# A New Vector Quantization Approach via Self-Organizing Map

Lixin Xu†, W. Q. Liu‡ and V. Svetha‡
† : Department of Automatic Control, Beijing Institute of Technology, P.R. China
‡ : School of Computing, Curtin University of Technology,
WA, 6102, Australia Email : wanquan@cs.curtin.edu.au

*Abstract In this paper, a new algorithm is proposed for vector quantization based on self-organizing map (SOM) network. First, a conventional self-organizing map is modified to deal with dead codebooks in the learning process and is used to obtain the codebook distribution structure for given input data. Second, it is proved that the modified SOM in the case of one codebook will converge globally to the mean value of the given input data. Thirdly, the Structured Self-Organizing Map (SSOM) and modified SSOM are proposed in order to obtain desired codebook locations. Finally, extensive simulations prove that the modified SSOM is very effective.*

**Key words :** *Codebook Construction, Self-organizing Map, LBG-U, Vector Quantization.*

## 1 Introduction

Vector quantization (VQ) [1] is an important technique for data compression. Roughly speaking, the principle of vector quantization can be described as follows : Given a sample data set $D$ with $|D| = N$, of $n$-dimension data vector $\xi_1, \xi_2, \cdots, \xi_N$, choose a set $C$ with $|C| = M$ with $M << N$, forming $n$ -dimensional codebook vectors $w_1, w_2, \cdots, w_M$. To transfer any data vector $\xi_i$ to the receiver, the sender only need transfer the index $j$ of the codebook $w_j$, which is the nearest codebook to $\xi_i$ according to curtain distance measure. Usually, the squared Euclidean distance is adopted.

$$d(a,b) = \sum_{i=1}^{n}(a_i - b_i)^2 \quad for\ a, b \in R^n \tag{1}$$

On the other hand, the received data differs from the original. A distortion error is defined to evaluate the quality of codebook construction.

$$E(D,C) = \sum_{\xi \in D} d(\xi, w_\xi) \tag{2}$$

where $w_\xi$ stands for the codebook that the sample data $\xi$ belongs to.

It can be seen that codebook construction is a key issue in VQ. Many codebook construction approaches have been proposed in which one tries to find a codebook $C$ such that it can minimize the distortion error in (2) for a given data set $D$. For random given data set, the corresponding optimization problem is a complicated nonlinear problem and only local optimal solution depending on initial codebook selection can be obtained via standard optimization techniques, such as LBG and its variants [3]. Recently, an intuitive LBG-U method is proposed in [4] via a new utility measure. This new approach can keep the distortion error (2) not increasing in every iteration and this will guarantee the convergence of the proposed LBG-U algorithm. Simulation results show the effectiveness of LBG-U though extra computing burden added.

Also some neural network approaches have been proposed for codebook construction [7, 8, 9, 10]. Among these algorithms, self-organizing feature map (SOM) [5] plays an important role since it can detect the codebook distribution structure for a given sample data set without prior knowledge. This feature will help us construct a reasonable codebook. In [8, 9], the convergence analysis for SOM algorithm was given when the input data was a sequence of i.i.d. one dimensional random variable with uniform distribution. However, the data distribution is usually unknown in practice, some of these algorithms are not so useful in practice though they are very important theoretically. Among all the neural network algorithms, a critical issue is that the convergence rate is very slow due to long time learning though many speedup techniques are proposed [10]. Recently, the idea of quantum classification was proposed in [11] and it can not be implemented numerically at present stage [11].

In this paper, a new classification algorithm called structured Self-Organizing Map (SSOM) will proposed based on SOM. There are several advantages for SSOM. Firstly, its initial codebook can be chosen randomly. Se-

condly, the local optimum issue associated with LBG will be overcome by SSOM. These are due to the characteristics for SOM. Thirdly, The slow convergence issue will be overcome since LBG will be applied with obtained codebooks from the modified SOM as initial values. Theoretically, we can prove that the SOM algorithm will converge to the mean value of the input data in case of one codebook. This motivates us to stop SOM learning after the codebook structure has been detected and this will speed up the process of classification. One more special character for SOM is that , in the process of competitive learning, some codebooks may be dead (which happens often in the SOM algorithm). In this paper, a new algorithm is designed to modify SOM and then speed up the learning process.

