A Novel Fuzzy Reinforced Learning Strategy in Vector Quantisation

Zhen Ji, Member, IEEE, Taikang Yang, Lai Jiang, and Wenhuan Xu

Abstract—This paper presents a new approach toward the design of optimised codebooks by vector quantisation (VQ). A strategy of fuzzy k-means reinforced learning (FRL) is proposed which exploits the advantages offered by fuzzy clustering algorithms, competitive learning and knowledge of training vector and codevector configurations. Reinforced learning, which is consisted of attractive factor and repulsive factor, is used as a pre-process before using the conventional VQ algorithm, i.e. fuzzy k-means (FKM) algorithm. At each iteration of RL, codevectors move intelligently and intentionally toward an improved optimum codebook design. This is distinct from the standard FKM in which a random variation is introduced in the movement of the codevectors to escape from local minima. Experiments demonstrate that this results in a more effective representation of the training vectors by the codevectors and that the final codebook is nearer to the optimal solution in applications such as image compression. It has been found that the standard FKM yields improved quality of codebook design in this application when RL is used as a pre-process. The investigations have also indicated that new fuzzy k-means reinforced learning vector quantisation (FRLVQ) strategy is insensitive to the selection of both the initial codebook and a learning rate control parameter, which is the only additional parameter introduced by FRL from the standard FKM.

I. INTRODUCTION

VECTOR quantisation (VQ) techniques have been widely used as a powerful data compression technique for use in storage and transmission systems. The conventional VQ techniques that have been used in image compression and other applications may be classified into two broad categories based upon either a hard or a soft decision in a membership function. These categories are exemplified for hard decisions by the GLA [1] and for soft decisions by FKM [2] algorithm.

The GLA, or Linde, Buzo and Gray (LBG) algorithm [3], forms the basis of most practical vector quantiser designs. GLA uses a form of gradient descent method to reach a local minimum, in terms of the measure of distortion. GLA employs hard decision which means that each training or local minimum, in terms of the measure of distortion. GLA uses a form of gradient descent method to reach a local optimum codebook design. This investigation has indicated that new fuzzy k-means reinforced learning (FRL) strategy of fuzzy k-means reinforced learning (FRL) is proposed which exploits the advantages offered by fuzzy clustering algorithms, competitive learning algorithms and knowledge of training vector and codevector configurations. RL is used as a pre-process before an FKM iteration is performed. At each iteration of RL, each codevector is made to move in the most effective direction according to the present distributions of training vectors and codevectors. Within each iteration of RL, the size and the direction of the movement of each codevector is decided by the overall pairwise competition between the attraction of each training vector and the repulsion of its corresponding winning codevector, i.e. its nearest codevector. Experiment results have shown this new Fuzzy k-means Reinforced Learning (FRL) method is a more effective representation of the training vectors by the codevectors and that the final codebook will thus be nearer to the optimal solution.

II. VECTOR QUANTISATION

Let χ be a training vector set of size M and dimension L, i.e. χ = {x₁, x₂, ..., x_M}, x_i ∈ R^L, ∀i = 1, 2, ..., M, where R^L is an L-dimensional Euclidean space. Let Y be a codevector set of size N and dimension L, i.e. Y = {y₁, y₂, ..., y_N}, y_j ∈ R^L, ∀j = 1, 2, ..., N. A vector quantiser assigns each of the M training vectors to one of N clusters. Each cluster is represented by a codevector and, hence, each training vector is also represented by a codevector.

The quality of the codebook design is often measured using the average of the absolute distortion, i.e. mean square error (MSE) between training vectors and corresponding nearest codevectors, represented by D:

$$D = \frac{1}{M} \sum_{i=1}^{M} [d_{\text{min}}(x_i)]^2.$$ (1)

Here d_{\text{min}}(x_i) is defined as:

$$d_{\text{min}}(x_i) = \min_{y_j \in Y} d(x_i, y_j).$$

with Euclidean distance $d(x_i, y_j) = \| x_i - y_j \|.$

A. GLA: Hard Decision Membership Function

In each iteration of GLA, each training vector is assigned to a certain cluster based on the nearest neighbour condition.
which is defined by a membership function $\mu_j(x_i)$:

$$\mu_j(x_i) = \frac{1}{\sum_{j=1}^{M} \mu_j(x_i)}$$

(2)

