m\|^2 Q(x_n, w_m) \right) + \lambda \sum_{m=0}^{M-1} p_m \log \frac{p_m}{q_m}.
\]

Since the data \( x_n \) is drawn uniformly from the finite source \( \{x_n\}_{n=0}^{N-1} \), the probability \( p_m \) in the long run is as follows:
\[
p_m = \frac{1}{N} \sum_{n=0}^{N-1} Q(x_n, w_m).
\]

Then, one obtains
\[
d = \frac{1}{N} \sum_{n=0}^{N-1} \left( \frac{1}{M} \sum_{m=0}^{M-1} \|x_n - w_m\|^2 + \lambda \log \frac{p_m}{q_m} \right) Q(x_n, w_m)
\]
\[
= \frac{1}{N} \sum_{n=0}^{N-1} \left( \|x_n - w_m(n)\|^2 + \lambda \log \frac{p_m(n)}{q_m(n)} \right)
\]

That is
\[
[g]_{m(n)} = \log \frac{p_m(n)}{q_m(n)}.
\]

Thus, harmonic competition for the equiprobability is
\[
w^{(t)}_{m(n)} = \arg \min_{0 \leq m \leq M} \left\{ \|x_n - w^{(t)}_m\|^2 + \lambda \log \frac{p_m}{q_m} \right\}.
\]

Note that (12) is the log-conscience [16] whose approximation of \( \log x \approx x - 1 \) around \( x = 1 \), as well as, \( q_m = 1/M \) is the conscience [5]. The second term, \( \Delta w^{(t)}_{m(n)} \), of (11) can be omitted in successive data inputs. Thus, the increment is
\[
\Delta w^{(t)}_{m(n)} \approx e^{(t)}(x_n - w^{(t)}_{m(n)}).
\]

The learning algorithm is then described as follows.

**Equi-probability Harmonic Competition:**

1. **Step 1** (initialization; \( t = 0 \))
   - A set of training data \( \{x_n\}_{n=0}^{N-1} \) and a set of initial weight vectors \( \{w^{(0)}_m\}_{m=0}^{M-1} \) are given.
2. **Step 2** (data feeding; increment \( t \))
   - A data \( x_n \) is given. The harmonic competition then selects the winner \( w^{(t)}_{m(n)} \) using (13).
3. **Step 3** (weight update)
   - Update the winner’s weight \( w^{(t)}_{m(n)} \) using (6) and (14).
4. **Step 4** (self-organization; optional)
   - Update the cooperating neurons using (9) and (14).
5. **Step 5** (test and termination)
   - If a predefined stop condition is met (e.g., the number of iterations), then iteration is halted. Otherwise, modify \( e^{(t)} \), \( \lambda^{(t)} \), and \( \alpha^{(t)}_{\tau}(m(n)) \) according to the given rules. Then, go to Step 2.

From both computational and performance view points, Step 5 includes additional strategies.

**Sweep-Based Update; Computational Cost Reduction:** In Step 5), the quantity of the competition bias \( p_m \) in the case of equiprobability is computed only at the end of every sweep. Since the update of the parameters \( e^{(t)} \), \( \lambda^{(t)} \), and \( \alpha^{(t)}_{\tau}(m(n)) \) is related to this competition bias, the sweep count is used as the time index. That is, \( e^{(t)} \), \( \lambda^{(t)} \) and \( \alpha^{(t)}_{\tau}(m(n)) \) are kept constant for \( N \tau \leq t < (\tau + 1)N \).

**Additional Strategies for Increasing Performance:** In Step 5), additional strategies can be incorporated to increase the total performance. Dynamic split and weight vector mutation are such strategies. Explicit descriptions of these methods will be given later.

**B. Equi-error Vector Quantization**

Equi-error vector quantization is considered to be asymptotically optimal for errors in the form of difference distortion measures [8, p. 376], [24]. One should, however, use this property only when there is a considerable number of weight vectors for much more rich source data. This is because the property holds only in the limit. Since vector quantization is used for low to medium rate compression, exact equi-error should not be requested. It is expected, however, that the optimal or near-optimal solutions appear as almost equi-error. This is a clue in finding a strategy for the escape from bad local minima where the distribution of errors is uneven.

Let \( r_m \) be a normalized subtotal of errors due to usage of the weight vector \( w_m \). That is
\[
r_m = \frac{f_m}{\bar{f}}
\]
with
\[
\begin{align*}
\{f_m\} &= \frac{1}{N} \sum_{n=0}^{N-1} \|x_n - w_m\|^2 Q(x_n, w_m), \\
\bar{f} &= \sum_{m=0}^{M-1} \bar{f}_m.
\end{align*}
\]

Therefore, \( \sum_{m=0}^{M-1} r_m = 1 \) holds. Let \( R = \{r_m\}_{m=0}^{M-1} \). The subcost for the equi-error is then
\[
\bar{d} = D(R\|V) = \sum_{m=0}^{M-1} r_m \log \frac{r_m}{q_m}.
\]

The total cost is then
\[
\bar{d} = \bar{f} + \lambda \bar{g}
\]
\[
= \frac{1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \|x_n - w_m\|^2 + \lambda \sum_{m=0}^{M-1} r_m \log \frac{r_m}{q_m}
\]
\[
= \frac{1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \left[ \lambda \left( \frac{r_m}{q_m} \right) \|x_n - w_m\|^2 Q(x_n, w_m) \right]
\]
\[
= \frac{1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \left( \lambda \frac{r_m}{q_m} \right) \|x_n - w_m\|^2 Q(x_n, w_m)
\]
\[
= \frac{1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \left( \frac{r_m}{q_m} \right) \|x_n - w_m\|^2 Q(x_n, w_m)
\]

TABLE II
MINIMUM VALUES OBTAINED FOR A FIXED $\lambda$ WITH DYNAMIC SPLIT

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Set A</th>
<th>Set B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\lambda$</td>
<td>$N_f$</td>
</tr>
<tr>
<td>Equi-probability</td>
<td>0.000001</td>
<td>2.783</td>
</tr>
<tr>
<td>Equi-error</td>
<td>0.0002</td>
<td>2.772</td>
</tr>
<tr>
<td>Equip/error</td>
<td>0.0050</td>
<td>2.791</td>
</tr>
</tbody>
</table>

Fig. 3. Performance of log-equiprobability competition with constant $\lambda$. (a) Result on data set A. (b) Result on data set B.

by $D(P\|Q)$ with the parameter $\lambda$ fixed. Fig. 4(a) and (b) are the results of equi-error constraint by $D(R\|Q)$. Fig. 5(a) and (b) are the results of joint equiprobability/error constraint by $D(U\|Q)$. In all experiments, the result of the dynamic split is used as the initial weights. The vertical axes specify the approximation error $N_f$ and each $\bar{g}$. The horizontal axes specify the combination parameter $\lambda$ in the logarithmic scale. Table II summarizes the minimum values for a fixed $\lambda$. These figures and table show the followings clearly:

- All figures indicate that excessive constraint by a large $\lambda$ degrades the performance of the approximation $N_f$.
- Fig. 3 indicates that the equiprobability by minimizing $f$ and $\bar{g}$ conflict with each other. There is a trade-off relationship.