The structure of this paper is as follows. Some preliminary results on LBG and SOM are presented in section 2. The convergence analysis of SOM with one codebook is proved in section 3 for any input data set. The new SSOM algorithm will be proposed in section 4 which can deal with dead codebooks after modifying SOM. In section 5, extensive simulations are impletemented and compared with SOM and LBG-U. Some conclusions will be given in section 6.

# 2 Preliminary Results

## 2.1 The LBG Method

LBG is a famous codebook construction method in the existing literature. Since both SSOM proposed in this paper and the recent LBG-U approach are based on this approach, we will explain LBG in detail here in order to do comparision with LBG-U in this paper.

For notational convenience, let us denote the set of data vectors associated with a particular codebook $w_i$ as

$$R_i = \{\xi \in D | I(\xi) = i\} \tag{3}$$

where

$$I(\xi) = min\{i | (d(w_i, \xi) \leq d(w_j, \xi)) \ \ \forall j \in \{1, \cdots, M\}\} \tag{4}$$

Now the LBG algorithm can be expressed as

1. Randomly initialize codebooks $C$ from data set $D$.
2. Determine the set $R_i$ for all $w_i \in C$.
3. Calculate the new codebook

$$w_i = \frac{1}{|R_i|} \sum_{\xi \in R_i} \xi \ \ (for\ all\ w_i \in C) \tag{5}$$

4. Repeat step2 and step3 until no $w_i$ changes anymore.

LBG algorithm is guaranteed to decrease the distortion error $E(D, C)$. The finial codebook corresponds to a local minimum of the distortion error function (2) [1]. Also, It should be noted that one round LBG iteration includes much operations. In order to overcome the local optimimum issue, a recent LBG-U [4] is proposed and proved to be much better improvement for LBG with mild extra computing cost. Actually, LBG-U takes much computing cost compared to LBG since in each step, several utility measures must calculated and compared in order to seek the minimum and maximum vectors from the codebook.

It should be noted that with a better initial codebook, LBG can achieve much better resuults. This indicates that it is important to choose a better initial codebook set in order to achieve better result with LBG. Further, one can prove that LBG will achieve global optimum in case of only one codebook.

## 2.2 The Self-organizing Map

SOM is developed by Kohonen [5]. One of its significant features is that it can preserve the most important topological and metric relationships of the primary data items. Actually, SOM is a kind of competitive neural network and always composed of one or two dimensional array of processing elements or neurons in the input space. All these neurons receive the same inputs from external world. Learning is accomplished by iterative operation for unlabled input data. In the training process, the neurons evolve in the input space in order to approximate the distribution function of the input vector. This structure property is very suitable for codebook construction, especially for problems with high dimensional input space. In the current literature, either the modified SOM is directly used in codebook construction [12] which is time consuming, or some special distribution is required for the input data [8, 9] with limited applications. In order to explain our new algorithm in this paper, we describe SOM in some detail as below.

The model of SOM here is a one-dimensional array of $M$nodes. To each neuron $C_i, \quad i = 1, 2, \cdots, M$, a weight vector $w_i = (w_{i1}, w_{i2}, \cdots, w_{in})^t \in R^n$ is defined. During learning process, a randomly selected input vector $x \in R^n$ from the training set will be connected to all neurons in parallel. At the kth step, we select the vector $x$ to a winning neuron $C_l$ according to the following competitive rule.

$$\|x - w_l^{[k]}\| = \min_i \|x - w_i^{[k]}\| \quad (6)$$

In this case, all the neurons within a certain neighbourhood around the winning neuron will participate in the weight-update process. With random initial $w_l^{[0]} (0 \leq i \leq n)$, this learning process can be described by the following iterative procedure.

$$w_i^{[k+1]} = w_i^{[k]} + h_{li}^{[k]}(x^{[k]} - w_i^{[k]}) \quad (7)$$

where $h_{li}^{[k]}$ is the neighbour-hood function which can be chosen as Gaussian function [13]

$$h_{li}^{[k]} = \alpha^{[k]} exp(-\frac{d^2(l,i)}{2(\sigma^{[k]})^2}) \quad (8)$$

where $d(l,i)$ is the Euclidean distance between the node $l$ and $i$, $\alpha^{[k]}$ is the learning-rate factor and $\sigma^{[k]}$ is the width of the Gaussian function at the iteration $k$. As the iteratiion increases, $\alpha^{[k]}$ tends to zero and the width of Kernel function tends to one. In practice, $\alpha^{[k]}$ can be chosen as

$$\alpha^{[k]} = \alpha^{[0]}(1 - \frac{k}{T}) \quad (9)$$

where $T$ is the total iteration number.