$\mu_j(x_i)$ indicates the degree to which training vector $x_i$ belongs to codevector $y_j$, and for GLA this degree is either full membership or no membership. Then, the optimum codevectors for this membership function may be evaluated using the function:

$$y_j = \frac{\sum_{i=1}^{N} \mu_j(x_i)x_i}{\sum_{i=1}^{M} \mu_j(x_i)}$$

(3)

B. FKM: Soft Decision Membership Function

In each iteration of FKM, each training vector is assigned to all codevectors with different membership values between zero and one indicating the degree to which the training vector belongs. The most popular membership function is defined as [7]:

$$\mu_j(x_i) = \left[ \frac{1}{\sum_{j=1}^{N} \frac{d(x_i, y_j)}{d(x_i, y_j)}} \right]^{-1}$$

(4)

where $\lambda$ is a parameter that controls the "fuzziness" of the membership, normally $1 < \lambda < \infty$. This function has been chosen such that $\mu_j(x_i)$ is near unity if $y_j$ alone is very close to $x_i$ and is very small if $y_j$ is much more remote than the codevector that is closest to $x_i$. In this way, every training vector has a degree of membership of each of the $N$ codevectors. Increasing $\lambda$ "hardens" the partitioning of the training vectors by the membership function so that it approaches the abrupt condition 2 of GLA as $\lambda \rightarrow \infty$.

At an FKM update, every codevector moves to some extent toward the training vector $x_i$, but the nearest, or "winning", codevector moves by the largest amount. This may be contrasted with GLA, in which only the winning codevector moves towards $x_i$.

C. FRLVQ Strategy

Figures 1 and 2 illustrate a new learning process, which is named reinforced learning (RL) in this paper. The search for a globally optimum codebook in VQ may be considered to be the search for the global minimum of a cost function such as average distortion. In principle, this might be achieved by making a full search over all possible solutions but, in multi-dimensional space, such a search would be too extensive to be realized in practice. A strategy of introducing randomly generated perturbation in a learning process, to permit escape from local minima, is not guaranteed to attain globally optimum codebook design. RL approaches this problem by attempting to ensure that, at each iteration of the learning process, each codevector moves effectively to escape from local minima and move the solution toward the global optimum.

For RL, the movement of each codevector, $y_j$, is decided not only by the attractions of all the training vectors but also by the repulsions of their corresponding winning codevectors. The competition of these two forces is measured in pairs, which means the attraction of each $x_i$ is contemporary with the repulsion of its corresponding winning codevector $w_i$, i.e. the nearest codevector to $x_i$. The strengths of attraction and repulsion are defined by means of membership functions $\gamma_j(x_i)$ and $\eta_j(w_i)$ in equations (6) and (7), respectively. As is the case for the membership function of FKM, $\gamma_j(x_i)$ is inversely proportional to the Euclidean distance between $x_i$ and $y_j$ raised to the power $\lambda$; $\eta_j(w_i)$ has a similar relationship to the Euclidean distance between $w_i$ and $y_j$. Figure 1 illustrates these relationships for the winning codevector, $w_i$, and two other codevectors $y_1$ and $y_2$ near a training vector $x_i$. In Fig.1, because $d(x_1, y_1) < d(w_i, y_1)$, we have $\gamma_1 > \eta_1$; and because $d(x_1, y_2) > d(w_i, y_2)$ we have $\gamma_2 < \eta_2$. Figure 2 shows the corresponding components of movement of $y_1$ and $y_2$. Note that these components are either toward or away from the training vector: the components towards $x_i$ are due to the training vector itself and are shown in Fig.2(a); and the components away from $x_i$ are due to the presence of the winning codevector and are shown in Fig.2(b). Since $w_i$ is one of the codevectors we have $d(w_i, x_i) = 0$ and this implies an infinite degree of repulsion. However, it is intended that $w_i$ should be stationary during the RL learning phase, this is achieved by replacing $d(w_i, x_i)$ with $d(x_i, w_i)$ in equations (6) and (7) so that an exactly counterbalancing attraction is created from $x_i$ to $w_i$.

The combined effects of the attractions and repulsions are shown in Fig.3. Any codevector $y_j$ in Zone 1 moves toward $x_i$ because $d(x_i, y_j) < d(w_i, y_j)$, so that $\gamma_j(x_i) > \eta_j(w_i)$.