- The equi-error constraint of Fig. 4 indicates that minimizing $f$ and $\bar{g}$ is compatible. Small $f$ (actually, small $N_f$ in the figure) is achieved at a small value of $\bar{g}$.
- Fig. 5 shows that the constraint of the joint equiprobability/error is quite strong. The approximation performance $N_f$ is flat over a wide range of the combination parameter $\lambda$. This is a desirable property. The minimum value, however, is larger than that in the case of equi-error (see Table II).
- In all cases, the dynamic range of $\lambda$ is quite wide. Appropriate values of $\lambda$ are never given a priori (see Table II).
- The minimum value of the approximation error is given by the equi-error harmonic competition.

5) Dynamic Control of $\lambda$: The minimization of $f$ and $\bar{g}$ is a typical case of multiple criteria optimization. Usually, the solution is not a unique pair $(\bar{f}, \bar{g})$. Actually, every $\lambda$ possesses a corresponding point in the Pareto optimal set which is a family of extreme pairs of $\bar{f}$ and $\bar{g}$. Thus, the determination
• $A_1$: $(\mu_x, \mu_y) = (0.5, 0.5), (\sigma_x, \sigma_y) = (0.15, 0.15)$; 512 points.
• $A_2$: $(\mu_x, \mu_y) = (0.5, 0.75), (\sigma_x, \sigma_y) = (0.1, 0.005)$; 512 points.
• $A_3$: $(\mu_x, \mu_y) = (0.25, 0.25), (\sigma_x, \sigma_y) = (0.0075, 0.00075)$; 512 points.
• $A_4$: $(\mu_x, \mu_y) = (0.75, 0.25), (\sigma_x, \sigma_y) = (0.1, 0.05)$; 512 points with a rotation of $-45^\circ$.

These are shown in Fig. 1(c)–(f). The total number of training data is $N = 2048$. The number of weight vectors is $M = 32$.

Set A looks complicated, however, it is rather well natured. We prepare another set for establishing the presented learning. Set B has two square doughnut ditches centered at $(0.5,0.5)$ with a width of 0.1. The exterior corners of the ditches include $(0.1,0.1)$ and $(0.3,0.3)$. The total number of data is kept to be $N = 2048 = 512 \times 4$. In set B, there are a few isolated points. In plain successive learning, neurons trapped at such points can never move anywhere. This occurs even if the neighborhood update of Step 4) is used.

2) Control of the Learning Parameter $\varepsilon^{(t)}$: The learning parameter $\varepsilon^{(t)}$ necessarily tends to zero as $t$ becomes large. This does not mean, however, that $\varepsilon^{(t)}$ is monotonically decreasing. There are occasions where it is better for $\varepsilon^{(t)}$ to increase. We choose the dynamic control of $\varepsilon^{(t)}$ to be updated at every sweep ($N$ data supplies). That is, the sweep-based update is adopted in Step 5) of the learning algorithms. This is because of the reduction in the neuron's communication cost. Let the sweep index be $\tau = \lfloor t/N \rfloor$. The rule is as follows:

$$ \varepsilon^{(\tau+1)} = \max\{\varepsilon^{(\tau)} + \Delta \varepsilon^{(\tau)}, \varepsilon^{\text{max}}\}. $$  \hspace{1cm} (17)

Here

$$ \Delta \varepsilon^{(\tau)} = \begin{cases} 
\gamma \left( \tilde{f}^{(\tau-1)} - \tilde{f}^{(\tau)} \right), & \text{if } \tilde{f}^{(\tau)} < \tilde{f}^{(\tau-1)} \text{ and } \tilde{g}^{(\tau)} > \tilde{g}^{(\tau-1)}, \\
0, & \text{otherwise}
\end{cases} $$  \hspace{1cm} (18)

with

$$ \gamma = \begin{cases} 
0.5, & \text{if } \varepsilon^{(\tau+1)} < \varepsilon^{(\tau)}, \\
0.25, & \text{if } \varepsilon^{(\tau+1)} \geq \varepsilon^{(\tau)}
\end{cases} $$  \hspace{1cm} (19)

The control (18) allows the learning parameter $\varepsilon^{(\tau)}$ to increase, while the control (19) ensures the decreasing trend of $\varepsilon^{(\tau)}$.

3) Dynamic Split: Our initial experiment is to see how ill-natured the data sets A and B are. We start with the case where the initial weight vectors are decided by random numbers. The next experiment with an additional strategy is the aforementioned dynamic split of the weight vectors which is incorporated in Step 5). The strategy of splitting the weight vectors has been widely accepted in the batch mode where all $N$ data are exposed to the learning mechanism simultaneously. Our purpose here is to see how effective the splitting is in the successive learning.

Weight Vector Splitting for Successive Learning: First, one weight vector is placed at the centroid of the training set. After every $\kappa$ sweeps ($\kappa N$ data supplies), the positions of the weight vectors are copied. Both the original and copied weight vectors may be perturbed by the addition of small random numbers.

After $\kappa \log_2 M$ sweeps, the positions of the $M$ weight vectors are decided.

The dynamic split is regarded as a process of finding a good initial state. Thus, one may use this weight vector set as the initial one for learning. Table I summarizes the effect of the dynamic split with $\kappa = 10$ sweeps by comparing the approximation error $N \tilde{f}$. The results of the batch mode after the dynamic split are also given. For both of data sets A and B, the dynamic split is found to be quite effective although the results are still suboptimal. Observe that the batch mode is more likely to get captured at inferior local minima. Fig. 2 shows the results of successive learning starting with random weights. The result of set A is almost all right. It is still, however, at a local minimum containing a null neuron to be eliminated. With the result of set B, one easily realizes it is a terrible pattern. These patterns of the resulting weight vectors will be compared later with those of the improved strategies.