**Remark 2.1**

1. *In the iteration process given in (7), the initial codebook can be given reandomly. With the learning process, the codebook distribution structure can be obtained gradually. This may provide a better initial codebook set for LBG.*
2. *In the learning process, the parameter $h_{li}^{[k]}$ will tend to zero as $k \to \infty$. In practice, one can regard it as a very small positiove constant parameter. This is helpful for the proof of convergence of SOM in the next section.*
3. *Much work has been done for the convergence analysis of SOM with known distribution of the input data [8, 9]. In general case, the convergence analysis is very hard if not impossible.*

# 3 The Structured Self-Organizing Map

The codebook construction method based on (7) has been investigated by several researchers [13, 8]. All these researchers used this method directly and one critical issue is time-consuming though some speedup techniques were proposed [10]. In order to overcome this shortcoming, we need to investigate the convergence issue when the codebook structure is obtained after some period learning. In order to describe the stability of codebook structure explicitly, the following definition is given.

First, let us give some notations. In each iteration given by (7), the codebook is denoted as $\{w_j^{[k]}\}$ in $kth$ step, $j = 1, 2, \cdots, M$. Also the set of input data associated with $\{w_j^{[k]}\}$ is denoted as $\{R_i^{[k]}\}$. With these notations, the following notations can be given.

**Definition 3.1** *The SOM given in (7) is said to be structural Stable if for any initial codebook set $\{w_j^{[0]}\}$, $j = 1, 2, \cdots, M$, there exists some number $K$ such that for all $k, l > K$, we have*

$$\{R_i^{[k]}\} = \{R_i^{[l]}\}, \quad i = 1, 2, \cdots, M$$

Though *structural Stable* for SOM is very hard to prove if not impossible, extensive simulations show that SOM will have this kind of property in practical application [13]. This motivates us that SOM will provide better codebook distrbution. Further, with one codebook case associated with each $\{R_i^{[k]}\}$, we will have the following important result.

## 3.1 Local Convergence Analysis

**Theorem 3.1** *Assume that there is only one codebook, Then the SOM given in (7) will converge to the average value of all the input data.*

**Proof** With assumption, SOM algorithm can be written as

$$w_i^{[k+1]} = w_i^{[k]} + h_{li}^{[k]}(p_j^{[i]} - w_i^{[k]}) \quad (10)$$

where $p_j^{[i]}$, $j = 1, \cdots, N$ is the input data. As discussed in Remark 2.1, $h_{li}^{[k]}$ can be assumed to be a very small positive constant $\eta$. Now one can rewrite (10) as

$$w_i[k+1] = w_i[k] + \eta(p_j^{[i]} - w_i[k]) \quad (11)$$

where $0 < \eta < 1$. In every epoch $k$, all the $p_j^{[i]}$, $j = 1, \cdots, N$, will be selected randomly in the iteration (11). For notational convenience, symbol $k+1$ doesn't change during this steps. After learning process to all the input data, one will have

$$w_i[k+1] = \eta p_N^{[i,k]} + \eta(1-\eta)p_{N-1}^{[i,k]} + \cdots + \eta(1-\eta)^{N-1}p_1^{[i,k]} + (1-\eta)^N w_i[k] \quad (12)$$