Fig. 1. Membership factors of codevectors $y_j(o)$ due to the training vector $x_i(x)$ and its corresponding winning codevector $w_i(•)$ within an RL iteration. Both of them are inversely proportional to their Euclidean distances.

Fig. 2. Movement components of codevectors $y_j(o)$ due to the training vector $x_i(x)$ and its corresponding winning codevector $w_i(•)$ within an RL iteration.
\[ y_j^{(v+1)} = y_j^{(v)} + \alpha^{(v)} \times \sum_{i=1}^{M} \gamma_j(x_i)(x_i - y_j^{(v)})\delta \]  \hspace{1cm} (5)

where \( \delta = \gamma_j(x_i) - \eta_j(w_i) / (\gamma_j(x_i) + \eta_j(w_i)) \) and \( \alpha^{(v)} = c/u \), \( c \) is a constant chosen to be a value between 50 and 100 for RL in the later experiments. Here, the attractor factor \( \gamma_j(x_i) \) and repulsive factor \( \eta_j(w_i) \) are defined as the membership functions of \( x_i \) and \( w_i \) for the codevector \( y_j \):

\[ \gamma_j(x_i) = \left\{ \sum_{p=1}^{N} \frac{d(x_i,y_j)}{d(x_i,y_p)} \right\}^{\lambda} + \sum_{p=1}^{j-1} \frac{d(x_i,y_j)}{d(w_i,y_p)} \]  \hspace{1cm} (6)

\[ \eta_j(w_i) = \left\{ \sum_{p=1}^{N} \frac{d(w_i,y_j)}{d(w_i,y_p)} \right\}^{\lambda} + \sum_{p=1}^{j-1} \frac{d(w_i,y_j)}{d(w_i,y_p)} \]  \hspace{1cm} (7)

If \( y_j = w_i \), equation (7) is modified as:

\[ \eta_j(w_i) = \left\{ \sum_{p=1}^{N} \frac{d(w_i,y_j)}{d(w_i,y_p)} \right\}^{\lambda} + \sum_{p=1}^{j-1} \frac{d(w_i,y_j)}{d(w_i,y_p)} \]  \hspace{1cm} (8)

In equation (5),

\[ \gamma_j(x_i) - \eta_j(w_i) \leq 1. \]  \hspace{1cm} (9)

This is called the RL factor (\( \xi \)). If all terms involving \( w_i \) are removed from (6) and (7) or (8) then \( \xi = 1 \) and RL becomes an FKM algorithm as a learning process.

An FKM iteration is performed after each RL process and before proceeding to the next. The combination of RL and FKM is named fuzzy reinforced learning vector quantisation (FRLVQ) in this paper. Each iteration of FRLVQ is the combination of one iteration of RL and one iteration of FKM, which are applied in that order. In practice the effects of \( w_i \) are found to diminish as learning progresses and the result of applying FRLVQ becomes similar to use of FKM. Since conventional VQ algorithms are much more economical than FRLVQ with regard to processing requirements it is then suggested that GLA or FKM should be used as a post-process to reach a final optimised codebook after a number of FRLVQ iterations.

The FRLVQ algorithm is summarised in Table I. Step 4 is sometimes needed to avoid over-spreading of codevectors, during an update, in the multi-dimensional Euclidean space caused by using a high value of \( c \) in an RL sub-iteration. RL sub-iteration and FKM sub-iteration are steps 2-4 and 5-6 respectively.

### III. Experimental Results and Discussions

The Lena image of size 256×256 was used as an experimental training vector set. The pixels of this image take values between 0 and 255. The training vectors were obtained by dividing the Lena image into 4096 blocks of size 4×4. Each block was rearranged to be a 16-dimensional vector, i.e., \( L = 16 \). Let \( X \) be the Lena image set, which contains 4096 (\( =M \)) vectors in \( R^{16} \). Let \( Y \) be the codebook, which contains 256 (=\( N \)) codevectors in \( R^{16} \). Then the compression rate was...
8 bits representing 16 pixels, i.e. 0.5 bits per pixel (bpp). The resulting images were evaluated by the peak signal to noise ratio (PSNR), which is defined as:

\[ \text{PSNR} = 10 \log_{10} \left( \frac{255^2}{\sum_{i=1}^{M} \left\| x_i - w_i \right\|^2} \right). \]  

