Fast Competition Approach using Self Organizing Map for Lip-Reading Applications

Sagheer, Alaa
Department of Intelligent Systems, Kyushu University

Tsuruta, Naoyuki
Department of Electronics Engineering and Computer Science, Fukuoka University

Taniguchi, Rin-ichiro
Department of Intelligent Systems, Kyushu University

Arita, Daisaku
Department of Intelligent Systems, Kyushu University

他

2006 International Joint Conference on Neural Networks
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Fast Competition Approach using Self Organizing Map for Lip-Reading Applications

Alaa Sagheer, *Member, IEEE,* Nayouki Tsuruta, Rin-Ichiro Taniguchi,
Daisaku Arita and Sakashi Maeda

***Abstract*—The Self Organizing Map, or Kohonen SOM, is one of the most widely used neural network paradigm based on unsupervised competitive learning. However, the search algorithm introduced by Kohonen is slow when the size of the map is large. This slowness is caused by seeking about the best matching neuron among "all" the map neurons which tuned to "each" input sample. In this paper, we present a new strategy capable to accelerate the SOM's competition algorithm. Instead of Kohonen SOM strategy, the new approach concerns "only" with the neurons which are aligned along the low-order principal components of the feature space and neglects the rest of neurons. The idea is based on the fact that most of the data variance lie on the low-order principal components of the manifold which often contain the most important features of the data [1] [6]. The new SOM can works effectively as a feature extractor for all kinds of manifolds even in the curved ones. Two data sets are utilized to illustrate how the proposed algorithm reduces the computation efforts (or time) of SOM effectively. For N-dimensions feature space, it is shown here that the computation effort to get the best matching units is reduced to $O(D_1+ D_2+\ldots+ D_N)$ instead of $O(D_1 \times D_2 \times \ldots \times D_N)$, where D_i is the number of neurons through the dimension i. Also, under same experimental conditions, our method computation time is less than that of fast DCT by sixth times. In all cases, the new SOM shows, at least, same recognition accuracy or may be better.**

I. INTRODUCTION

In many situations in pattern recognition, machine intelligence, and computer vision, it is necessary to achieve fast feature extraction for large-size multivariate data sets using a low cost tool in order to handle the information contained in these data easily. Usually, features are extracted as functions (linear or non-linear) of the original set of features. Unsupervised linear feature extraction techniques more or less all rely on *Principal Component Analysis* (PCA) [1]. It yields a linear mapping (or representation) with the most minimum amount of required cost and information loss.

However, in some situations, there is a possibility that the feature space is curved. Since PCA summarizes the data by the mean and the standard deviation (the covariance matrix), the linear representation is proper only if the data distribution is Gaussian. In other words, PCA is inappropriate tool for modeling nonlinear effects such as data bending or shape rotation [2].

Several approaches have been developed also to be as a non-linear extension for PCA, for instance, *Kernel PCA* (KPCA) [3]. It executes the linear PCA algorithm for the image of input data mapped by a nonlinear mapping function to construct the ordered principal components (*PCs*). However, KPCA does not give an adequate way to determine the exploited nonlinear mapping function. In addition, much pre-experiment computations are needed such as solve an Eigen-equation for the covariance matrix of the input image and also calculations of kernel functions between the objective data and all training data to calculate a principal component score [4].

The conventional *self-organizing map* (SOM) [5] can be also viewed as a non-linear extension of PCA. It replaces the linear subspace of PCA by a *nonlinear manifold* that can represent even the curved data distributions. The manifold is constructed by an iterative learning procedure and can be viewed as a non-linear 'topology-preserving map' of the original data space. Mainly, it models the self-organization of topographic maps in populations of discrete neurons in the brain.

Though SOM is simple enough and does not require much pre-computation; like KPCA, its discrete nature can be a limitation when the construction of smooth, higher-dimensional map manifolds is desired by the more powerful learning algorithms such as face or robot applications. It is worthy to say that, increasing the feature space dimensions will increase the computation effort of conventional SOM. In mathematical terms, in case of N-dimensions feature space, the required computation effort is $O(D_1 \times D_2 \times \ldots \times D_N)$, where D_i is the number of neurons through dimension i.

Therefore, a powerful competition algorithm preserves the properties of conventional SOM and, in the same time, avoids much computational effort or time is desired. This paper presents a new paradigm for Fast SOM (FSOM) consumes less recognition time than conventional SOM. Similar to SOM, FSOM is a non-parametric simple method does not need any pre-calculation steps, and consists of a piecewise one dimension SOM networks. The idea is based on the fact that the greatest variance of the data distribution comes to lie on the low-order axes; or principal components PCs. It is demonstrated that such low-order components often contain the most important aspects of the data set [1] [6]. Thus, our approach starts by extracting the first principal component *PC* and then picks up the first winner neuron among this component's neurons. Then extracts the second *PC* and picks up the second winner neuron and so

A. Sagheer is with department of Intelligent Systems, Kyushu University, Japan (corresponding author to provide phone: +81-80-5267-4724; fax: +81-92-583-1338; e-mail: alaa@limu.is.kyushu-u.ac.jp).

N. Tsuruta and S. Maeda are with Department of Electronics Engineering and Computer Science Fukuoka University, Japan.

R. Taniguchi and D. Arita are with the department of Intelligent Systems, Kyushu University, Japan.

on. Accordingly, the computation effort required for FSOM has been reduced to $O(D_1 + D_2 + \ldots + D_N)$, which of course, less than $O(D_1 \times D_2 \times \ldots \times D_N)$ required by SOM. In the same time, FSOM still shows, at least, same recognition accuracy of conventional SOM, if not better.