Where $k$ in $p_j^{[i,k]}$ stands for the input data sequence of the $k$th epoch. Theresore, after epoch $k+1, k+2, \cdots, k+l-1$, one will obtain

$$
\begin{aligned}
w_i[k+1] &= \eta p_N^{[i,k]} + \eta(1-\eta)p_{N-1}^{[i,k]} + \cdots + \\
&\eta(1-\eta)^{N-1}p_1^{[i,k]} + (1-\eta)^N w_i[k] \\
w_i[k+2] &= \eta p_N^{[i,k+1]} + \eta(1-\eta)p_{N-1}^{[i,k+1]} + \cdots + \\
&\eta(1-\eta)^{N-1}p_1^{[i,k+1]} + (1-\eta)^N w_i[k+1] \\
\vdots & \qquad\qquad \vdots \\
w_i[k+l] &= \eta p_N^{[i,K+l-1]} + \eta(1-\eta)p_{N-1}^{[i,k+l-1]} + \cdots + \\
&\eta(1-\eta)^{N-1}p_1^{[i,k+l-1]} + (1-\eta)^N w_i[k+l-1] \qquad (13)
\end{aligned}
$$

Let's analyze the right part of above equations. Except for the last term $w_i$, all other items are the weighted sums of all the input data. Since $p_j^{[i]}$ is selected randomly, if the epoch $l$ is large enough, the possibility for an input vector $p_j$ appearing in one of the positions of $N$ will be same. Adding all above equations together, one will get

$$
\begin{aligned}
& w_i[k+1] + w_i[k+2] + \cdots + w_i[k+l] = \\
& (1-\eta)^N (w_i[k] + w_i[k+1] + \cdots + w_i[k+l-1]) \\
+ & \sum_{j=1}^{N} p_j \left(\frac{l}{N}(\eta + \eta(1-\eta) + \cdots + \eta(1-\eta)^{N-1})\right) \qquad (14)
\end{aligned}
$$

i.e.,

$$
\begin{aligned}
& w_i[k+1] + w_i[k+2] + \cdots + w_i[k+l] = \\
& (1-\eta)^N (w_i[k] + w_i[k+1] + \cdots + w_i[k+l-1]) \\
& + \frac{l}{N} \sum_{j=1}^{N} p_j [1 - (1-\eta)^N] \qquad (15)
\end{aligned}
$$

Adding one more item on both sides, one will get

$$
\begin{aligned}
& w_i[k+1] + w_i[k+2] + \cdots + w_i[k+l+1] = \\
& (1-\eta)^N (w_i[k] + w_i[k+1] + \cdots + w_i[k+l]) \\
& \quad + \tfrac{l+1}{N} \textstyle\sum_{j=1}^{N} p_j [1 - (1-\eta)^N] \qquad (16)
\end{aligned}
$$

By (16)-(15), one will obtain

$$
w_i[k+l+1] = \frac{1}{N}[1-(1-\eta)^N] \sum_{j=1}^{N} p_j + (1-\eta)^N w_i[k+l] \qquad (17)
$$

Let $y = \frac{1}{N}[1-(1-\eta)^N] \sum_{j=1}^{N} p_j$, Then (17) will become

$$
\begin{aligned}
w_i[k+l] &= y + y(1-\eta)^N + y(1-\eta)^{2N} + \cdots + \\
& y(1-\eta)^{(l-1)N} + (1-\eta)^{lN} w_i[k] = \\
& y\tfrac{1-(1-\eta)^{lN}}{1-(1-\eta)^N} + (1-\eta)^{lN} w_i[k] \qquad (18)
\end{aligned}
$$

Let $l \to \infty$, then (18) will be

$$
w_i[k+l] = \frac{1}{N}[1-(1-\eta)^N] \sum_{j=1}^{N} p_j \frac{1}{1-(1-\eta)^N} = \frac{1}{N} \sum_{j=1}^{N} p_j \qquad (19)
$$

due to

$$
(1-\eta)^{lN} \to 0, \quad l \to \infty
$$

□.

**Remark 3.1** *The theorem indicates that in the case of one codebook, the SOM will converge to the average value of all the input data. This is same as LBG, which will give a global optimum in one codebook case. This motivates us to stop SOM and calculate the average value in each $\{R_i\}$ when the codebook structure is stable.*

## 3.2 Structured Self-Organizing Map

Inspired by the convergence analysis of SOM in one codebook case as well as the topological grasping ability of SOM, we propose a new mechanism called *Structured Self-organizing Map* (SSOM) according to the idea of optimization idea for large scale system [14]. The basic idea is to decompose the original large dimensional problem into several sub low dimensional problems. Now the proposed SSOM can be described as below.

