REFERENCES


IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 5, NO. 6, JUNE 1996

Order Statistics Learning Vector Quantizer
I. Pitas, C. Kotropoulos, N. Nikolaidis, R. Yang, and M. Gabbouj

Abstract—In this correspondence, we propose a novel class of learning vector quantizers (LVQ’s) based on multivariate data ordering principles. A special case of the novel LVQ class is the median LVQ, which uses either the marginal median or the vector median as a multivariate estimator of location. The performance of the proposed marginal median LVQ in color image quantization is demonstrated by experiments.

I. INTRODUCTION

Research in neural networks (NN) [1], [2] is a rapidly expanding area that has attracted the attention of scientists and engineers throughout the last decade. A large variety of artificial neural networks has been developed based on a multitude of learning techniques and having different topologies [2]. One prominent example of neural networks is the learning vector quantizer (LVQ). It is an autosegmental nearest-neighbor classifier that classifies arbitrary patterns into classes using an error correction encoding procedure related to competitive learning [1]. In order to make a distinction between the (standard) LVQ algorithm and the proposed variants that are based on multivariate order statistics, the LVQ algorithm will be called linear LVQ algorithm hereafter.

Let us assume a sequence of vector-valued observations \( x(n) = (x_1(n), \ldots, x_p(n)) \) where \( n \) denotes discrete-time index and \( p \) denotes the dimensionality of vector-valued observations. Let \( \{w_i(n); i = 1, 2, \ldots, \ K\} \) be a set of variable \( p \times 1 \) reference vectors that are randomly initialized. Competitive learning tries to find the best-matching reference vector \( w_i(n) \) to \( x(n) \) (i.e., the winner) where \( c = \arg \min_i ||x - w_i|| \) with \( || \cdot || \) denoting the Euclidean distance between any two vectors. In the linear LVQ, the weight vectors are updated as blocks concentrated around the winner using the recursive relations [4] as follows:

\[
\begin{align*}
\hat{w}_i(n+1) &= \hat{w}_i(n) + \alpha(n) [x(n) - \hat{w}_i(n)] \\
\end{align*}
\]

where \( \alpha(n) \) is the adaptation step sequence and \( \mathcal{N}_c(n) \) denotes a neighborhood set around the winner. Equation (1) implements an unsupervised learning procedure. In order to obtain optimal global ordering, \( \mathcal{N}_c(n) \) has to be wide enough initially and to shrink monotonically with time [1], [4]. Variants of LVQ implementing supervised learning have also been proposed [4]. Several choices of the adaptation step sequence \( \alpha(n) \) (called schedules) are possible [5]. The recall procedure of LVQ is used to determine the class \( C_g \) represented by \( w_g \) with which the vector of input observations is
most closely associated, i.e.,
\[
x(n) \in C_p \text{ if } \|x - \bar{w}_p\| = \min_i \{\|x - \bar{w}_i\|\}
\] (2)

where \(\bar{w}_i\) denotes the weight vector of the \(i\)th neuron after the convergence of the learning procedure.

When the learning procedure reaches equilibrium, it results in a partition of the domain of input vector-valued observations called Voronoi tessellation [6]. This means that the input space is partitioned into regions (called Voronoi neighborhoods) bordered by hyperplanes, such that each region contains a reference vector that is the nearest neighbor to any input vector within it. Furthermore, the reference vector of a Voronoi neighborhood is the centroid, e.g., the sample arithmetic mean of all input vectors belonging to that neighborhood. From this point of view, LVQ can be classified as a method that belongs to the class of \(K\)-means algorithms [7].