4) Effects of the Subcosts for a Fixed $\lambda$: Here, we show the effects of the subcosts $D(P\|Q)$, $D(R\|Q)$ and $D(U\|Q)$. Fig. 3(a) and (b) are the results of equiprobability constraint

| Table I | APPROXIMATION ERROR $N \tilde{f}$ FOR PLAIN STRATEGIES ($\lambda = 0$) |
|-----------------|-----------------|-----------------|
| Strategy | Set A | Set B |
| Successive learning started with random weights | 2.883 | 2.314 |
| Successive learning with dynamic split | 2.839 | 1.769 |
| Batch learning after dynamic split | 2.922 | 1.790 |

Fig. 2. Learning results starting with a random initial weight vector set: Plain strategy ($\lambda = 0$). (a) Resulting weight vectors for data set A. (b) Resulting weight vectors for data set B.
Fig. 6. Resulting weight vector sets by log-equirror with a dynamically controlled $\lambda$. (a) Result on data set A. (b) Result on data set B. (c) Specified adjustment of $\bar{\mu}$. (d) Trend of $e$. (e) Dynamic control of $\lambda$ obtained. (f) Convergence of the performance $N_f$. (g) Effective constraint $\tilde{y}$. 
of an appropriate \( \lambda \) to find a preferred solution requires some more conditions. Since our purpose is to find the minimal \( f \), a decreasing trend of \( \lambda \) is preferred. Direct control of \( \lambda \), however, may not be stable because of its wide dynamic range (see Figs. 3–5). Considering the relationship

\[
\dd{1} = \dd{1} + \lambda \dd{1} = \dd{1}(1 + \lambda \dd{1} / \dd{1}) \overset{\text{def}}{=} \dd{1}(1 + \dd{1})
\]

we adjust the parameter \( \lambda^{(r)} \) by the subcost ratio control

\[
\mu^{(r)} = \lambda^{(r)} \dd{1}^{(r)}/\dd{1}^{(r)} = (1 + a\dd{1})^{-1}. \tag{20}
\]

Note that the superscript \( r \) stands for the sweep \( (r = [t/N]) \). Thus, the controlled parameter \( \lambda \) is updated only for every sweep as was explained in Section III-A. This is because of the reduction in the communication cost. The coefficient \( a \) can easily be chosen by taking the maximum number of iterations into account.

Table III summarizes the results of the dynamic control of \( \lambda \) using the control (20). Both equiproability and equierror are tried. Fig. 6(a)–(g) describes the resulting weight vector positions and the learning process for the case of equierror using (20) \( (a = 0.14) \). In Fig. 6(c)–(g), the horizontal axes specify the number of sweeps \( r \). From Table III, the followings are noted:

- The equierror competition is usually superior to the equiprobability with respect to the approximation error \( N \dd{1} \).
- The resulting approximation errors for the equierror are close to the smallest values given in Table II. Such values can be obtained by a wide range of the coefficient \( a \).

Fig. 6 indicates the following:

- In Fig. 6(a), the regular positioning of the weight vectors can be observed according to the shape of the data clusters.
- As in Fig. 6(d), the learning parameter \( \varepsilon^{(r)} \) can go up, however, the trend is to decrease.
- As in Fig. 6(e), the wide dynamic range of \( \lambda^{(r)} \) is covered.

Instead of using the dynamic control (20), there is a perfect autonomous control of \( \lambda^{(r)} \)

\[
\mu^{(r+1)} = \mu^{(r)} + \Delta \mu^{(r)}. \tag{21}
\]

The amount \( \Delta \mu^{(r)} \) is described by observing the rise and fall of \( \dd{1}^{(r)} \), \( \dd{1}^{(r-1)} \), \( \dd{1}^{(r)} \), and \( \dd{1}^{(r-1)} \). The rule is similar to (18) and (19). The performance is similar to that of case (20). The computation is more complex, therefore, we omit the details here.

6) Log-Conscience Mutation of the Weight Vectors: So far, we have observed that a combination of strategies (dynamic initial split, log-equirror competition and dynamic control of \( \lambda \)) was the best way to guide the learning process to an almost optimal result. In this section, we consider a method of guiding the performance to a near-optimal solution in a discontinuous way. The method is log-conscience mutation. We choose the case of log-equirror for the following explanation: That is, \( \{r_m\}_{m=0}^{M-1} \) is selected. In the case of other conscience mechanisms, \( r_m \) is replaced by \( p_m \) or \( u_m \).

Log-Conscience Mutation for Equierror:

Step 1)

Let \( \{r_m\}_{m=0}^{M-1} \) be sorted. That is, \( r_i \leq r_j \) for \( i < j \), as well as \( r_0 = r_{\text{min}} \) and \( r_{M-1} = r_{\text{max}} \).

Step 2)

Compute

\[
x_m = M \left\{ \log \frac{r_m}{r_{\text{min}}} / \left( \sum_{i=0}^{M-1} \log \frac{r_i}{r_{\text{min}}} \right) \right\}. \tag{22}
\]

Note that \( \sum_{m=0}^{M-1} x_m = M \). The real number set \( \{x_m\}_{m=0}^{M-1} \) is rounded to a set of integers \( \{\delta_m\}_{m=0}^{M-1} \). Fractions are rounded to one or zero according to their magnitudes so that \( \sum_{m=0}^{M-1} \delta_m = M \) is maintained.
TABLE V

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Set A</th>
<th>Set B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log-equipr mutation (λ = 0.000)</td>
<td>2.848, 0.2479</td>
<td>1.595, 0.3388</td>
</tr>
<tr>
<td>Log-equipr mutation (λ = 0.001)</td>
<td>2.915, 0.1403</td>
<td>1.883, 0.1211</td>
</tr>
<tr>
<td>Log-equipr error mutation (λ = 0.000)</td>
<td>2.815, 0.0598</td>
<td>1.844, 0.0544</td>
</tr>
<tr>
<td>Log-equipr error mutation (λ = 0.001)</td>
<td>2.773, 0.0425</td>
<td>1.561, 0.1321</td>
</tr>
<tr>
<td>Log-equipr/pr error mutation (λ = 0.010)</td>
<td>2.805, 0.0433</td>
<td>1.598, 0.0709</td>
</tr>
<tr>
<td>Log-equipr/pr mutation (λ = 0.001)</td>
<td>2.841, 0.0634</td>
<td>1.559, 0.484</td>
</tr>
</tbody>
</table>

TABLE VI

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Set A</th>
<th>Set B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log-equipr bias and mutation</td>
<td>2.589, 0.2529</td>
<td>2.144, 0.1094</td>
</tr>
<tr>
<td>Log-equipr error and mutation</td>
<td>2.819, 0.0505</td>
<td>1.600, 0.0002</td>
</tr>
</tbody>
</table>

like Fig. 3–5 are required. Such computationally demanding occasions are rather usual. The dynamic control of d) or the weight vector mutation of e) is preferable. In that case, log-conscience should be used for the equierror. Thus, our recommendation based upon the approximation performances is listed as follows:

1) \( [b] + x \{ \text{log-equipr'} \{ c + e \} \text{ with a fixed small } \lambda \} \).
2) \( [b] + \{ \text{log-equipr'} \} d \).
3) \( [\text{log-equipr'} \{ c + e \} \text{ with a fixed small } \lambda] \).
4) \( [b] \) alone.