**SSOM Algorithm**

1. Initialize codebook set $C$ randomly. Usually choose them to be in the center of all the input data.
2. The SOM algorithm (7) is applied to obtain codebook structure.
3. When SOM is *Structured stable*, implement the average operation for each $\{R_i\}$ and obtain the desired codebooks.

We called this algorithm as *Structured Self-organizing Map* (SSOM). There are following advantage over LBG and other existing codebook construction methods based on SOM.

1. SSOM usually can reach global or sub-global optimum since the SOM in step 1 can comprehend the strucutral distribution of the required codebooks [13]. This overcomes the local optimum problem associated with LBG and its variants.
2. After *Structural stable* of the codebook distribution structure, average operation is used directly to create the desired codebook set. This overcomes the time-consuming issue associated with conventional SOM algorithms.

Theoretically, SSOM seems very good. The optimization technique for large scale system is used here [14]. First we consider the structure distribution of the codebooks. Once we find it *Structural stable*, the simple

average operation is taken instead of the long learning process associated with SOM.

However, there are several critical issues for practical implementations. First, during the training process, some codebooks will never be winner in the competitive iterations and hence never be updated. This is due to a fact that no input data are within the nearest distance with those codebooks. These codebooks are called *dead codebooks*. These dead codebooks will create large distortion error, In order to implement the SOM algorithm smoothly, one need to activate these dead codebooks for further competition in the learning process. The following algorithm is designed to deal with the dead codebooks, in which one move the dead codebooks to a region associated with largest distortion error. Next, we will describe the idea in detail.

Let us assume that a codebook $d_i$ is found to be dead during the learning process, then it will not be calculated in the future iterations. In this case, one can find a codebook $c_j$ associated with largest distortion error. Then we move the dead codebook $d_i$ to a neighbour of codebook $c_j$ and update both of them with the following algorithm.

$$\begin{aligned} w_i[k+1,q] &= w_j[k,q] - \alpha\delta_j^{[q]} \\ w_j[k+1,q] &= w_j[k,q] + \alpha\delta_j^{[q]} \\ w_i[k+1,v] &= w_j[k,v],\ \forall v \neq q,\ v = 1,2,\cdots,n \end{aligned} \quad (20)$$

where $\alpha$ is a small number $0 < \alpha < 1$. Denote $N_j$ is the number of input elements associated with codebook $c_j$ and its elements are $x_i \in R^n, \quad i = 1,2,\cdots N_j$. Further, $q$ and $\delta_j^{[q]}$ can be calculated with the following formula.

$$q = \left\{ l | \sum_{i=1}^{N_j} \left( x_i^{[l]} - c_j^{[l]} \right)^2 = \max_k \sum_{i=1}^{N_j} \left( x_i^{[k]} - c_j^{[k]} \right)^2 \right\}$$

$$k = 1,2,\cdots,n.$$

$$\delta_j^{[q]} = \sum_{i=1}^{N_j} \left( x_i^{[q]} - c_j^{[q]} \right)^2$$

where $q$ represents the direction with largest deviation in neighborhood of $c_j$. This algorithm will activate the dead codebooks quickly and speed up the the SOM algorithm. Moreover, this will make the SOM algorithm to grasp the codebook structure more fairely since the updated code book will share the distortion error with the codebook with the largest distortion error. This modification will be embeded in the step 2 in SSOM. In the sequal of this paper, the SOM with dead codebook update will be called *modified SOM*.

The other potential issue with SSOM is that we do not know when the modified SOM will be *Structural Stable*. This will hinder the applications for the modified SSOM in practice. Actually, the *Structural Stable* will be depend on the input data distribution. So, we will use the following algorithm in practice to replace SSOM.

**Modified SSOM**

1. Initialize codebook $C$ randomly. Usually choose them to be in the center of all the input data.
2. The Modified SOM algorithm is applied to construct codebookstructure
3. With obtained codebooks from step (ii), LBG is used directly and obtained the desired codebooks.

The modified SSOM can be programmed and implemented easily. The last step is motivated by the following facts.