It can easily be seen that the reference vector for each class \(i = 1, \ldots, K\) at time \(n + 1\) is a linear combination of the input vectors \(x(j)\) \(j = 0, \ldots, n\) that have been assigned to class \(i\). Moreover, it can be shown that only in the special case of one data class, which has a multivariate Gaussian distribution and for the adaptation step sequence \(\alpha(n) = 1/(n + 1)\), the winner vector is the maximum-likelihood estimator of location (i.e., the arithmetic mean of the observations that have been assigned to the class). Neither in the case of multiple classes that are normally distributed nor in the case of non-Gaussian multivariate data distributions does the linear LVQ yield the optimal estimator of the cluster means. In general, linear LVQ and its variations suffer from the following drawbacks: 1) they do not use optimal estimators for obtaining the reference vectors \(w_i, i = 1, \ldots, K\) that match the maximum likelihood estimators of location for the pdf \(f_i(x)\) of each class \(i = 1, \ldots, K\); and 2) they do not have robustness against the outliers that may exist in the sample observations, since it is well known that linear estimators have poor robustness properties [8].

In order to overcome these problems, we propose a novel class of LVQ’s that are based on order statistics (e.g., the median) which have very good robustness properties [8], [9]. In the case of LVQ’s, we should rely on multivariate order statistics [10]. In this novel class, each cluster is represented by its median, which is continuously updated after each training pattern presentation.

Based on multivariate data ordering principles, we introduce the marginal median LVQ (MMLVQ), the vector median LVQ (VMLVQ) and the marginal weighted median (MWMVQ) in Section II. The performance of the proposed MMLVQ in color image quantization is studied in Section III. Conclusions are drawn in Section IV.

II. LEARNING VECTOR QUANTIZERS

BASED ON MULTIVARIATE DATA ORDERING

The notion of data ordering cannot be extended in a straightforward way in the case of multivariate data. There is no unambiguous, universally agreeable total ordering of \(N\) \(p\)-variate samples \(x_1, \ldots, x_N\), where \(x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T, i = 1, \ldots, N\). The following so-called subordering principles are discussed in [9] and [10]: marginal ordering, reduced (aggregate) ordering, partial ordering, and conditional (sequential) ordering. In marginal ordering, the multivariate samples are ordered along each one of the \(p\)-dimensions as follows:

\[
x_{j(1)} \leq x_{j(2)} \leq \cdots \leq x_{j(N)} \quad j = 1, \ldots, p
\]
i.e., the sorting is performed in each channel of the multichannel signal independently. The \(i\)th marginal order statistic is the vector \(x_{(i)} = (x_{1(i)}, x_{2(i)}, \ldots, x_{p(i)})^T\). Accordingly, the marginal median is the vector \(x_{\text{med}}\) defined by

\[
x_{\text{med}} = \begin{cases} 
(x_{1(p+1)}, \ldots, x_{p(p+1)})^T & \text{for } N = 2r + 1 \\
\frac{x_{1(p)} + x_{1(p+1)}}{2}, \ldots, \frac{x_{2(p)} + x_{2(p+1)}}{2}, \frac{x_{2(p+1)}}{2} & \text{for } N = 2r.
\end{cases}
\] (3)

It can be used in the following way in order to define the marginal median LVQ. Let us denote by \(X_i(n)\) the set of the vector-valued observations that have been assigned to class \(i\), \(i = 1, \ldots, K\) until time \(n + 1\). We find at time \(n\) the winner vector \(w_i(n)\) that minimizes \(\|x(n) - w_i(n)\|, i = 1, \ldots, K\). The marginal median LVQ (MMLVQ) updates the winner reference vector as follows:

\[
w_i(n + 1) = \text{median } \{x(n) \cup X_i(n)\}.
\] (4)