Note that the strategy "(b) alone" is recommended only if running the risk of local optima is allowed.

E. \( \alpha \)-divergence and log-AEP

Divergence has a more general form \([2], [10]\). It is called \( \alpha \)-divergence (see the equation shown at the bottom of the next page.) We used the case of \( \alpha = -1 \) on the normalized measures \( P \) and \( Q \) (Kullback–Leibler number). The choice of \( \alpha \) affects the degree of the log-conscience and mutation. That is, this number can change the trend of the logarithmic asymptotic equi-partition property (AEP) in the sense of the quantization for source coding. The case of \( \alpha = -1 \) gave our log-AEP in probability, error, and joint probability/error. There exists a general trend that the effect becomes weaker as \( \alpha \) increases \([18]\). The optimal value of \( \alpha \), however, depends on both the nature of the source data and the number of neurons. Thus, the optimal choices for the data set A and B are different. Such a fine tuning is not required so far.

IV. REGULARIZATION WITH CONSTRAINTS

Here, we discuss a class of problems with the number, \( M \), of neurons which is larger than that of the data, \( N \). Since the set of data is fed into a larger population of neurons, additional constraints are necessary. This is called
TABLE IV
EFFECT OF A LOG-CONSCIENCE MUTATION AFTER DYNAMIC SPLIT

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Set A</th>
<th>Set B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\lambda$</td>
<td>$N_f$</td>
</tr>
<tr>
<td>Log-equiprob. mutation</td>
<td>0.000005</td>
<td>2.784</td>
</tr>
<tr>
<td>Log-equierror mutation</td>
<td>0.01</td>
<td>2.774</td>
</tr>
<tr>
<td>Log-equip. /error mutation</td>
<td>0.09</td>
<td>2.778</td>
</tr>
</tbody>
</table>

1) log-equiprob. mutation with $\lambda = 0.000005$ is a result of intensive search (see Fig. 7).

Step 3)
There are three cases

$$
\begin{align*}
    s_m & \geq 2 & w_m \text{ is copied } (s_m - 1) \text{ times onto} \\
    & \text{other } w_m \text{'s for which } s_n = 0, \\
    s_m & = 1 & \text{no operation on } w_m, \\
    s_m & = 0 & w_m \text{ is changed to one} \\
    & \text{of other } w_m \text{'s for which } s_n \geq 2.
\end{align*}
$$

The log-conscience mutation of (23) maintains $\sum_{s_m=1}^{M-1} s_m = M$. In the weight copy of Step 3, some small noise may be added as was seen in the dynamic split.

For the cases of equiprobability and joint equiprobability/error, the logarithmic quantity $\log(r_m/r_{\min})$ is replaced by $\log(p_m/p_{\min})$ and $\log(u_m/u_{\min})$, respectively.

It is important to point out that the above mutation strategy is based on the log-consciences (13), (15), and (16), respectively. The log-conscience terms result from the original harmonic competition of the cost (10). Thus, the log-conscience always exists in each competition. The mutation is interpreted as occasional strong applications of the log-conscience. In [22], a similar mechanism to our log-conscience mutation is used. Their method, however, is simply based on $r^b_m$. The exponent $b < 1$ is an experimentally determined ad hoc number. Thus, there is no theoretical foundation for the method of equierror learning. Besides, there is no competition bias at all. Taking these into account, their case corresponds to $\lambda^0 = 0$ and $r^b_m$, ($b < 1$), for the approximation of our $\log(r_m/r_{\min})$. Since $\lambda^0 = 0$ and the mutation should be ceased before convergence, their method stays within the bounds of the selection of better initial weight vectors.

Fig. 7(a) and (b) shows the results of log-equiprobability mutation with dynamic split. Fig. 8(a) and (b) shows the results of log-equierror mutation with dynamic split. Table IV compares the minimum values of the approximation errors $N_f$ for the three strategies; log-equiprobability, log-equierror, and joint log-equiprobability/error with mutations. From these figures and table, one finds the following:

- Fig. 7(a) and (b) indicate that log-equiprobability mutation is again incompatible with the approximation performance.
- From Fig. 8(a) and (b), one observes that the approximation error $N_f$ is flat over a wide range of the combination parameter $\lambda$. Its level is good enough (compare with Fig. 4). This is quite a desirable property.
- From Table IV together with Fig. 7(a) and (b), and Fig. 8(a) and (b), the log-equierror mutation after dynamic split is found to be the best. Comparing the approximation errors of Table IV with those of Tables II and III, one finds most of the minimum values here. Equalization of the error by logarithmic order was proved to be effective.

Next, we measure basic ability of the log-equierror mutation without the dynamic split of initial weights. Table V is the resulting performance starting with the same random initial weight set used in the case of Fig. 2. The performance of the log-equieor correlation is again around the minimum obtained. It is slightly inferior to the values given in Table IV, however, where the initial dynamic split exists. Fig. 9 (a) and (b) are the resulting positions of the weight vectors for log-equieor started with the random set. Comparing Fig. 9 with Fig. 2, we can easily observe quite an improvement. Null neurons never appear. The similarity of Fig. 6 and Fig. 9 can also be observed.

In the last experiment, we measure the case of the full strategy: dynamic split, dynamic control of $\lambda$ and log-equieor mutation. Table VI shows the performance. Comparing Table VI with Tables II–V, we can judge that the full strategy (dynamic split, dynamic control of $\lambda$, and log-equieor mutation) is not necessary. Thus, too strict an equieor property is harmful.

7) Recommended Strategies: We have presented the following five strategies:

a) Dynamic control of the learning parameter $\varepsilon^{(r)}$.

b) Dynamic split of the initial weight vectors.
with
\[ \mu^{(r)} = \alpha \tau. \]

This is again a dual case of (20). The second adjustment rule is a weaker version on the rate of the growth
\[ \mu^{(r)} = \alpha \tau / \epsilon^{(r)}. \]

The third rule for adjusting \( \lambda^{(r)} \) is the following:
\[ \mu^{(r+1)} = \mu^{(r)} + \Delta \mu^{(r)} \]
with (33) shown at the bottom of the page. Here
\[ \tilde{\eta} = \tilde{g}^{(r)} / \tilde{f}^{(r)}. \]

This is an autonomous control rule. In later experiments, the above three adjustment rules on \( \lambda^{(r)} \) will be examined.