In conclusion, FSOM network is enough stable, simple and low cost alternatives to the original SOM network and viewed as a non linear extension to PCA. It is applicable until N-dimensions according to the application in hands. The outline of this paper is as follows: Section II shows overview for the original SOM, the motivation of our method and related works. Then the new, FSOM, algorithm and a simple example are provided in section III and V, respectively. Later, fair comparisons with conventional SOM and Fast DCT are given in section V. Conclusion and future works are figure out in section VI.

II. SOM & Computation Complexity

A. Overview

Conceptually, the SOM [5] simulates the functioning of the hyper column in human brain. Mathematically, it is an unsupervised learning algorithm learns the distribution of a set of patterns without any pre-class information. In the context of image processing, the conventional SOM provides a good quantization of the image samples into a topological low-dimensional space such that the input samples which are nearby in the original space are also nearby in the output space. This topological preservation of SOM makes it so useful in the classification of data even if the dimensionality of SOM is smaller than this of input data.

Consider the input data $\mathbf{X} = \{x_i, 1 < i < M\}$ belongs to a high dimensional space, i.e. $x_i = (x^{(l)}_i)_{1<l<n} \in \mathbb{R}^n$. SOM is usually represented as a neural network sheet or map whose units, usually called nodes or neurons, become tuned to different input vectors x_i. A weight vector w_j, sometimes called reference, is associated with each neuron j and the map weight vectors are given by $\mathbf{W} = \{w_j, 1 < j < N\}$; such that $N < M$.

In each training step, the following two steps are repeated for each input sample x_i.

1) Find the best matching neuron c using a similarity measure between the input and all the map's neurons. This step name is *winner-take-all* (*WTA*) where c is the desired *winner* and should satisfy:

$$\|x_i - w_c\| = \min_j(\|x_i - w_j\|) \tag{1}$$

2) Update the weigh vector of the winner c and also all its topological neighborhood in the map towards the prevailing input according to the rule:

$$w_j(t+1) = w_j(t) + h_{cj}(t)[x_i(t) - w_j(t)] \tag{2}$$

$$h_{cj}(t) = \alpha(t).\exp\left(\|r_c - r_j\| / 2\sigma^2(t)\right) \tag{3}$$

where $h_{cj}(t)$ is the neighborhood kernel function around the *winner* c at time t, $\alpha(t)$ is the learning rate and is decreased gradually toward zero and $\sigma^2(t)$ is a factor used to control the width of the neighborhood kernel. The term $\|r_c - r_j\|$ refers to the distance between the *winner* neuron c and neuron j. After the training data is exhausted, the neurons sheet is automatically organized, without external supervision, into a meaningful N-dimensional order denoted by feature map (or codebooks). Beside the advantage of topological preservation, it is demonstrated that, the Probability Distribution Function (PDF) of SOM codebook is a good approximation for the PDF of the training data [5].

B. Motivation

Generally, the higher computational cost of training artificial neural network algorithms, including SOM, limits the use of large systems capable of processing complex problems. From the computation complexity point of view, SOM is expensive approach especially when a large sized map is needed. *This is because "each" learning pass in (1) requires a computation of the distance of the "current" sample to "all" nodes in the map [7].*

It has been noted that a manageably sized maps with two dimensions admit only very few nodes along each axis direction and can, therefore, not be sufficiently smooth for many purposes where continuity is very important, as e.g. in control tasks in robotics or face applications [8]. On the other side, as the number of nodes grows exponentially with the dimensions number of the map, then using more than 2 dimensions will move the performance to be slow. In conclusion, conventional SOM search algorithm is not easily affordable for most of image recognition problems [19].

The authors of this paper observed this phenomenon and already developed a lip-reading system based on conventional SOM. Table I shows the relation among number of dimensions, recognition accuracy and recognition time; time is measured in seconds. We got these results using one of the lip-reading data sets which utilized in this paper. The most left column in Table I represents number of dimensions of the feature map. Each feature map is extended through 2, 3 and 4 dimensions with sizes 14x12, 14x12x10 and 14x12x10x8 neurons, respectively. Second column is for recognition time consumed through each dimension. Then the recognition accuracy for both training and testing phases for word unit and sentence "sent" unit are given. As it is shown, if we increase the number of dimensions (or map size) the recognition accuracy is growing up and the recognition time becomes longer.

TABLE I
RELATION AMONG NUMBER OF DIMENSIOS, TIME "SECOND" AND ACCURACY "%" FOR WORD AND SENTENCE UNITS

# Dim	Time "second"	Training Data		Testing Data	
		Word	Sent	Word	Sent
2-Dim [10]	92.4	76.9	61.1	51.6	40.1
3-Dim [11]	656.7	82.7	64.8	70.5	40.7
4-Dim	5312.8	92.3	85.2	76.9	51.8

Motivated by SOM complex computation and our previous work [9-11] using SOM for lip-reading applications, we propose a new fast search strategy can enhance the SOM competition algorithm.

C. Related Works

Several structures have been developed in order to enhance the SOM competition algorithm. Here we highlight three major types from these structures:

1) *Tree structure SOMs,*
2) *Hierarchical structure SOMs and*
3) *Randomized structure SOM.*

Regarding tree structure, in [12] an SOM tree is generated dynamically. Though the tree SOM is able to handle many complex data sets, it spends almost twice longer than the conventional SOM [13]. In [14], the layers of the tree are organized layer by layer. Therefore, the input data set needs to be input many times, which is not desirable in real-time and large data set applications [15]. In general, most tree structure approaches require alleviation for the bias brought by the tree structure, so a lateral search module is usually used in such approaches. On the other side, this lateral search will increase the computational complexity, and in same time, destroy the interlayer topology between the root neuron and leaf neurons [15].

