This is the published version:


Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30044861

©2002 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Copyright: 2002, IEEE
Abstract In this paper, a two-stage algorithm for vector quantization will be proposed based on self-organizing map (SOM) neural network. Firstly, a conventional self-organizing map will be modified to deal with dead codebooks in the learning process and then will be used to obtain the codebook distribution structure for a given set of input data. Secondly, sub-blocks will be classified based on the previous structure distribution with a prior criteria. Then the conventional LBG algorithm will be applied to these sub-blocks for data classification with initial values obtained via SOM. Finally, extensive simulations illustrate that the proposed two-stage algorithm is very effective.

1 Introduction

Vector quantization (VQ) is an important technique for data compression [1, 2]. 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 \( \xi_1, \xi_2, \ldots, \xi_N \), choose a set \( C \) with \( |C| = M \) and \( M \ll N \), forming \( n \)-dimensional codebook vectors \( w_1, w_2, \ldots, w_M \). To transfer any data vector \( \xi_i \) to the receiver, the sender only needs to transfer the index \( j \) of the associated codebook \( w_j \), which is the nearest codebook to \( \xi_i \) according to a certain distance measure. Usually, the squared Euclidean distance is adopted.

\[
d(a, b) = \sum_{i=1}^{n} (a_i - b_i)^2 \quad \text{for } a, b \in \mathbb{R}^n
\]

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

\[
E(D, C) = \sum_{\xi \in D} d(\xi, w_k)
\]

where \( w_k \) stands for the codebook that the sample data \( \xi \) belongs to.

The 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 (2) for a given data set \( D \). For randomly given data set, the corresponding optimization problem is a complex 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 [6, 7, 8, 9]. Among these algorithms, self-organizing feature map (SOM) [5] plays an important role since it has the capability of grasping the codebook distribution structure for a given sample data set without prior knowledge. This feature will help us construct a possible global optimal codebook. In [7, 8], 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 [9, 12]. In [12], a new algorithm is proposed and proved to be effective in practice. This new algorithm combines SOM learning and LBG together. However the learning process and LBG are still very slow with large data compression. In order to improve the algorithm, a two-stage approach will be proposed in this paper. There are several advantages for this two-stage parallel VQ algorithm. Firstly, its initial codebook can be chosen randomly. Secondly, the local optimum issue associated with LBG will be overcome by modifying the SOM learning algorithm. Thirdly, The slow convergence issue will be overcome since sub-block calculation mechanism is introduced and these sub-block calculations can be implemented in parallel.
2 Preliminaries

The LBG classification algorithm is a well-known codebook construction method in the existing literature. Its main shortcoming is that the final codebook depends on the initial codebook selection and usually only a local optimal value can be reached. As pointed out in [12] that this can be overcome by using the SOM to obtain a better initial codebook. Next, we introduce SOM briefly.

SOM is developed by Kohonen [5]. One of its significant features is that it can grasp 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 unlabeled 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.

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

$$||x - w_{l[k]}|| = \min ||x - w_i[k]||$$ (3)

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

$$w_{i[k+1]} = w_{i[k]} + h_{l[i]}^{(k)}(x[k] - w_{i[k]})$$ (4)

where $h_{l[i]}^{(k)}$ is the neighborhood function which can be chosen as Gaussian function [11]

$$h_{l[i]}^{(k)} = \alpha^{(k)} \exp \left(-\frac{d^2(l, i)}{2\sigma^{(k)}} \right)$$ (5)

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 iteration increases, $\sigma^{(k)}$ tends to zero and the width of Kernel function tends to one. In practice, $\alpha^{(k)}$ can be chosen as

$$\alpha^{(k)} = \alpha^{(0)}(1 - \frac{k}{T})$$ (6)

where $T$ is the total iteration number.

3 Two-stage parallel Classification Approach

3.1 The modified Self-Organizing Map

Theoretically, SOM seems very good. However, there exist some critical issues in practical implementations. During the training process, one important issue is that 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 and they 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. Let us assume that a codebook $d_l$ is found to be dead during the learning process, then it will not be updated 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 $d_l$ to a neighbour of codebook $c_j$ and update both of them with the following algorithm.

$$w_{i[k+1], q} = w_{i[k], q} - \alpha \delta_{ij}$$

$$w_{j[k+1], q} = w_{j[k], q} + \alpha \delta_{ij}$$

$$w_{i[k+1], v} = w_{i[k], v}, \forall v \neq q, v = 1, 2, \ldots, n$$ (7)

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 $x_i \in \mathbb{R}^n, i = 1, 2, \ldots, N_j$. Further, $q$ and $\delta_{ij}$ can be calculated with the following formula.