The harmonic competition for TSP is summarized as follows.

**Harmonic Competition for TSP with Dynamic Adjustment:**

**Step 1** (initialization; \( t = 0, \tau = [t/N] = 0 \))

The following data set and initial values are given:
- Set of cities \( X = \{x_{n}\}_{n=0}^{N-1} \).
- Set of neural weight vectors \( W = \{w_{m}\}_{m=0}^{M-1} \). Here, each vector \( w_{m} \) is placed on a closed curve.
- A dynamic rule for \( \epsilon^{(r)} \) of (26).
- A dynamic rule for \( \alpha^{(r)} = \alpha^{(r)} \lambda^{(r)} \) by one of the (30)–(32).
- Neighborhood weight \( f(m, \tau)I(\|m - n\| < h^{(r)}) \). Here, \( I(B) \) is an \{0, 1\}-indicator function for the event \( B; h^{(r)} \) is a decreasing function of sweep \( \tau \). Note that \( f(0, \tau) \equiv 1 \) is the maximum.
- Catch rate vigilance \( r_{0} \).

**Step 2** (feed city and increment time \( t \))

A city \( x \) is selected at random from \( X \). Find a winner \( w_{m}^{(t)} \) satisfying
\[ \min_{0 \leq m < M} \|x - w_{m}^{(t)}\|^2. \]

**Step 3** (weight update and self-organization)

For \( w_{m}^{(t)} \) with \( |m - \ell| \leq h^{(r)} \), compute the following:
\[ w_{m}^{(t+1)} = w_{m}^{(t)} + \epsilon^{(r)} f(m - \ell, \tau)I(\|m - n\| < h^{(r)}) (x - w_{m}^{(t)}) \]
\[ + \alpha^{(r)} (w_{m+1}^{(t)} - 2w_{m}^{(t)} + w_{m-1}^{(t)}). \]

Here, \( m = \ell \) is the case of the winner update. The rest of the cases \( m \neq \ell \) are for the self-organization.

**Step 4** (test and termination)

The following test and update are executed at every sweep. If each city has a distinct winner, then the learning is completed. Otherwise, if the catch percentage is greater than \( r_{0} \) and not increasing, there is a winner for multiple cities. This weight is copied. Then, \( \epsilon^{(r)}, \alpha^{(r)}, f(m, \tau) \) are updated according to (26) and one of (30)–(32). Then, go to Step 2.

2) Experiments: The USA 532 set [20] is used for testing the ability of the harmonic competition with dynamic control of the parameters. This data set contains \( N = 532 \) cities in the USA which are located quite nonuniformly. The initial neural weights are placed on an ellipse of the USA territory. The total number of neurons is \( M = 2500 \). Common specifications throughout the experiments are: \( \epsilon^{(0)} = 0.25; f(k, \tau) = \exp(-k^2/2\sigma^2(\tau)); \sigma(\tau) = \sigma(0)(1 - s^n); \) and \( h^{(r)} = 2\sigma(\tau). \)

Fig. 10(a), (b), and (c) show the process of learning and the resulting tour. The adjustment rule for \( \lambda^{(r)} \) uses (31) with \( a = 2N^2 \times 10^{-7} \). Fig. 10(a) shows the initial configuration of neurons (\( \tau = 0 \)). Fig. 10(b) illustrates the intermediate state of learning (\( \tau = 85 \)). Fig. 10(c) is the resulting tour with a length of 8.9731 at \( \tau = 130 \). Fig. 10(d) illustrates the progress of learning by showing the parameter \( \epsilon^{(r)} \). One observes from this figure that the learning parameter \( \epsilon^{(r)} \) has an increasing trend. This is the dual case of Chapter III where \( \epsilon^{(r)} \) had a decreasing trend [Fig. 6(d)].

Table VII compares the performances of the previously obtained results of the static control [17] and the newly obtained ones by dynamic control. Note that there are three dynamic rules on the adjustment of \( \lambda^{(r)}; (30)–(32). \) Comparing the performance and the required computation, the dynamic rule of (31) is recommended. We note here that the method of Angéniol et al. [11] is omitted. This is because that method corresponds to the case of \( \lambda^{(r)} \equiv 0 \) which gives inferior results.

**B. Extended Vehicle Routing Problems**

The vehicle routing problem [4] arose from the multiple person TSP. In the vehicle routing problem, however, drivers visit cities and collect or deliver their items (demands). There is a maximum load for the acceptable amount for each vehicle. Thus, the target of optimization is both the route length and vehicle load.

The extended vehicle routing problems (EVRP’s) tried here have more constraints to be satisfied: Each city has its own preference for specific vehicles. Rejected vehicles cannot visit the cities which refuse them. This property, together with the optimization of the tour and demands, is a good example showing multiplicative constraints.

1) Formulation and Algorithm: There are \( K \) vehicles which start from, and come back to, the same depot. The \( k \)th vehicle’s subtour length is
\[ D_k = \sum_{i_k = 0 \mod N_k}^{N_k - 1} d(x_{\pi(i_k)}, x_{\pi(i_k + 1)}), \quad (k = 0, \cdots, K - 1). \]

\[ \Delta \mu^{(r)} = \begin{cases} \mu_{\text{down}}^{(r)}(\tilde{\eta}^{(r)} - \tilde{\eta}^{(r-1)}) / \tilde{\eta}^{(r-1)}, & \text{if } \tilde{f}^{(r)} \leq \tilde{f}^{(r-1)} \text{ and } \tilde{g}^{(r)} \leq \tilde{g}^{(r-1)}, \\ \mu_{\text{up}}^{(r)}(\tilde{\eta}^{(r)} - \tilde{\eta}^{(r-1)}) / \tilde{\eta}^{(r-1)}, & \text{if } \tilde{f}^{(r)} \leq \tilde{f}^{(r-1)} \text{ and } \tilde{g}^{(r)} > \tilde{g}^{(r-1)}, \\ \mu_{\text{up}}^{(r)}(\tilde{\eta}^{(r)} - \tilde{\eta}^{(r-1)}) / \tilde{\eta}^{(r-1)}, & \text{if } \tilde{f}^{(r)} \geq \tilde{f}^{(r-1)} \text{ and } \tilde{g}^{(r)} \leq \tilde{g}^{(r-1)}, \\ \mu_{\text{up}}^{(r)}(\tilde{\eta}^{(r)} - \tilde{\eta}^{(r-1)}) / \tilde{\eta}^{(r-1)}, & \text{if } \tilde{f}^{(r)} \geq \tilde{f}^{(r-1)} \text{ and } \tilde{g}^{(r)} > \tilde{g}^{(r-1)}. \end{cases} \]
regularization which prevents ill-conditioning. Among such types, Euclidian traveling salesperson problems (Euclidian TSP) and extended vehicle routing problems (extended VRP) are addressed here. The traveling salesperson problem via harmonic competition is a typical case of $M > N$. The extended vehicle routing problems are much more complicated than TSP. They are good examples of the multiple criteria case.