In the hierarchy structure, one treatment is to build an SOM's map with a relatively small size and then associate additional maps to specific winner neurons. This naturally decomposes the training and recall overheads as only the smallest SOM requires a winner estimation before the problem is associated with independent SOMs [16]. In another treatment, the network is created by doubling its size periodically during training. In general, such treatments are conditioned to assuming that the topological order is optimal prior to each double step [17]. Moreover, the hierarchy map loses some properties of the conventional SOM feature map by their structure [18].

Using a randomized technique, the authors in [19-20] presented a fast search algorithm for conventional SOM. The method starts with handling a random subset from the input data and then the whole of the input data set. As stated before, the input data is feed to the system repeatedly, which is not proper in case of large data sets. The most important notice is that the method depends on two experimental parameters, to control the computation, without any theoretical paradigm. These two parameters are pixel usage ratio and neighborhood range, the network's user should follow try and error rule to decide their values.

Not only the above three structures are developed, there are other fast SOM approaches also presented. For example in [21], the authors are combining SOM with *K*-means algorithm in three stages. First stage, the authors use *K*-means to select N^2 (map size) cluster centers from the data set. Then a heuristic assignment strategy is used to organize those *N*2 selected data points, and later, they use SOM to fine tune the feature map to input samples. Of course, the computation cost became higher than using SOM only. In addition, every stage depends upon the one before in a supervised manner. Moreover, the method is conditioned for running with small number of iterations. The following section addresses the SOM feature map and how the principal components are extracted through it.

III. Feature Map and Principal Components

Let us addressing our target now as follows: *Giving a large-sized high-dimensional data space, how can we get a well-ordered feature map using SOM without consuming much time and effort and, in same time, preserving the SOM performance?* Answering this question is the topic of this paper. To this end, we utilize two lip-reading datasets. First, let's investigate and visualize this data using PCA.

A. Curved Feature Map

Fig. 1 shows a distribution of 2000 lip-reading image through the first three *PCs* (1-2), (2-3) and (1-3) in the manifold, respectively. In (a) and (b), it is easy to remark that the center of each distribution is not clear. In (c) the curved data is so clear too.

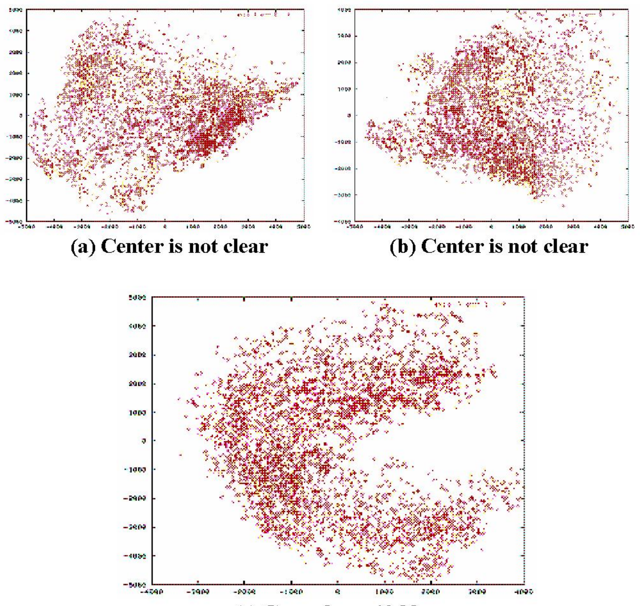

Fig. 1. Lip-Reading image distribution by the *PC*s (a)1-2, (b)2-3 and (c)1-3.

Therefore, the most important issue now is: *How can the topological orders of SOM meet with the analysis of non-linear PCs in such a curved manifold?*

B. Principal Components Extraction

As we explained in the previous section that SOM forms a discrete space (map) such that each point (node) j has a reference vector $\boldsymbol{w}_j$ indicates the corresponding point in the original space and has a neighborhood range $\boldsymbol{w}_j^k$. When a training data sample $\boldsymbol{x}_i$ is given, then SOM tries to find the best matching unit (winner) that can:

1) *Minimize the quantization error* $\sum_i \left\| \boldsymbol{x}_i - \boldsymbol{w}_c \right\|$, *included in (1), where* $\mathbf{c}$ *is the winner and given by the (WTA) rule;*

$$\boldsymbol{c} = \min_j \sum_i \left\| \boldsymbol{x}_i - \boldsymbol{w}_j \right\|,$$

2) *Minimize the total distance among the neighborhoods measured in the feature space* $\sum_{jk}\left\|w_j - w_j^k\right\|$.

To answer the question at the end of the previous section, let us imagine, artificially, the shape of our curved manifold as it is shown in Fig. 2, for two dimensions case. In this figure, to minimize the quantization error, the codebooks should be aligned along the first *PC*, which colored by blue. Then, to meet with the second factor and minimize the total distance between neighborhoods, they should be aligned in order of the lateral neighborhood which constructs the second *PC*, which colored by red. In that sense SOM is able to extract the *PCs* such that each dimension matches a principal component.

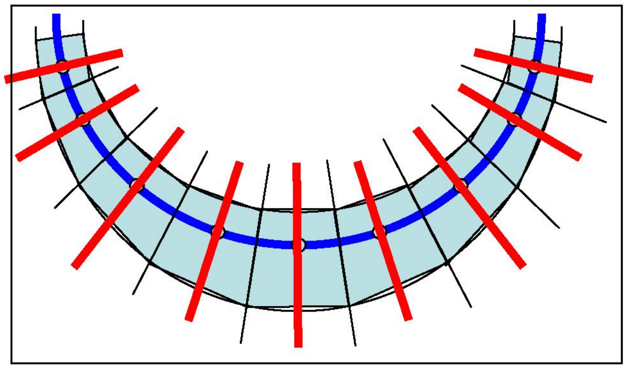

Fig. 2. Curved manifold explaining first "blue" and second "red" *PCs*.