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

$$k = 1, 2, \ldots, n.$$
modification will be embeded in the first stage in the two-stage parallel VQ.

3.2 The Two-stage Algorithm

As described previously, the LBG algorithm depends on the selection of initial codebook values. Its variants, which may achieve better performance, are usually limited by initial value selection or with much more computation cost. In order to overcome these shortcomings, we design a two-stage optimization structure. First, the modified SOM algorithm is applied to catch the global topological structure. Then the data space is divided into several sub-classes in term of cluster codebooks. Finally, the LBG algorithm is used in each sub-classes parallelly to increase the convergence speed.

The algorithm can be described as below.

Step 1. Initialization. One first uses the modified SOM to catch the structural topology of data set.

Step 2. Codebook clustering. One can cluster these two codebooks into several sub-blocks as described below. Given a threshold \( D \), one can construct a distance matrix in the following way. Firstly, one computes the Euclidean distance between each two codebooks to get a distance matrix with the dimension of the codebook set. Secondly, let’s find the codebooks with the minimal distance in the distance matrix. If the minimal distance is less than the threshold \( D \), one clusters these two codebooks into one new codebook which will be chosen as the center of these codebooks. Thirdly, one can construct a new distance matrix with lower dimension for the new codebook and the left codebooks. The previous procedure can be repeated until the minimal distance is larger or equal to the threshold \( D \). Finally, some sub-blocks can be obtained accordingly.

Step 3. With the LBG algorithm, one can do classification for each sub-blocks in parallel with the codebooks obtained via the SOM as the initial values.

In this algorithm, the SOM was first used to produce a better initial values for the LBG algorithm. This overcomes the shortcomings of the LBG algorithm. With large input data, the LBG algorithm is usually very slow. In order to cope with slow computation, we proposed a scheme to separate the input data into different sub-blocks based on the construction of distance matrix. This will increase the computation capability since each block can be done in parallel.

4 Illustrative Examples

In this section, some simulations have been implemented for the two-stage parallel algorithm and comparisons have been made with different existing codebook construction methods. The experiment data used here is from [4], Total 500 two-dimensional data points are included in the simulations. Here we consider ten different cases in which the codebook size are \((10, 20, \ldots, 100)\) respectively. The Root Mean Squared Error (RMSE) will be used for comparisons of the LBG, the LBG-U and the two-stage parallel algorithm. Here the experiment results for the LBG and the LBG-U are from [4]. The values of the mean RMSE for the LBG, the LBG-U and the two-stage parallel algorithm are illustrated in Figure 1.

We can see from Figure 1 that the two-stage parallel algorithm can achieve much better classification result compared to the LBG and nearly the same performance with the LBG-U. Here, the initial codebook is chosen randomly. Further, the two stage parallel algorithm can be implemented in parallel and this can reduce the computation load for each processor and increase the speed of classification.

If the number of the codebooks is twenty, Figure 2 shows codebook distribution after implementation of the SOM algorithm. Figure 3 shows the codebook distribution improvement. With the threshold \( D = 0.002 \), the codebook clustering result is shown in Figure 4.

5 Conclusions

In this paper, a new two-stage parallel classification algorithm was proposed based on the modified SOM algorithm. This new algorithm can cope with any input data and simulation results proved to be much better than the conventional LBG approach. Its initial codebook can be chosen randomly and the local optimum issue associated with the LBG approach can be overcome by the modified SOM. The convergence issue can also be improved significantly due to the introduction of a parallel sub-block computation mechanism. The parallel mechanism proposed in this paper will direct us for designing possible quantum VQ algorithm in the future.

Références

FIG. 1 - RMSE mean comparison, '+' for two-stage parallel VQ, '*' for LBG, '.' for LBG-U


FIG. 2 - Data set and codebooks trained by the modified SOM with 10000 epoches

FIG. 3 - Data set and codebooks trained by two-stage parallel algorithm

FIG. 4 - codebook clustering distribution with threshold $D = 0.002$