1. The steps of (i) and (ii) will provide a better initial codebooks for LBG since SSOM will grasp the structure distribution of the codebooks. LBG with these codebooks will achieve better results.
   .
2. If the *modified SOM* in steps of (i) and (ii) stated in SSOM is *Structural Stable*, Then the LBG in step (iii) in *modified SSOM* will provide same result as in the step (iii) in SSOM.

These facts imply that modifed SSOM and SSOM will converge to same codebooks if the *modifed SOM* is *structural stable*. In practice, the *modified SSOM* will provide a sub-optimal solution for SSOM.

# 4 Simulation

In this section, some simulations have been implemented for the modified SSOM and comparisions have been made with different existing codebook construction methods. The experiment data used here is from [4], including the data outside unit square which is not included in the simulations in [4]. Total 500 two-dimensional data points are included in the simulations.

## 4.1 Comparisions with SOM

Here we considere ten different cases in which the codebook size are $(10,20,\cdots,100)$ respectively. In order to compare the results fairly, all the codebooks are initialized at the same mean value of all the input data. In each simulation we first implement the modified SOM

(two dead codebooks are found in the processes) and calculated the root mean square error(RMSE) over all data points. Then the modified SSOM is implemented. Here we perform the modified SOM 50000 epoches (including adjust the two dead codebooks), and 10000 epoches for the modified SSOM. The comparision results are presented in table 1.

| size | RMSE ± stddev (SOM) | RMSE ± stddev (SSOM) | Gain |
|---|---|---|---|
| 10 | 0.0820 ± 15% | 0.0434 ± 1% | 90% |
| 20 | 0.0397 ± 5% | 0.0328 ± 3% | 21% |
| 30 | 0.0358 ± 3% | 0.0273 ± 3% | 30% |
| 40 | 0.0320 ± 3% | 0.0235 ± 2% | 36% |
| 50 | 0.0302 ± 4% | 0.0205 ± 3% | 47% |
| 60 | 0.0293 ± 3% | 0.0186 ± 3% | 57% |
| 70 | 0.0284 ± 3% | 0.0169 ± 2% | 68% |
| 80 | 0.0271 ± 5% | 0.0159 ± 4% | 71% |
| 90 | 0.0263 ± 6% | 0.0147 ± 3% | 78% |
| 100 | 0.0259±4% | 0.0138 ± 2% | 87% |

Table 1. Comparison of SOM and Modified SSOM

The mean values of RMSE after implementing the modified SOM for 50000 epoches and that after implementing the modified SSOM for 10000 epoches are plotted in Figure 1. Their codebook distrbutions with twenty codebooks are plotted in Figure 2 and Figure 3 respectively.

1. From table 1, one can see that though we implement the modified SSOM much less iterations, the final results are much better due to the implementation of LBG. This fact indicates that the final stage in the proposed algorithm not only speed up the converegence rate but also improves the performance significantly.
2. From Figure 1, one can see that the mean values of RMSE by implementing the modified SOM 100000 epoches and 50000 epoches do not change so much. These values are reduced significantly via the modified SSOM. This indicates that the modified SSOM can produce much better results than the modified SOM.
3. From Figure 2 and Figure 3, one can see that the codebook distribution is different in the case of twenty codebooks. The codebook distribution obtained via the modified SSOM is more reasonable.

### 4.2 Comparisions with LBG and LBG-U

The LBU is a popular codebook construction approach [1] and it has been used widely in practice [2]. One critical problem is that it will be stucked at a local optimum. Recently, the LBG-U was proposed and overcome the local optimum issue. Simulations have proved that LBG-U is much better than LBG with mild additional computation costs [4]. Actually, LBG-U will use LBG repeatedly and its computation cost will grow significantly with the sizes of input data and codebook.