Now, the original *N*-dimensional SOM is denoted by:

$$\mathrm{U} = \left\{ \boldsymbol{u}_{d_1,d_2,\ldots d_j} \mid d_j = 1,2,..,D_j, j = 1,..,N \right\} \tag{4}$$

where $\boldsymbol{u}_{d_1,d_2,..,d_N}$ terms to the neuron $\boldsymbol{u}$ aligned through the *N*-dimensions $d_1, d_2, .., d_N$. In each dimension $d_j = 1,2,..,D_j$, D_j refers to the maximum number of neurons distributed through the dimension *j*. Each neuron $\boldsymbol{u}_{d_1,d_2,..,d_N}$ has a codebook vector $\boldsymbol{w}_{d_1,d_2,..,d_N}$. According to (1) the *winner* neuron is decided according to the following (WTA) rule:

$$\left\| \boldsymbol{x}_i - \boldsymbol{w}_{c_1,c_2,\ldots c_N} \right\| = \min_{d_j}\left(\left\| \boldsymbol{x}_i - \boldsymbol{w}_{d_1,d_2,\ldots d_N} \right\|\right) \tag{5}$$

It is clear that, the computational steps which are required to get the winner neurons list $C = (c_1, c_2, .., c_N)$ along the *N*-dimensions $d_1, d_2, .., d_N$ using (5) is ***O*(*D*₁ x *D*₂ x … x *D*)**.

IV. Fast SOM (FSOM) Algorithm

A. FSOM Structure

It is demonstrated that, the greatest variance of the data distribution comes to lie on the low-order principal components *PCs* of the manifold. Accordingly, such low-order *PCs* often contain the most important aspects of the data set [1] [6]. In *N*-dimensions data space, the basic idea of FSOM is to measure the distance between the input pixels and the nodes distributed along these *PCs* only, instead of all the map's nodes as the conventional SOM requires in (5). Therefore, the original *N*-dimensions feature map of SOM can be viewed as "*N*" one-dimension SOM, such that each one-dimension SOM matches a principal component *PC* in the feature space.

In other words, the FSOM structure consists of the following *N*-series of one-dimension SOM:

$$\boldsymbol{u}_1 = \left\{ u_{1,d_1} \mid d_1 = 1,2,..,D_1 \right\} \tag{6}$$

$$\boldsymbol{u}_2 = \left\{ u_{2,d_1,d_2} \mid d_2 = 1,2,..,D_2 \right\} \tag{7}$$

…

$$\boldsymbol{u}_N = \left\{ u_{N,d_1,..,d_N} \mid d_N = 1,2,..,D_N \right\} \tag{8}$$

where the term u_{1,d_1} refers to the neurons aligned along the first "one-dimension SOM" d_1 (or first *PC*), u_{2,d_1,d_2} refers to the neurons aligned along the two dimensions which are the second "one dimension SOM" d_2 (or second *PC*) plus the dimension d_1 (or first *PC*) and so on. According to the above scenario we denote our method as *piecewise one-dimension SOM*. The following sections give more details.

B. Learning Phase

The learning process is described as a recursive call for the function $\mathrm{Learn}\left(1, u_{1,d_1}, c_1\right)$, where 1 is the order (*n*) of the extracted component, u_{1,d_1} refers to the neurons aligned along this component where c_1 is the winner neuron. Therefore, the function $\mathrm{Learn}\left(n, u_{n,d_1,..,d_n}, C\right)$ can be generated as given in Table II below.

TABLE II
Fast SOM learning algorithm

```
{// start of algorithm
 If ($n \neq N$)
 { 1- Train the following ($N$-$n$+1)-dimensions SOM
      $u = \left\{ u_{d_n,d_{n+1},..,d_N} \mid d_j = 1,2,..,D_j, j = n, n+1,..,N \right\}$
      using (WTA) rule in (5) and get the winner list.
   2- For each $d_n$, regard to the central column of the
      current codebook units;
      $u_n = \left\{ u_{d_n,D_{n+1}/2,..,D_N/2} \mid d_n = 1,2,..,D_n \right\}$,
      as the $n^{th}$ $PC$ and "copy" it onto $u_{n,d_n}$ .
   3- For each $d_n$, train $u_{n,d_n}$ using (WTA) rule in (1-3) and
      get the winner neuron through $d_n$.
   4- For each $n$, do: $\mathrm{Learn}\left(n+1, u_{n+1,d_{n+1}}, C\right)$
 }
 else
 { 5- For each input sample $x_i$, train each neuron
      $u_{N,d_1,..,d_N}$. That is, for each $\left(d_N = 1,2,..,D_N\right)$ apply
      the following (WTA) rule
      $\left\| \boldsymbol{x}_i - \boldsymbol{w}_{c_1,c_2,..c_N} \right\| = \min_{d_N}\left(\left\| \boldsymbol{x}_i - \boldsymbol{w}_{c_1,c_2,...,c_{N-1},d_N} \right\|\right)$
   } //end of else
 } //end of algorithm
```

where $C = c_1,\ldots,c_n$ is the winner list through n-dimensions. In step 2 in the "if" part, it is worthy to explain that we decide each *PC* by using the central column of current map and copy it to first *PC* (or u_1), then for second *PC* (or u_2) and so on until extracting all *PCs*.

The simplicity of the proposed approach is obvious as this recursive function can easily translate to a one For-loop statement. In this "For-loop", the function $\text{Learn}\left(n, u_{n,d_n}, C\right)$ calls the function $\text{Learn}\left(n+1, u_{n+1,d_{n+1}}, C\right)$ D_n times. Now, as the sizes of $u_{n+1,d_{n+1}}$ is $1/D_n$ of u_{n,d_n}, therefore, the computational complexity to train u_{n+1} is $1/D_n$ of the complexity to train u_n. In notations, if the computational complexity to train u_n (or conventional N-dimension SOM) is $\boldsymbol{C}$, then, the overall complexity of the FSOM is

$$\boldsymbol{C}\left(1 + 1/D_1 + 1/D_1 D_2 + \cdots + 1/\prod_i D_i\right) \approx \boldsymbol{C} \qquad (9)$$

This means that the consumed time to train the FSOM is "approximately" the same of that to train conventional SOM. So, the new learning method does not add any extra load or burden to the conventional SOM training phase.