The median operator is given by (3). Thus, all past class assignment sets \(X_i(n), i = 1, \ldots, K\) are needed for MMLVQ, MMLVQ requires the calculation of the median of data sets of ever increasing size, as can be seen from (4). This may pose severe computational problems for relatively large \(n\). However, for integer-valued data, a modification of the running median algorithm proposed by Huang et al. [13] can be devised to facilitate greatly median calculations by exploiting the fact that the marginal median of the already assigned samples \(X_i(n)\) is known. This algorithm leads to very large computational savings. It must be noted that, although MMLVQ employs the entire past data set for the calculation of the new weight vectors, the algorithm does not require the storage of the past data samples. Only the storage of the marginal histograms for each class is needed. The asymptotic properties of MMLVQ have been studied in [14]. It has been proven that MMLVQ outperforms the (linear) LVQ with respect to the bias in estimating the true cluster means both for a contaminated Gaussian data model as well as for a contaminated Laplacian data model. As far as the mean squared estimation error is concerned, it has been proven that MMLVQ outperforms the (linear) LVQ in the case of a contaminated Laplacian data model.

Another definition of the multichannel median is based on R-ordering principles. It is the so-called vector median defined in [12]. In R-ordering, the various data \(x_i\) are ordered according to their distances from a predefined point. That is, multivariate ordering is reduced to 1-D ordering. The vector median is defined as the vector \(x_{\text{med}}\) that minimizes the \(L_1\) error norm \(d(x_i, x_{\text{med}}) = \sum_{i=1}^{N} |x_i - x_{\text{med}}|\) under the condition that it belongs to the set \(\{x_i, i = 1, \ldots, N\}\). In other words,

\[
\sum_{i=1}^{N} |x_i - x_{\text{med}}| = \sum_{i=1}^{N} |x_i - x_j|, \quad j = 1, \ldots, N.
\] (5)

The vector median LVQ (VMLVQ) uses the following formula to update the winner vector \(w_i(n)\) at step \(n\):

\[
w_i(n + 1) = \text{vector median } \{x(n) \cup X_i(n)\}.
\] (6)

The vector median operator in the previous expression is the one defined in (5). Other distance measures (e.g., Mahalanobis distance) can be used in (5) instead of the \(L_1\) norm as well.

Another possible extension results by employing the weighted median (WM) filter [11]. The marginal weighted median LVQ (MWM-LVQ) can be defined as follows. Let us denote by \(w_i(n) = \)
\[(w_{i1}(n), w_{i2}(n), \ldots, w_{ip}(n))^T\] the winner vector, i.e., \(c = i\). In MWMVQ, the elements of the winner vector are updated as follows:

\[w_{ij}(n+1) = \text{median}\{C_{i0} \odot x_j(n), \ldots, C_{in} \odot x_j(0)\}\]  \hspace{1cm} (7)

where \((C_{i0}, C_{i1}, \ldots, C_{in})^T\) is the vector of the duplication coefficients for the \(i\)th class. The duplication coefficients can be chosen in such a way that they weigh heavily the desired section of the observation data (i.e., the new observations or the old ones). If a weight \(C_{ij}\) is zero, this means that the corresponding sample \(x(n-i)\) has not been assigned to the \(i\)th class.

III. Simulations

In practice, 8 bits are used for representing each of the R, G, B components in color images. That is, 24 bits are needed for each pixel in total. Color image quantization aims at reducing the number of RGB triplets (which are \(2^{24}\) at most) to a predefined number (e.g., 16 up to 256) of codevectors so that the image can be displayed in limited palette displays. Several algorithms have been proposed recently [15]–[18]. In the following, we shall focus on the performance of the marginal median LVQ. A set of experiments have been conducted in order to assess its performance in color image quantization and to compare it to one of the well-known vector quantization (VQ) methods such as the Linde–Buzo–Gray (LBG) algorithm [19], [20] and the linear LVQ. Moreover, we aim at studying the robustness of the codebooks (i.e., color palettes) determined by the above-mentioned VQ techniques. To this end, we have also included noisy color images as inputs to the learning phase.
Let us first define when we declare that the learning procedure in the VQ techniques included in our study has converged. During the learning phase, each VQ algorithm is applied to the training set several times. Each presentation of the training set is called training session hereafter. At the end of each training session \( k \), the mean squared error (MSE) between the quantized and the original training patterns (i.e., RGB triplets) is evaluated as follows:

\[
D(k) = \frac{1}{\text{card}(S)} \sum_{(i,j) \in S} ||x(i,j) - \hat{x}^{(k)}(i,j)||^2
\]

where \( S \) denotes the training set, \( \text{card}(S) \) stands for the cardinality of the training set, \( x(i,j) = (x_R(i,j), x_G(i,j), x_B(i,j))^T \) represents the original training pattern, and \( \hat{x}(i,j) \) is the quantized pattern. The training patterns can be obtained from the input color image, e.g., by subsampling. We decide that the learning procedure has converged if

\[
\left| \frac{D(k - 1) - D(k)}{D(k)} \right| \leq \rho
\]

where \( \rho \) is a small number, e.g., \( \rho = 10^{-4} \). In LBG or in linear LVQ using the adaptation step sequence

\[
\alpha(n) = \frac{1}{n+1}
\]

\( D(k) \) is a monotonically decreasing function, so termination rule (9) is well suited. This is not always the case with the MMLVQ or the linear LVQ using the linear schedule

\[
\alpha(n) = \lambda \left( 1 - \frac{n}{M} \right), \quad 0 < \lambda \leq 1
\]
TABLE I
PERFORMANCE OF LBG, LINEAR LVQ AND MARGINAL MEDIAN LVQ IN COLOR IMAGE QUANTIZATION FOR SEVERAL CODEBOOK SIZES

(a) Learning phase

<table>
<thead>
<tr>
<th>Codebook Size</th>
<th>LBG</th>
<th>linear LVQ</th>
<th>linear LVQ</th>
<th>MMLVQ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iter.</td>
<td>MSE</td>
<td>Iter.</td>
<td>MSE</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>297.905</td>
<td>16</td>
<td>296.116</td>
</tr>
<tr>
<td>32</td>
<td>18</td>
<td>150.78</td>
<td>24</td>
<td>163.03</td>
</tr>
<tr>
<td>64</td>
<td>29</td>
<td>96.147</td>
<td>21</td>
<td>94.55</td>
</tr>
<tr>
<td>128</td>
<td>18</td>
<td>59.138</td>
<td>37</td>
<td>58.52</td>
</tr>
<tr>
<td>256</td>
<td>17</td>
<td>35.40</td>
<td>19</td>
<td>34.98</td>
</tr>
</tbody>
</table>

(b) Recall phase

<table>
<thead>
<tr>
<th>Codebook Size</th>
<th>LBG</th>
<th>linear LVQ</th>
<th>linear LVQ</th>
<th>MMLVQ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>297.114</td>
<td>296.116</td>
<td>298.55</td>
<td>300.95</td>
</tr>
<tr>
<td>32</td>
<td>168.88</td>
<td>172.80</td>
<td>171.109</td>
<td>175.27</td>
</tr>
<tr>
<td>64</td>
<td>105.844</td>
<td>105.106</td>
<td>105.74</td>
<td>111.16</td>
</tr>
<tr>
<td>128</td>
<td>69.137</td>
<td>68.84</td>
<td>68.07</td>
<td>71.22</td>
</tr>
<tr>
<td>256</td>
<td>46.11</td>
<td>46.46</td>
<td>45.85</td>
<td>48.30</td>
</tr>
</tbody>
</table>

where $M$ is an arbitrary large number. Consequently, in the latter algorithms, the termination rule (9) should take the form $|D(k - 1) - D(k)|/D(k) \leq \rho$. In addition, the reference vectors determined by these algorithms are the ones for which $D(k)$ attains its minimum value. In our comparative study, we have used as figures of merit (i) the MSE between the quantized patterns and the original ones at the end of the learning phase; (ii) the MSE at the end of the recall phase that is our ultimate goal; and (iii) the number of training sessions needed for convergence. The MSE at the end of the recall phase is defined similarly to (8) i.e.,

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \| x(i,j) - \hat{x}(i,j) \|^2$$

where $N$ is the number of image rows/columns.