In this section, we will use the same input data set as in the previous sub-section. Also ten cases for different codebook size are considered for LBG, LBG-U and the modified SSOM. The coparision resutls are listed in table 2 below.

| size | RMSE ± stddev (LBG) | RMSE ± stddev (LBG-U) | RMSE ± stddev (SSOM) |
|---|---|---|---|
| 10 | 0.0642 ± 25% | 0.0453 ± 13% | 0.0434 ± 1% |
| 20 | 0.0367 ± 17% | 0.0322 ± 1% | 0.0328 ± 3% |
| 30 | 0.0302 ± 18% | 0.0265 ± 2% | 0.0273 ± 3% |
| 40 | 0.0252 ± 4% | 0.0224 ± 1% | 0.0235 ± 2% |
| 50 | 0.0224 ± 5% | 0.0198 ± 1% | 0.0205 ± 3% |
| 60 | 0.0199 ± 4% | 0.0177 ± 2% | 0.0186 ± 3% |
| 70 | 0.0184 ± 5% | 0.0160 ± 2% | 0.0169 ± 2% |
| 80 | 0.0168 ± 4% | 0.0147 ± 1% | 0.0159 ± 4% |
| 90 | 0.0156 ± 3% | 0.0135 ± 1% | 0.0147 ± 3% |
| 100 | 0.0145 ± 3% | 0.0125 ± 2% | 0.0138 ± 2% |

Table 2. Comparisons of LBG, LBG-U and SSOM

Also the mean values of RMSE for LBG, LBG-U [4] and the modified SSOM are illustrated graphically in Figure 4.

It can be seen from table 2 and Figure 4 that the modified SSOM can improve LBG significantly. It is much better than LBG-U in the case of ten codebooks. With more codebooks, the differences among LBG, LBG-U and SSOM are becoming smaller.

Also from the codebook distribution in Figure 3 for the modified SSOM and Figures in [4] for LBG-U, one can see that the codebook distribution structures for LBG-U and the modified SSOM are different. This indicates that either LBG-U or the modified SSOM is not global optimal and there is still possible to improve them. This will motivate us to investigate the convergence property further for the modified SSOM in the future.

## 5 Conclusions

In this paper, a new codebook construction method SSOM was proposed based on SOM neural network. This new algorithm can cope with any input data and proved to be much better than conventional LBG approach and SOM. First it used the SOM property to grasp the data structure distribution. This is very important and it can overcome local optimum issue as-

sociated with LBG. Also the SOM is modified in this paper to deal with the dead codebook issue in the learning process. Further, After the codebook is stable, the averaging operation was taken or LBG is applied due to a fact that the modified SOM with one codebook will converge to its mean value of input data. This operation overcomes the time-consuming shortcomings associated with all the neoural network algorithms. Simulation results showed the effectiveness of the modified SSOM.

Though the modified SSOM approach is very effective. It is still possible to improve it via adjusting the codebook locations from local region point of view. Since it may be hard for the modified SOM to be *Structural Stable* as defined in this paper. In broad sense, the *regional structural stable* will be more applicable and may bring better performance. This is under our current investigation.

## Références


[1] Gray. R. M., *Vector quantization , IEEE ASSP Magazine,* 1984, pp. 4-29.

[2] R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Press, 1992.

[3] Macqueen.J., *Some methods for classification and analysis of multivariate observations, Volume 1 of Proceedings of the Fifth Berkely Symposium on Mathematical statistics and probability* , Berkely, 1967.University of California Press. pp. 281-297

[4] Fritzke. B., *The LBG-U method for vector quantization-an improvement over LBG inspired from neural networks, Neural Processing Letter*, vol. 5, no. 1, 1997, pp. 35-45.

[5] Kohonen. T., *Self-organized formation of topologically correct feature maps , Biological Cybernetics*, vol.43, 1982, pp 59-69.

[6] Linde.y., Buzo. A. and Gray. R. M., *An algorithm for vector quantizer design, IEEE Transactions on Communication*, vol. 28, 1980, pp. 84-95.

[7] T. M. Martinetz and K. J. Schulten. *Topology representing networks. Neural Networks*, 7(3), 1994, pp. 512 :-522, 1994.

[8] Sun Y., *On Quantization error of self-organizing map network, Neurocomputing*, 34 (2000), pp. 169-193.

[9] Z. P. Lo, Y. Yu, B. Bavarian, *Analysis of the convergence properties of topology preserving neural networks, IEEE Trans. Neural Networks* 4 (March, 1993), pp. 207-220.