C. Recognition (Competition) Phase

In image recognition domain, the most challenge is to achieve recognition in a real time or near to real time. In FSOM, after getting a well ordered feature map during learning phase, now we try to get the winner list through recognition phase using the following N-steps in turn.

- First, (WTA) rule is applied to select first *winner* c_1 from first *PC* (or u_1) in (6) according to:

$$\left\|x_i - w_{1,c_1}\right\| = \min_{d_1}\left(\left\|x_i - w_{1,d_1}\right\|\right) \qquad (10)$$

- Second *winner* c_2 is picked from second *PC* (u_2) in (7) as:

$$\left\|x_i - w_{2,c_1,c_2}\right\| = \min_{d_2}\left(\left\|x_i - w_{2,c_1,d_2}\right\|\right) \qquad (11)$$

- Finally, the *N-winner* c_N is picked from the N^{th} *PC* (or u_N) given in (8):

$$\left\|x_i - w_{N,c_1,c_2,\ldots,c_N}\right\| = \min_{d_N}\left(\left\|x_i - w_{N,c_1,c_2,\ldots,c_{N-1},d_N}\right\|\right) \qquad (12)$$

Obviously, computation efforts (or steps) during FSOM recognition phase is $\boldsymbol{O(D_1 + D_2 + \ldots + D_N)}$. Of course, this amount is less than that of the conventional N-dimension SOM in (5); which is $\boldsymbol{O(D_1 \times D_2 \times \ldots \times D_N)}$. According to the above scenario, FSOM consumes less computation time than conventional SOM during recognition stage.

V. EXAMPLE

Let's consider a simple example to illustrate how the new algorithm runs. For instance, consider the case of 3-dimensions; i.e. we have d_1, d_2, d_3, such that $d_j = 1, 2, .., D_j$. For further simplicity, consider D_1=30, D_2=20 and D_3=10, respectively.

A. Learning Phase in 3-dimensions

According to the training algorithm given in Table II, the training steps here will run in 3 stages as follows:

(1)

- For each input sample $\boldsymbol{x_i}$, train the normal SOM in 3-dimensions (u_{3,d_1,d_2,d_3}) using (1) (or (5)) as:

$$\left\|x_i - w_{3,c_1,c_2,c_3}\right\| = \min_{d_i}\left(\left\|x_i - w_{3,d_1,d_2,d_3}\right\|\right) \qquad (13)$$

- Update the codebook of the winner neurons and their neighbors using (2-3).

(2)

- For each d_1, put

$$u_{1,d_1} = u_{3,d_1,d_2/2,d_3/2} \qquad (14)$$

- $$\left\|x_i - w_{1,c_1}\right\| = \min_{d_1}\left(\left\|x_i - w_{1,d_1}\right\|\right) \qquad (15)$$
- $$\left\|x_i - w_{3,c_1,c_2,c_3}\right\| = \min_{d_i}\left(\left\|x_i - w_{3,c_1,d_2,d_3}\right\|\right) \qquad (16)$$
- Update the codebook of the winner neurons and their neighbors using (2-3).

(3)

- For each d_1 and d_2, put

$$u_{2,d_1,d_2} = u_{3,d_1,d_2,d_3/2} \qquad (17)$$

- $$\left\|x_i - w_{1,c_1}\right\| = \min_{d_1}\left(\left\|x_i - w_{1,d_1}\right\|\right) \qquad (18)$$

  get c_1.
- $$\left\|x_i - w_{2,c_1,c_2}\right\| = \min_{d_2}\left(\left\|x_i - w_{2,c_1,d_2}\right\|\right) \qquad (19)$$

  get c_2.
- $$\left\|x_i - w_{3,c_1,c_2,c_3}\right\| = \min_{d_3}\left(\left\|x_i - w_{3,c_1,c_2,d_3}\right\|\right) \qquad (20)$$

  get c_3.

B. Recognition Phase in 3-Dimension

It supposed that after the training phase, the SOM feature map is became a well ordered map and has all the qualities of conventional SOM feature map. To achieve the recognition task, we exploit the sequence in (10-12) as follows:

- First, winner-take all rule is applied to first *PC*, then first *winner* $\boldsymbol{c}$ is selected by:

$$\left\|x_i - w_{1,c}\right\| = \min_{l}\left(\left\|x_i - w_{1,l}\right\|\right) \qquad (21)$$

- Second, we will concern, ONLY, with the neurons of second *PC*, which already passing through first winner $\boldsymbol{c}$. Then second winner $\boldsymbol{d}$ is selected from:

$$\left\|x_i - w_{2,c,d}\right\| = \min_{j}\left(\left\|x_i - w_{2,c,j}\right\|\right) \qquad (22)$$

- Similarly, we concern, ONLY, with the neurons of third *PC* which passing through first and second *winner* $\boldsymbol{c}$ and $\boldsymbol{d}$, respectively. Then third *winner* $\boldsymbol{m}$ is selected from;

$$\left\|x_i - w_{3,c,d,m}\right\| = \min_{k}\left(\left\|x_i - w_{3,c,d,k}\right\|\right) \qquad (23)$$

Obviously, computational efforts during FSOM recognition phase is (30+20+10) steps or (60) steps. In contrast, to exploit conventional SOM with same number of dimensions and same number of neurons you need (30x20x10) which is equal to (6000) steps. This means that, the required effort to use FSOM is "1%" from that which is required for using conventional SOM. Please note that, this speedup rate is only for this example, therefore, if you increase or decrease each dimension size (*D*'s) the speedup rate will increase or decrease accordingly.