A. Euclidian Traveling Salesperson Problem

1) Formulation and Algorithm: The traveling salesperson problem is a well-known NP-complete problem [7]. If the set of cities visited by a salesperson is located in a Euclidian space, the problem is described as follows.

**Euclidian TSP**: Given a positive number, $T$, and a set of cities, $X = \{ x_0, \ldots, x_{M-1}\}$ specified by position vectors, $x_m \in R^L(\forall m)$, determine if there exists an ordering $(x_0, \ldots, x_{M-1})$ which satisfies

$$
\sum_{m=0}^{M-1} d(x_\pi(m), x_\pi(m+1)) \leq T. \tag{24}
$$

Here, $d$ is the Euclidian metric and $\pi$ is a permutation.

This problem is equivalent to finding the minimum value of the total tour length with respect to the permutation $\pi$. We are interested in finding good approximations to the minimal length tour using competitive learning. Two-dimensional cases ($L = 2$) are of special interest.

Let

$$
\bar{f} = \frac{1}{N} \sum_{n=0}^{N-1} ||x_n - w_m||^2
$$

and

$$
\bar{g} = \sum_{m=0}^{m(n)+M-1} ||w_m - w_{m+1}||^2 = \sum_{j=m(n)+M-1}^{M-1} ||w_j - w_{j+1}||^2 = [g]_{m(n)}.
$$

The use of the square norm is for simplicity of the learning equations. The difference from the case of the Euclidian metric is absorbed by the change in learning parameters. The total cost is

$$
\bar{d} = \frac{1}{N} \sum_{n=0}^{N-1} D_n = \bar{f} + \lambda \bar{g}
$$

with

$$
D_n = \sum_{m=0}^{M-1} ||x_n - w_m||^2 + \lambda \sum_{j=0}^{M-1} ||w_j - w_{j+1}||^2 \right\} Q(x_n, w_m).
$$

Therefore, the harmonic competition is

$$
t_{m(n)}^{(t)} = \arg\min_{0 \leq m < M} D_n = \arg\min_{0 \leq m < M} ||x_n - w_m^{(t)}||^2.
$$

The update term is obtained from (7) as follows:

$$
\Delta t_{m(n)}^{(t)} = \varepsilon^{(t)} (x_n - w_{m(n)}) + \alpha^{(t)} \left( w_{m(n)-1} - 2w_{m(n)} + w_{m(n)+1} \right) \tag{25}
$$

where $\alpha^{(t)} \equiv \varepsilon^{(t)} \lambda^{(t)}$. An important point of this problem is how to adjust the parameters $\varepsilon^{(t)}$ and $\lambda^{(t)}$, or equivalently, $\varepsilon^{(t)}$ and $\alpha^{(t)}$. In the previous study [17], the temporal change in these numbers was predefined. In this paper, dynamic control of these parameters is attempted.

The rule of control of the learning parameter is as follows: The update is computed at every sweep ($N$ data supplies). The superscript $\tau$ is used instead of $t$. That is, $\tau = \lfloor t/N \rfloor$. The learning parameter $\varepsilon^{(\tau)}$ is adjusted by the following:

$$
\varepsilon^{(\tau)} = \max\{ \varepsilon^{(\tau)} + \Delta \varepsilon^{(\tau)}, \varepsilon^{\text{max}} \} \tag{26}
$$

with

$$
\Delta \varepsilon^{(\tau)} =
\begin{cases}
0, & \text{if } \bar{f}^{(\tau)} > \bar{f}^{(\tau-1)} \text{ and } \bar{g}^{(\tau)} < \bar{g}^{(\tau-1)}, \\
\gamma \frac{\bar{f}^{(\tau-1)} - \bar{f}^{(\tau)}}{\bar{f}^{(\tau-1)}}, & \text{otherwise}.
\end{cases}
\tag{27}
$$

The increment $\gamma$ is

$$
\gamma = \begin{cases}
0.5, & \text{if } \varepsilon^{(\tau+1)} < \varepsilon^{(\tau)}, \\
0.25, & \text{if } \varepsilon^{(\tau+1)} \geq \varepsilon^{(\tau)}.
\end{cases}
\tag{28}
$$

Comparing the (26)–(28) with (17)–(19), one finds a duality. Equations (26)–(28) ensure the increasing trend of the learning parameter $\varepsilon^{(\tau)}$. On the other hand, (17)–(19) show the decreasing trend of $\varepsilon^{(\tau)}$.

The first adjustment rule for $\lambda^{(\tau)}$ is based upon the following subcost ratio:

$$
\mu^{(\tau)} = \lambda^{(\tau)} \bar{g}^{(\tau)} / \bar{f}^{(\tau)} \tag{29}
$$

$$
D^{(\alpha)}(P||Q) =
\begin{cases}
\sum_{m=0}^{M-1} \left( q_m - p_m + p_m \log \frac{p_m}{q_m} \right), & (\alpha = -1) \\
\frac{1}{1 - \alpha^2} \sum_{m=0}^{M-1} \left\{ \frac{1 - \alpha}{2} q_m - p_m + \frac{1 + \alpha}{2} q_m - p^{(1-\alpha)/2} q^{(1+\alpha)/2} \right\}, & (-1 < \alpha < 1) \\
D^{(-1)}(Q||P), & (\alpha = 1).
\end{cases}
$$
Step 5) (halt iteration and parameter update)

If there is a distinct winner for each city, the learning is completed. Else if the catch percentage is greater than \( r_0 \) and not increasing with the iterations, then there is a multiple winning neuron. The weight of such a neuron is copied. Parameters and functions such as \( \varepsilon^{(r)} \) and \( \alpha^{(r)} \) are then updated.