Let us describe the selection of parameters in the different VQ techniques. All learning procedures in our study are initialized by the splitting technique used for initializing the LBG algorithm [19]. The choice of the adaptation step sequence in the linear LVQ is the most crucial for its performance. If LVQ is properly initialized and a right choice for the adaptation step sequence is made, then the experimental results indicate that LVQ will attain a performance similar to the one of LBG. Both the adaptation step sequences (10) and (11) have been considered. The adaptation step sequence (10) does not pose any further problem. For the linear schedule (11) for small codebook sizes, e.g., 16, 32, we set $\lambda = 0.003$ and $M = 10^2$. For codebook sizes 64 to 256 we have chosen $0.03 \leq \lambda \leq 0.08$ and $M = 10^2$. In the case of MMLVQ, we have to choose an adequate odd number of samples to be assigned to each class in order to initialize the median calculation. This number was usually given a small value e.g., 5, 9, etc. Its selection does not influence the performance of the algorithm, which always converges in a similar way. That is, the MSE achieved at the end of the learning phase or the number of the training sessions is not significantly affected. Furthermore, in cases where the running algorithm proposed by Huang et al. [13] is applicable (e.g., in image processing), each training session of the MMLVQ requires less computation time than the one of LBG/LVQ because the learning procedure of MMLVQ does not involve any floating-point arithmetic. This is not the case with LBG and linear LVQ that require floating point operations. Having defined the parameters involved in the VQ techniques under study, we proceed to the description of the experiments.

Fig. 1(a) shows the original Pepper image of dimensions $256 \times 256$ with 24 b/pixel. A training set of 4096 RGB triplets has been created by subsampling the original image by a factor of four both horizontally and vertically. The performance of the LBG, the linear LVQ with adaptation step sequences (10) and (11), and the marginal median LVQ in the learning phase is listed in Table I(a). In the same table, we have included the number of iterations required for the convergence of the learning procedure of each VQ technique. For MMLVQ, two numbers are given for each entry because the training session for which the minimum is found does not coincide with the last one. The first number denotes the training session where the minimum is found and the second one the training session for which the modified termination rule employing absolute values is satisfied. The MSE at the end of the recall phase is tabulated in Table I(b). From Table I(a) and (b) it is seen that all VQ techniques attain an approximately identical performance with respect to the MSE at the end of the learning and recall phase. However, it is worth noting that MMLVQ yields close to optimal reference vectors in fewer iterations ($\approx 10$) compared to other VQ techniques. For example, when the learning phases of LVQ and MMLVQ algorithms with 256 output neurons (last row in Table I(a)) were running on a Silicon Graphics Indy Workstation, the execution times were 37.95 s (LVQ with adaptation step (10)), 91.22 s (LVQ using adaptation step (11)) and 28.47 s (MMLVQ), respectively.

Next, we have considered the case of a noisy training set. Fig. 1(b) shows the original color image corrupted by adding mixed white zero-mean Gaussian noise having standard deviation $\sigma = 20$ and impulsive noise with probability of impulses (both positive and negative ones) $p = 5\%$ independently to each R,G,B component. A training set of 4096 noisy training patterns has been created by subsampling the noisy image of Fig. 1(b) by a factor of four both in the horizontal and vertical direction. The reference vectors determined by the learning procedure of the LBG, the linear LVQ and the MMLVQ on the noisy training set have been applied subsequently to the original image of Fig. 1(a) in order to reduce the number of RGB triplets to a
in color image quantization. The results demonstrate the superior performance of MMLVQ in comparison with the performance of classical LQ and LBG, especially in the case of noisy data. This fact, along with its smaller execution time, makes MMLVQ a good competitor to the classical LQ algorithm.

REFERENCES