VI. EXPERIMENTAL RESULTS

A. *First Experiment- Artificial data*

Here we exploit a kind of artificial data to test FSOM's ability; this is considered as a kind of academic judgment [2]. We use a package able to plot normal and curved manifolds randomly. Then we feed these data to FSOM directly. Let's first show how we created such artificial data.

In case of single class data, if we have a data set denoted by $D = \{\mathrm{d}_j \mid j = 1, 2, .., N\}$, where

$$d_i = [d_{i1}, d_{i2}, .., d_{in}, .., d_{iN}]^T \tag{24}$$

is N-dimensional data and generated from the M-dimensional data source;

$$S_i = [s_{i1}, s_{i2}, .., s_{im}, .., s_{iM} \mid M \le N]^T \tag{25}$$

by using the following quadric transformation:

$$d_{in} = S_i^T A_n S_i, \tag{26}$$

where $\boldsymbol{A}$ is a (M+1) x (M+1) matrix**.** The data source S_i is generated as a random data in a range of $-1 \le s_{im} \le 1$, or as a Gaussian distribution data $P(s_i) = N(0,1)$. Fig.3 shows the generation of curved manifold in case of single class; such kind of artificial data can also be generated in case of multiple classes [2].

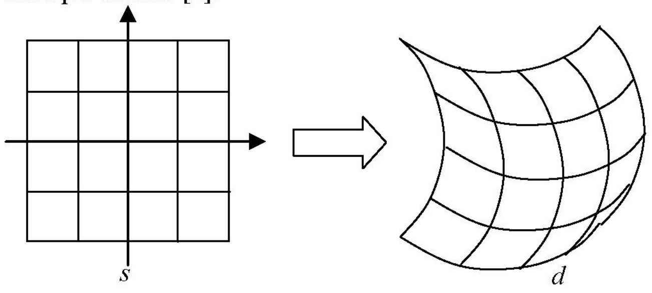

Fig. 3. Transform a plain manifold to a curved one

Exploiting the above data generator, we generated 6 different types of artificial data (plain and curved). Namely, we generated: Triangle, curved Triangle, Gaussian, curved Gaussian, Trapezoid and Lozenge data type. Due to space limitation we will show only the distribution of two types of them using 2-dimensions FSOM; since 2-dim is so enough for such kind of applications. Fig. 4 and 5 shows the distribution of: "curved" Gaussian and Trapezoid data types, respectively. The blue points represent the FSOM map nodes whereas the pink line connects the nodes of the first *PC*. As it is clear in the curved Gaussian type, Fig.4, how the first *PC* bends with the data bending. Also, in the other distribution, Fig. 5, the first *PC* is centralized and seems natural and coincides with the data distribution.

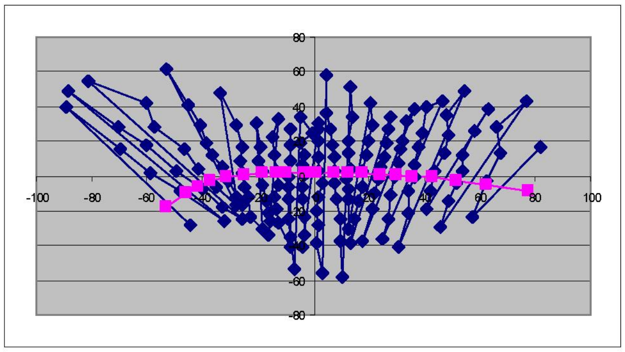

Fig. 4. Curved Gaussian distribution including first *PC* nodes (pink)

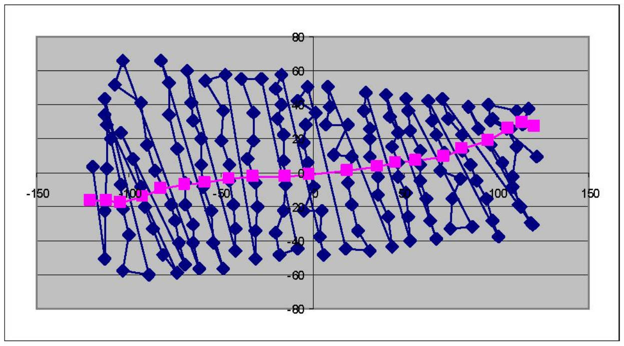

Fig. 5. Trapezoid data distribution

B. *Second Experiment-Lip Reading*

1) *Databases overview*

The authors of this paper are already presented a lip-reading system applied for Japanese [9] and Arabic [10] data sets; *please note that, the results of [9] and [10] are updated in [11].* Images of each set are captured from 9 native subject, *6 subjects for training and 3 for test*, such that each subject uttered 9 different sentences using his own language. For Japanese data set, the number of image is 5670 gray image divided as 3780 image for training and 1890 for test. For Arabic data set, the number of image is 5760 image divided as 4320 for training and 1440 for test. Each original image resolution is 160x120 pixels. In this paper, we used the above resolution to conduct the comparison between FSOM and SOM. However, to run Fast Discrete Cosine Transform (FDCT) the input image dimensions should take the power of 2 (i.e. 2^x), therefore, to conduct a comparison between FSOM and FDCT we crop each original image to the size of 128x128 to just show the mouth area. Examples for the used images are shown in Fig. 6. For more details about the two data sets and used sentences please refer to [9-11].