2) Experiments: In the following experiments, problems with four vehicles, \( K = 4 \), are selected. The data set is the USA-532 set. The depot \( x_0 \) is \((0.6745,0.6781)\). Each city’s demand and city type are generated by the following method: If each city position is \((x_1,x_2,x_3,x_4,0,y_1,y_2,y_3,y_4)\) in decimal floating point numbers, the demand is set to be \( x_3 + y_3(\text{mod} \ 10) \) and the type number is \( x_4 + y_4(\text{mod} \ 4) \). Type 0 cities (marked by “□” in the illustrations) accept only vehicle \( k = 0 \). Type 1 cities (marked by “△” in the illustrations) accept vehicles \( k = 0 \) and 1. Types 2 and 3 cities (marked by “□□” in the illustrations) are treated as the same class: They accept any type of vehicles \( (k = 0,1,2,3) \).

Fig. 11(a) shows the initial state of neural weight vectors on four overlapping circles. Each circle contains \( N = 532 \) neurons. Thus, the total number of neurons is \( M = 4N = 2128 \) at the start. The city described by a filled square is the depot. Fig. 11(b) is the progress of the self-organization \( (\tau = \lceil t/N \rceil = 1) \). Fig. 11(c)–(f) are the results of EVRP1 to EVRP4. In these experiments, \( a = N^2 \times 10^{-7} \). Other parameters are the same as those specified in Section IV.

Table VIII compares the results of the presented dynamic method and the previous static case [17]. Underlined are numbers to be compared. For the EVRP1, only the total length is tested. In the case of the EVRP2, the total length and the maximum subtour length are evaluated. For the EVRP3, the total length and the maximum demand are checked. In the EVRP4, the total length and both maxima of the subtour length and the demand are compared. By this Table VIII, one finds that the results of the EVRP1, EVRP3 and EVRP4 are superior to those of the static method. Especially, the EVRP4 which is the most difficult problem with optimization conflicts, the obtained answer is more desirable by far: It has a much shorter total length. The maximum subtour length is also shorter. The maximum demand is smaller too. Thus, we can conclude that the dynamic control presented in this paper is very effective.

V. CONCLUDING REMARKS

General competitive learning was discussed. The cost to be minimized included the main cost for data approximation and the subcosts for constraints reflecting the problem to be solved. The harmonic competition is a learning strategy to minimize such composite costs for multiple criteria optimization.

In this paper, we presented two classes of problems: multiple criteria vector quantization and traveling salesman problems with their sophisticated versions. All of these are based on competitive learning with subcosts. Such a subcost approach can also be effective in problem solving with other learning
TABLE VII
TOUR LENGTHS OBTAINED FOR THE USA-532 SET

<table>
<thead>
<tr>
<th>Method</th>
<th>Tour length</th>
<th>Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Tour lengths of static control on ( \lambda ), simulated annealing [21], and elastic net [6] were obtained from the study [17].</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2) The tour length of “branch and cut” was given in [20]. This solution required the linear programming package XMP on a Cyber 205 supercomputer.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3) SS1 is a conventional workstation with a speed of 12.5 MIPS and 1.4 MFLOPS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4) Simulated annealing [21] required many repeated trials to find the best cooling schedule.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that \( x_0 \) is the depot. Every city is visited by a single vehicle. The total number of cities is \( \sum_{k=0}^{K-1} N_k = N \). The total demand of the \( kth \) vehicle is

\[
B_k = \sum_{i=0}^{N_k-1} b(x_{\pi(i_k)}), \quad (k = 0, \ldots, K - 1).
\]

The grand total demand is \( B = \sum_{k=0}^{K-1} B_k \). Thus, the basic vehicle routing problem is an optimization of \( (D_0, \ldots, D_{K-1}; B_0, \ldots, B_{K-1}) \) given a set of city positions and demands, \( (x_n, b(x_n)), (n = 0, \ldots, N - 1) \). In the extended vehicle routing problems, each city has a type for specifying acceptable vehicles, \( \{q_n\}_{n=0}^{N-1} \). Here, \( q_n \) is an element of the power set of \( K \) vehicles excluding the null set. For instance, the city \( x_2 \) accepts only even-numbered vehicles \( (k = 0, 2, \ldots, [K/2]) \). Thus, the extended vehicle routing problem is a multiple criteria optimization problem of \( (x_n, b(x_n), q_n), (n = 0, \ldots, N - 1) \). We will discuss the following four types of problems:

1. **EVRP1**: Minimize \( \sum_{k=0}^{K-1} D_k \) as long as the city preference and the upper bound \( B_k \leq B^{*} \) is met.
2. **EVRP2**: Try EVRP1 with the further restriction that the \( \text{max}_k D_k \) is kept small.
3. **EVRP3**: Try EVRP1 with the further restriction that the \( \text{max}_k B_k \) is kept small.
4. **EVRP4**: Try EVRP1 with the further restriction that the \( \text{max}_k D_k \) and \( \text{max}_k B_k \) are jointly kept small.

The constraints imposed on the EVRP’s are realized by the following multiplicative handicaps for harmonic competition

\[
h_1(x_n, 2m, q_n) = \begin{cases} 1, & \text{if the city } x \text{ accepts the vehicle } k, \\ \infty, & \text{otherwise}. \end{cases}
\]

The infinite penalty is equivalent to “using other vehicles only.” To suppress the maximum of the subtour lengths

\[
h_2(x_n, 2m, q_n) = \frac{D_k}{D} \times \frac{D_k}{D - D_k}, \quad (k = 0, \ldots, K - 1)
\]

is further multiplied for competition cost. Here, \( D = \sum_{k=0}^{K-1} D_k \). To enhance this penalty, the maximum of the subtour lengths \( D_k \), say \( D_{\text{max}} \), is replaced by \( D_{\text{max}}(KD_{\text{max}}/D) \). This is valid for \( 2D_{\text{max}} < D \). In all of the experiments, this enhancement was used. For the minimization of the maximum of subtotals demands, the above handicap (36) with the substitutions of \( D_k \) by \( B_k \) and \( D \) by \( B \) are used. This handicap is denoted by \( h_3 \). Thus, for EVRP1 to EVRP4, the multiplicative handicaps \( h_1, h_1 h_2, h_1 h_3, \) and \( h_1 h_2 h_3 \) are used, respectively.