(10)

Root mean square error per pixel (RMSE p.p.) gives a more immediate measure of error in the image compression problem and is sometimes quoted in the results. It is defined as:

\[ \text{RMSE} = (D/L)^{1/2}. \]  

(11)

The overall performance of the proposed algorithms in this paper has been compared with those of the GLA, FVQ and FKM algorithms. In these comparisons, their quality of codebook design was measured by PSNR, computational efficiency and robustness. The default values of the parameters for VQ algorithms are listed in Table II. Here, the learning rate control parameter \( \alpha^{(v)} \) is redefined as:

\[ \alpha^{(v)} = D^{(v)} e / v. \]

The learning rate should be increased if the distortion \( D^{(v)} \) between the training vector and nearest neighbour vector is large and vice versa. The codebook design process is terminated if the decrease of distortion, \( (D^{(v-1)} - D^{(v)}) / D^{(v-1)} \), is below a threshold \( \varepsilon \), where \( v \) is the index of the iterations.

All simulations in this paper were run on the system using code written in MATLAB. Since the MATLAB language was chosen for its convenience rather than speed of execution, the computation times reported in this paper should be regarded as relative time measures, and not the minimum values achievable.

A. **FRLVQ Performance**

The first set of experiments evaluated the performance of the VQ algorithms in relation to quality of codebook design and computational effort. The results are shown in Table III, where the same initial codebook was used with different VQ algorithms. In all cases, the first 256 training vectors were selected as the initial codebook. These results are achieved after the iterations satisfy \( (D^{(v-1)} - D^{(v)}) / D^{(v-1)} < \varepsilon \) except for the FRLVQ algorithm. It is clear, from Table III, that each iteration of FRLVQ is a very time-consuming, this is principally due to the computation of the membership functions (6) and (7). Against this, fewer iterations are needed to achieve an improved quality of performance if reinforced learning is used in FKM as a pre-process and the increased computational effort that is needed is then not severe. It is apparent that performance FRLVQ algorithms, i.e. computation time and PSNR, benefited from using reinforced learning as a pre-process. There was a 1.83dB improvement in PSNR from GLA to FRLVQ. FRLVQ achieved a 0.74dB improvement in PSNR from GLA to FRLVQ. FRLVQ achieved a 0.74dB improvement in PSNR from GLA to FRLVQ.

B. **FRLVQ Robustness**

The second set of experiments evaluated the robustness of the proposed algorithm with respect to the selection of initial codebook, sequence of learning rate factors and different images. To investigate the effect of the selection of the initial codebook upon the quality of codebook design, simulations were undertaken using codebooks that were initialised by selecting randomly from the set of training vectors. Table IV shows the summary of the simulation results with five initial codebooks, which were generated in this way. Comparing the results of Table IV with those of Table III, there was only 0.13dB difference in PSNR at largest for FRLVQ codebooks.

Further simulations demonstrated that RL is robust with regard to the choice of \( c \) in \( \alpha^{(v)} \) (see Table V). Values of \( c \) between 50 and 100 were found to give similar performance for all experiments in this paper. Thus, 75 was chosen as a

### Table II

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( M )</th>
<th>( L )</th>
<th>( k )</th>
<th>( \lambda )</th>
<th>( W )</th>
<th>( \alpha^{(v)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLA</td>
<td>4096</td>
<td>256</td>
<td>( 10^{-3} )</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FKM</td>
<td>4096</td>
<td>256</td>
<td>( 10^{-3} )</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FVQ[8]</td>
<td>4096</td>
<td>256</td>
<td>( 10^{-3} )</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FRLVQ</td>
<td>4096</td>
<td>256</td>
<td>( 10^{-3} )</td>
<td>10</td>
<td>3</td>
<td>0.1v</td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( v )</th>
<th>( D^{(v)} )</th>
<th>RMSE p.p.</th>
<th>PSNR(dB)</th>
<th>( T ) (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLA</td>
<td>21</td>
<td>1323</td>
<td>28.96</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>FKM</td>
<td>49</td>
<td>1028</td>
<td>30.05</td>
<td>330</td>
<td></td>
</tr>
<tr>
<td>FVQ[8]</td>
<td>21</td>
<td>1101</td>
<td>29.75</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>FRLVQ</td>
<td>17</td>
<td>867</td>
<td>30.79</td>
<td>147</td>
<td></td>
</tr>
</tbody>
</table>