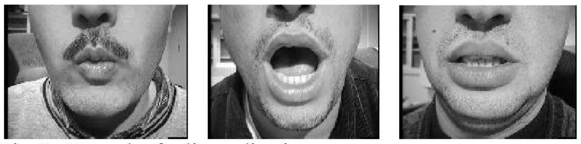

Fig. 6. Examples for lip reading images

2) Experiments overview

Exploiting same data sets, same dictionary and same environment, we provide here a comparison among SOM, FDCT and FSOM. For all models in the paper, we used Hidden Markov Model (HMM) as a recognizer, as it is given in details in [9-11]. Regarding to recognition "accuracies", the experiments have been conducted using different number of dimensions and different number of neurons, and finally, the "best" accuracy over these runs is recorded below for each model for each data set. Regarding to the recognition "time", all experiments are conducted on the same machine, Pentium IV Intel 2.0 GHz. The time unit in the below comparisons is "second". Since our main motivation is recognition time, the column "S.R." in the tables given below represents Speedup Ratio or *the ratio of SOM recognition time to FSOM recognition time*.

3) Comparison between SOM and FSOM

The feature map of both SOM and FSOM model is extended through 2, 3 and 4 dimensions.

A. *For Arabic data set*, the "best" size of each dimension was 14x12, 14x12x10 and 14x12x10x8 neuron, respectively.

TABLE III
ARABIC DATA SET: RECOGNITION TIME "SECOND"

# Dim	SOM	FSOM	S.R.
2	92.4	15.1	6
3	656.7	36	18
4	5312.8	101.9	52

TABLE IV
ARABIC DATA SET: RECOGNITION ACCURACY "%"
IN 3 DIMENSION (14X12X10)

Model	Training data		Test data	
	Word	Sent	Word	Sent
SOM	82.7	64.8	70.5	40.7
FSOM	**85.3**	**70.4**	**75.6**	**51.6**

As it is shown in Table III, the required recognition time to run FSOM is less than this required for original SOM by 6 times in case of 2 dimensions. As we increase the number of dimensions (or the map size) as the time required for FSOM becomes less and less than this which is required for SOM by 18 and 52 times for 3 and 4 dimensions respectively. In Table IV, the accuracy results are for the case of 3-dimensions 14x12x10 for both models. It is clear that FSOM accuracies are better than this for SOM for both word and sentence units through training and test phases. In our opinion, this is due to that FSOM concentrate the information contained in the feature space in few neurons, which are aligned along principal components. From these neurons FSOM choose the best matching one for input sample.

B. *For Japanese data set*, one of the best sequences for each dimension is 11x9, 11x9x7 and 11x9x7x5 neuron, respectively. In Table V, and similar to Arabic data set, as we increase the number of dimensions as the recognition time of FSOM is less than this of SOM. Also as we increase the number of dimensions as SOM became slower than FSOM.

The accuracy results given in Table VI are for the case of three dimensions 11x9x7 for both models. Unlike Arabic data set, for Japanese set the original SOM performs better than FSOM for training data. However, for test data or new subjects, which is more challenging, FSOM performs like SOM for word unit and exceeds SOM for sentence unit.

TABLE V
JAPANESE DATA SET: RECOGNITION TIME "SECOND"

# Dim	SOM	FSOM	S.R.
2	144.7	25.2	5.7
3	713.6	51.4	14
4	6652.1	154.7	43

TABLE VI
JAPANESE DATA SET: RECOGNITION ACCURACY "%"
IN 3 DIMENSION (11X9X7)

Model	Training data		Test data	
	Word	Sent	Word	Sent
SOM	**89.8**	**83.3**	**63**	48.2
FSOM	86.1	77.8	**63**	**55.6**

From the above results and comparison it is easily to remark the promising of our method than original SOM because it does not require much efforts SOM and, in the same time, preserving the quality of SOM by behaving at least same performance if not better. In other words, FSOM is a proper and fast alternate for the conventional SOM.

4) Recognition Time for One Image

Moreover, the recognition time for one image with a resolution 160x120 pixels using FSOM is less than this using conventional SOM. Definitely, using a feature map in 3 dimensions, the recognition time for one image using SOM and FSOM is 0.104 and 0.005 "second", respectively. In other words, FSOM can process 200 image frames per second in computer Pentium IV 2.0 GHz.

5) Comparison between FSOM and FDCT

It is demonstrated that Fast Discrete Cosine Transform (FDCT) is one of the fastest state-of-the-art algorithms which can perform image compression in a well manner and by a similar way to the Fast Fourier Transform (FFT) [22]. The authors of this paper already showed this fact experimentally when we compared recognition time of FDCT with that of SOM which was longer [11]. Also we showed experimentally in [11] that FDCT accuracy is better a little than this of SOM for the data sets of lip-reading.

Unlike SOM, here we show that FDCT is not faster than FSOM. Table VII provides a comparison between the recognition time of FDCT and the proposed algorithm FSOM. It is clear that FSOM is faster than FDCT by about sixth times for both data sets exploited here.

TABLE VII
COMPARISON OF COMPUTATION TIME BETWEEN FSOM AND FDCT

Data	FDCT	FSOM
Japanese	302.8	47.2
Arabic	374.4	58

Also Table VIII shows a comparison between FDCT and FSOM for recognition accuracy. The feature map of FSOM is in 3 dimensions including 14x12x10 neuron for both data sets. For FDCT we chose the number of coefficients which gives higher accuracies which was 7 coefficients for Japanese and 8 for Arabic. Due to using FDCT, the resolution of input image for both models is 128x128 pixels.

TABLE VIII
RECOGNITION ACCURACY "%" COMPARISON BETWEEN THE THREE MODELS

Model	Japanese		Arabic	
	Word	Sent	Word	Sent
FDCT	68.5	51.9	73.1	48.2
FSOM	74.1	51.9	64.1	48.1

It is obvious that both models show same performance approximately. For example, FSOM shows better accuracy than FDCT as in the case of Japanese word unit, whereas FDCT gives better performance in Arabic word unit also. Both of them give same accuracy for Japanese and Arabic sentence unit. But regarding to the superiority of FSOM in recognition time then we can also regard to FSOM as a better and accelerated alternate to fast DCT.