The dynamic control is the same as (31). The algorithm is as follows:

**Algorithm for EVRP’s:**

Step 1) (initialization; \( t = 0, \tau = [t/N] = 0 \))

The following data set and initial values are given:

- Set of cities \( \{x_0, \ldots, x_{N-1}\}, x_n \in R^2 \).
- City \( x_0 \) is the depot.
- \( K \) vehicles with specified capacities.
- Each city has a fixed amount of demand \( \{b_0, \ldots, b_{N-1}\} \).
- Each city specifies acceptable vehicles by \( \{q_0, \ldots, q_{N-1}\} \).
- Neural weight vectors are located on \( K \) closed curves. \( K \) overlapping circles are allowed if tie-breakers of competition are incorporated. The \( K \) neural rings are as follows:

\[
W_k^{(t)} = \{w_{0,k}^{(t)}, \ldots, w_{m_k,M_k}^{(t)}, \ldots, w_{M_k-1,k}^{(t)}\}, \quad w_{m_k}^{(t)} \in R^2,
\]

\( (k = 0, \ldots, K - 1) \).

1. Dynamic control rule of \( \epsilon^{(t)} \) and \( \lambda^{(t)} \).
2. Neuron’s cooperation weight \( f(m, \tau)I(|m| \leq h^{(t)}) \).
3. Initial handicap of each neuron is unity.
4. Catch rate vigilance \( \tau_0 \).

Step 2) (feed city)

A city \( x \) is selected at random from the city set. If \( x \) is a regular city, go to Step 3). If it is the depot, go to Step 4).

Step 3) (update weights for a regular city).

Find a winner \( w_{\ell,k}^{(t)} \) such that

\[
w_{\ell,k}^{(t)} = \min_{0 \leq k < K - 1} \min_{0 \leq m < M_k - 1} \frac{\text{[handicap]} ||x - w_{m,k}^{(t)}||^2 \text{[handicap]} ||x - w_{m,k}^{(t)}||^2}.
\]

For \( w_{\ell,k}^{(t)} \) itself and \( w_{m,k}^{(t)} \) with \( |m| \leq h^{(t)} \), the weight vectors are updated as follows:

\[
w_{m,k}^{(t+1)} = w_{m,k}^{(t)}
\]

\[
+ \epsilon^{(t)} f(m - \ell, \tau)I(|m - \ell| \leq h^{(t)})(x - w_{m,k}^{(t)})
\]

\[
+ \alpha^{(t)} w_{m,k}^{(t+1)} - 2w_{m,k}^{(t)} + w_{m,k}^{(t-1)}.
\]

Here, \( \alpha^{(t)} = \epsilon^{(t)} \lambda^{(t)} \). Then, go to Step 5).

Step 4)

For the depot \( x = x_0 \), find \( K \) winners \( w_{m,k}^{(t)} \) satisfying

\[
\min_{0 \leq k < K - 1} \frac{||x - w_{m,k}^{(t)}||^2}{\text{[handicap]} ||x - w_{m,k}^{(t)}||^2} \quad (k = 0, \ldots, K - 1).
\]

The update of the weights are the same as in (37). Then, go to Step 5).
TABLE VIII
COMPARISON OF THE DYNAMIC AND STATIC CONTROLS
OF λ FOR THE EXTENDED VEHICLE ROUTING PROBLEMS

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Dynamic control</th>
<th>Static control</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total length</td>
<td>Sublength</td>
</tr>
<tr>
<td>EVRF1</td>
<td>10.6425</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td>4.7170</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>3.5500</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>3.5600</td>
<td>331</td>
</tr>
<tr>
<td>EVRF2</td>
<td>11.9626</td>
<td>737</td>
</tr>
<tr>
<td></td>
<td>2.0737</td>
<td>170</td>
</tr>
<tr>
<td></td>
<td>2.3741</td>
<td>253</td>
</tr>
<tr>
<td></td>
<td>3.5442</td>
<td>370</td>
</tr>
<tr>
<td></td>
<td>3.8550</td>
<td>462</td>
</tr>
<tr>
<td>EVRF3</td>
<td>3.2965</td>
<td>279</td>
</tr>
<tr>
<td></td>
<td>3.6854</td>
<td>564</td>
</tr>
<tr>
<td></td>
<td>2.7124</td>
<td>114</td>
</tr>
<tr>
<td></td>
<td>4.6462</td>
<td>419</td>
</tr>
<tr>
<td></td>
<td>3.1641</td>
<td>274</td>
</tr>
<tr>
<td></td>
<td>3.5843</td>
<td>378</td>
</tr>
<tr>
<td></td>
<td>3.6028</td>
<td>234</td>
</tr>
</tbody>
</table>

is kept minimal. Even if the problem is described as a maximization, the neuron of “arg max” instead of “arg min” is the minimal learning element. Suppose that nonminimal learning neurons were selected to learn. Such a learning strategy will not properly converge. Thus, the minimal learning is stable and effective by finding the most appropriate element to be modified. This interpretation remains valid for most of supervised learning: Only the minimum necessary modifications are applied to achieve the learning.

All of the aforementioned learning strategies were described based upon the Euclidian metric. The presented methods, however, remain valid for a class of non-Euclidian distortion measures [14]. Especially, the log-AEP can be derived for various distortion measures.

ACKNOWLEDGMENT

The author is grateful to A. Matsuno, T-S. Moon, K. Morisawa, M. Kobayashi, H. Nakayama, T. Sasaki, Y-P. Chen, and T. Furuya for their discussions and software supports.

REFERENCES


Yasuo Matsuyama (S’77–M’78–SM’92) received the B. Eng., M. Eng. and Dr. Eng. degrees in 1969, 1971, and 1974, all from Waseda University, Japan. In addition, he received the Ph.D. degree from Stanford University, California, in 1978. He worked at Bunka University as a Professor and Division Chairperson of the doctor course. In 1994, he was with the National Personnel Authority as a Cochairperson of the governmental personnel selection. Since 1996, he has been a Professor at Waseda University. His current research interests include communication and computational mechanisms, symbol/subsymbol learning algorithms, multimedia processing, and their electronic implementations.

Dr. Matsuyama received awards from the Institute of Electronics, Information, and Communication Engineers (IEICE) and the Electrical Communication Foundation. He has been a Councilor of the IEICE Tokyo Chapter since 1995.
Fig. 11. Extended vehicle routing problems. (a) Initial configuration of neurons (four overlapping circles). (b) Progress of self-organization ($\tau = 1$). (c) Resulting tours of vehicles for EVRP1. (d) Resulting tours of vehicles for EVRP2. (e) Resulting tours of vehicles for EVRP3. (f) Resulting tours of vehicles for EVRP4.

paradigms. Such an extended study including the supervised learning is given in [18].

All algorithms are expected to minimize the total cost (4) every time a data $x_n$ is fed. Finding a winner to learn was interpreted as minimal learning. This concept is not because of the minimization (5). Rather, such a notion came from the interpretation that the learning should be performed by the most appropriate neuron so that the system modification...