### Table IV

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( v )</th>
<th>( D^{(v)} )</th>
<th>RMSE p.p.</th>
<th>PSNR(dB)</th>
<th>( T ) (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLA</td>
<td>12-15</td>
<td>1050-1085</td>
<td>30.03-30.25</td>
<td>220</td>
<td></td>
</tr>
<tr>
<td>FKM</td>
<td>29-36</td>
<td>982-1034</td>
<td>30.60-30.73</td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>FVQ[8]</td>
<td>18-32</td>
<td>879-905</td>
<td>30.60-30.73</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>FRLVQ</td>
<td>18-32</td>
<td>879-905</td>
<td>30.60-30.73</td>
<td>170</td>
<td></td>
</tr>
</tbody>
</table>

### Table V

<table>
<thead>
<tr>
<th>( c ) in ( \alpha^{(v)} )</th>
<th>( v )</th>
<th>( D^{(v)} )</th>
<th>PSNR(dB)</th>
<th>( T ) (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>3+30</td>
<td>970</td>
<td>30.31</td>
<td>159</td>
</tr>
<tr>
<td>50</td>
<td>3+19</td>
<td>872</td>
<td>30.77</td>
<td>153</td>
</tr>
<tr>
<td>FRLVQ</td>
<td>75</td>
<td>3+14</td>
<td>867</td>
<td>30.79</td>
</tr>
<tr>
<td>100</td>
<td>3+21</td>
<td>883</td>
<td>30.71</td>
<td>156</td>
</tr>
<tr>
<td>125</td>
<td>3+20</td>
<td>904</td>
<td>30.61</td>
<td>156</td>
</tr>
</tbody>
</table>

suitable value for all codebook designs. However, it has been found that the best choice of $c$ does vary with the number of codevectors and the size of training vector set.

IV. CONCLUSIONS

This paper has described a new vector quantiser design method, reinforced learning (RL), which is recommended for use as a pre-process before applying one of the standard VQ training algorithms such as FKM. The purpose of RL is, then, to achieve an intelligent initial spread of the codevectors throughout the input vector space and thereby avoid the tight and inefficient clustering of the codevectors that may be encountered when using other training methods. It does this most effectively if its use is alternated with one of the standard algorithms; when alternated with FKM it is designated FRLVQ.

The behaviour of FRLVQ has been carefully tested by generating codebooks to represent the Lena image. The testing has revealed that consistently superior results are obtained if FRLVQ is used for as few as two or three iterations prior to final training with a standard method. Since FRLVQ is significantly more demanding of processing requirements than other methods it is sensible to use the smallest number of FRLVQ iterations needed to yield best optimisation.

In addition to improvement of final codebook quality, use of RL as a pre-process in the standard FKM has the benefit of effectively eliminating the sensitivity of the final solution upon choice of initial codebook. Another attractive feature of FRLVQ is that it has only one adjustable control parameter, the learning rate control parameter, and it has been found that this may be set anywhere within a very wide range of values without compromising the quality of performance.

The standard FKM and FRLVQ both yield higher quality codebook designs than GLA, but detailed consideration of the final codebooks obtained by these methods has shown that the improvements are obtained in different ways: the standard FKM introducing randomly generated perturbation in a learning process, to permit escape from local minima, is not guaranteed to attain globally optimum codebook design and is very time consuming. RL approaches this problem by attempting to ensure that, at each iteration of the learning process, the movement of each codevector is decided not only by the attractions of all the training vectors but also the repulsions of their corresponding winning codevectors. Therefore, each codevector moves in the most effective direction to escape from local minima and move the solution toward the global optimum with saving time.

ACKNOWLEDGMENT

This work was partially supported by the National Natural Science Foundation of China under grant no. 60572100, Royal Society (U.K.) International Joint Projects 2006/R3 - Cost Share with NSFC under grant no. 60711130233, Foundation of State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications, CHINA) and Science Foundation of Shenzhen City under grant no. 200408.

REFERENCES