VII. CONCLUSION AND FUTURE WORK

In this paper we presented a fast search algorithm for SOM able to reduce its computational complexity and, in the same time, preserving the basic quality of SOM. The new SOM algorithm is based on the fact that most of the variance of the data distribution comes to lie on the low-order axes; or principal component. Accordingly, it is widely known that these low-order components often contain the most important features of the data stream. Therefore in *N*-dimensions feature space, the proposed algorithm extracts *N* "one-dimension SOM" such that each "one-dimension SOM" matches a principal component. Then the search about best matching unit will be through the nodes of these components "only" instead of the search through the nodes of the "entire" map, as it in the original SOM.

Exploiting two lip-reading data sets, we showed that FSOM needs computation time much less than the conventional SOM. In addition, recognition time using FSOM is less than this of FDCT using same data bases. On the other hand, FSOM shows same accuracies, or may be better a little bit, than conventional SOM and fast DCT. According the performance presented in this paper, we expect that FSOM impact will be great in case of using multivariate or large data sets which require a large size feature map especially for complex applications. We are planning to investigate this matter using some of face recognition databases.

REFERENCES

[1] I. Jolliffe, *Principal Component Analysis*. 2nd Ed, Springer, 2002.

[2] B. Chalmond and S.C. Girard, "Nonlinear Modeling of Scattered Multivariate Data and Its Application to Shape Change", *The IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp 422-432, May 1999.

[3] J. Twining and J. Taylor, "The use of kernel principal component analysis to model data distributions," Pattern Recognition, vol 36, no. 1, pp 217-227, 2003.

[4] R. Saegusa, H. Sakano and S. Hashimoto, "Nonlinear principal component analysis to preserve the order of principal components", *Neurocomputing, Elsevier,*vol 61, pp. 57 – 70, 2004.

[5] T. Kohonen, *Self Organizing Maps*. 3rd edition, Springer series in information sciences, 2001.

[6] B. Lu and W. W. Hsieh, "Simplified nonlinear principal component analysis", in *Proc. Int. J. Conf. on Neural Networks IJCNN03*, vol 1, pp. 759 – 763, July 2003.

[7] S. Lawrence, C. Giles, A. Tsoi and A Back, " Face Recognition: A Convolutional Neural Network Approach," *The IEEE Trans. on Neural Networks, Special Issue on Neural Networks and Pattern Recognition*, vol 8, no. 1, pp. 98–113, 1997.

[8] J. Walter and H. Ritter," Rapid learning with parametrized self-organizing maps," *Neurocomputing, Elsevier,* vol. 12, pp 131-153, 1996.

[9] A. Sagheer, N. Tsuruta, R. Taniguchi, S. Maeda, "Visual Speech Features Representation for Automatic Lip-Reading," *Proc. the 30th IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP05,* vol 2, March 2005, pp 781 – 784.

[10] A. Sagheer, N. Tsuruta, R. Taniguchi, S. Maeda, "Hyper-Column Model vs. Fast DCT for Feature extraction in Visual Arabic Speech Recognition," *Proc. IEEE Int. Symposium in Signal Processing and Information Technology, ISSPIT05*, Dec. 2005, pp. 761-766.

[11] A. Sagheer, N. Tsuruta, R. Taniguchi, S. Maeda, "Appearance Features Extraction vs. Image Transform for Visual Speech Recognition," *Submitted to the Journal of Japanese Society of Information Processing Systems (IPSJ).*

[12] J.A.F. Costa, M.L. de Andrade Netto, A new tree-structured self-organizing map for data analysis, *Proc of Int. J. Conf. on Neural Networks*, vol. 3, pp. 1931–1936, 2001.

[13] H. Jin, W. Shum, K. Leung and M. Wong, "Expanding Self-Organizing Map for data visualization and cluster analysis," *Information Sciences*, vol. 163, pp157–173, 2004.

[14] P. Koikkalainnen, "Progress with the tree-structured self organizing map," *Proc. the 11th European Conf. on Artificial Intelligence ECAI94*, San Diego, CA, 1994, pp. 211–215.

[15] P. Xu and C. Chang, "Self-Organizing Topological Tree," *Proc of the Int. Symposium on Circuits and Systems, ISCAS04,* vol 5, May 2004, pp732 -735.

[16] P. Suganthan, "Pattern classification using multiple hierarchical overlapped self-organising maps Pattern Recognition, vol 34, no. 11, pp 2173-2179, 2001.

[17] A. Rauber, D. Merkl and M. Dittenbach, "The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data", *IEEE Trans. on Neural Networks*, vol 13, no 6, pp1331 – 1341, 2002.

[18] Y. P. Jun, H. Yoon, and J. W. Cho, "L learning: A fast self-organizing feature map learning algorithm based on incremental ordering," *IEICE Trans. Inform. Syst.*, vol. E76, no. 6, pp. 698–706, 1993.

[19] T. El. Tobely, N. Tsuruta, M. Amamiya, "Online speech-reading system for Japanese language," *Proc 9th Int. Conf. on Neural Information Processing, ICONIP 00,* 2000, vol 3, pp 1188 – 1193.

[20] T. El. Tobely, N. Tsuruta, M. Amamiya, "Randomized Self-Organizing Maps and its Application," *Proc of the 6th Int. Conf. on Soft Computing*, pp 588 – 596, 2000.

[21] M. Su, H. Chang, "Fast self-organizing feature map algorithm," *IEEE Trans. Neural Networks*, vol. 11, no. 3, pp721–733, 2000.

[22] K. Rao and P. Yip, *Discrete cosine transform : algorithms, advantages, applications*. Academic Press, 1990.