[10] B. Frizke, *A growing neural gas network learns topologies*, In G. Tesauro, D. S. Touretzky and T. K. Leen, ediotrs,*Advances in Neural Information Processing Systems* 7, pp. 625-632. MIT Press, Cambridge MA 1995.

[11] Hong Yan, *Quantum Classification Algorithms for Signal and Image Processing, 2000 International Workshop on Multimedia Data Storage, Retrial, Integration and Application*, 2000, pp. 75-80.

[12] Lin, J. K., D. G. Grier and J. D. Cowan, *Faithful representation of separable distributions, Neural Computation*, 1997. vol. 9, pp. 1035-1320.

[13] T. Kohonen, *Self-Organization and Associative Memory*, Springer Series in Information Sciences, Springer, New York, 1988.

[14] M. Jamshidi, *Large-Scale Systems : Modeling, Control and Fuzzy*, North-Holland Series in System Science and Engineering, North-Holland, New York, 1996.

[15] B. Fritzke. *Incremental learning of local linear mappings, In F. Fogelman and P. Gallinari, editors, ICANN'95 : International Conference on Artificial Neural Networks*, pages 217-222, Paris, France, 1995b. EC2 and Cie.

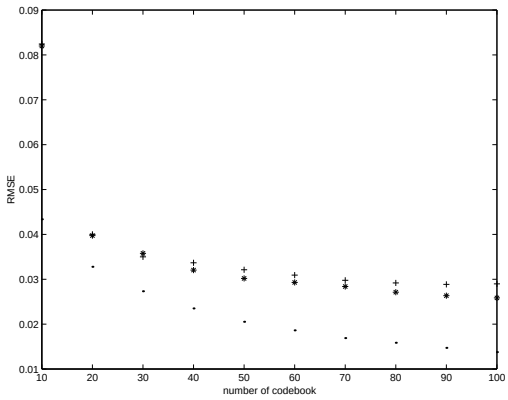


FIG. 1 – '+'SOM (10000 epoches) ;'*' SOM (50000 epoches) ;'.' SSOM (10000 epoches)

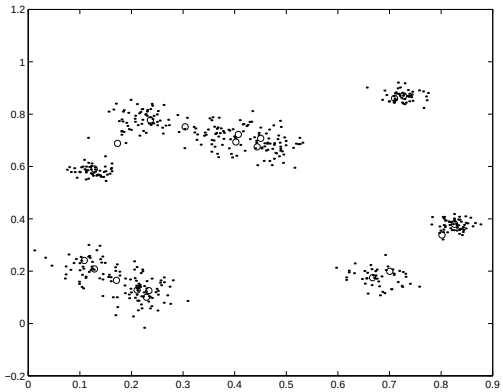

FIG. 2 – Data set and codebooks (SOM)

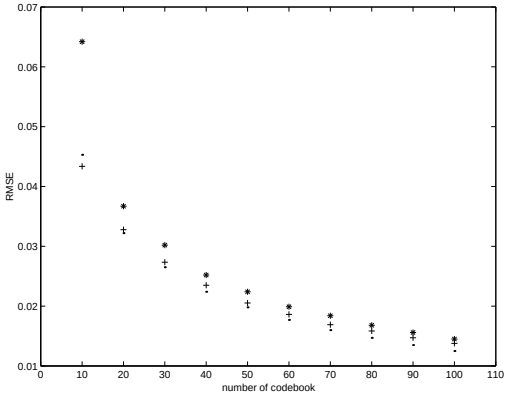


FIG. 4 – '*' LBG ;'.'LBG-U ; '+' SSOM

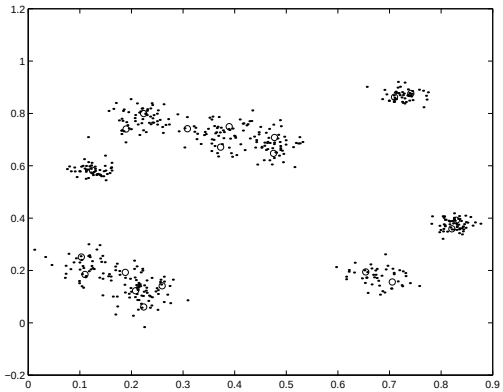

FIG. 3 – Data set and codebooks (SSOM)