Vector Quantization by Local and Hierarchical LMSER *

Lei Xu

1. Dept. of Computer Science, The Chinese Univ of Hong Kong
2. National Machine Perception Lab, Peking Univ, Beijing

Abstract

Via extending Least MSE Reconstruction(LMSER) unsupervised learning to *local* and *hierarchical* forms, a new *Vector Quantization* (VQ) scheme is proposed which includes the conventional VQ scheme as a degraded case. Algorithms for both local and hierarchical implementation of this new VQ scheme have been designed. Experiments on coding the standard LENA image have shown that the new scheme provides a promising new VQ tool with several advantages.

1 Introduction

In 1991, the present author proposed the Least MSE Reconstruction(LMSER) unsupervised learning for neural networks (Xu, 1991; 1993). Recently, it was applied to the problem of signal separation by Karhunen & Joutsensalo (1994) and shown that it improves the performance of Oja's nonlinear constrained Hebbian learning (Oja et al, 1991) for the problem. This paper further extends LMSER learning into the *local* and *hierarchical* forms, which provides a new VQ scheme that includes the conventional VQ scheme as a degraded case. VQ plays important roles in speech and image coding (Rabbani& Jones, 1991). The key idea of the conventional VQ is using the center vector m_k to represent a set S_k of data. The key idea of our new scheme is that a direction vector w_k, in addition to m_k, is also used togather to form a line segment for representing S_k.

In sequel, secion 2 proposes the new VQ scheme and several local LMSER implementation algorithms. Section 3 gives a simple binary tree implementation algorithm and its experimental results. Section 4 discusses the new VQ scheme's advantages.

2 Local LMSER for VQ

A set of points can be represented by a line segment passing a center vector m_k along a direction specified by another vector w_k such that the following error is minimized.

$$E_k = \sum_{\mathrm{x}\in S_k} \|\mathrm{x}' - \mathrm{u}\|^2 = \sum_{\mathrm{x}\in S_k} \|\mathrm{x} - \mathrm{m}_k - \mathrm{w}_k s(\mathrm{w}_k^t[\mathrm{x} - \mathrm{m}_k])\|^2, \qquad (1)$$

*Correspondence address: Dept. of Computer Science, The Chinese Univ of Hong Kong, Shatin, NT, Hong Kong. Supported by the HK RGC Earmarked Grant CUHK250/94E.

where $u = w_k z, z = s(y), y = w_k^t x', x' = x - m_k$. $s(y)$ is a nonlinear differentiable antisymmetrical function with $s(0) = 0$, $s(-y) = -s(y)$ and $\lim_{y \to \infty} \frac{s(y)}{y} = constant \geq 0$, e.g., linear $s(y) = y$ or sigmoid $s(y) = tanh(\beta y)$. Actually, Eq.(1) is just the special case of one layer LMSER learning rule proposed by Xu (1991,1993).

With this presentation for data set, we can propose a new VQ scheme as follows:

Given a set of vectors $X = \{x_i\}$, we find a set of codebooks $MW = \{(m_j, w_j)\}_1^{N_c}$ to represent X such that each $x \in X$ is assigned to one of the N_c subsets according to

$$S_k = \{x : e_k < e_j,\ j \neq k\}, e_j = \|x - m_j - w_j s(w_j^t[x - m_j])\|^2, \tag{2}$$

The set of codebooks is decided by minimizing the MSE error:

$$E_{MW} = \sum_{k=1}^{N_c} E_{mw}^{(k)}, \quad E_{mw}^{(k)} = \sum_{x \in S_k} \|x - m_k - w_k s(w_k^t[x - m_k])\|^2. \tag{3}$$

Discarding w_k's (i.e. letting $w_k = 0$), the new scheme will degrade into exactly the conventional VQ, i.e., the new scheme includes the conventional VO as a special case.

The new scheme can be implemented by a competitive learning algorithm, e.g., the *Rival Penalized Competitive Learning (RPCL)* (Xu, Krzyzak & Oja, 1993), which results in an algorithm:

$$\theta_k^{new} = \theta_k^{old} + \Delta\theta_k, \quad \Delta\theta_k = \begin{cases} -\alpha_c \frac{\partial e_k}{\partial \theta_k}, & \text{if } k = c,\ \gamma_c e_c = \min_j \gamma_j e_j \\ \alpha_r \frac{\partial e_k}{\partial \theta_k}, & \text{if } k = r,\ \gamma_r e_r = \min_{j \neq c} \gamma_j e_j, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Where $\theta_k = [m_k, w_k]^t$, and the parameters α_c, α_r, γ_j's are the same as given in Xu, Krzyzak & Oja (1993).

For the special case of linear $s(y) = y$, eq.(3) becomes

$$E_{MW} = \sum_{k=1}^{N_c} E_{mw}^{(k)}, \quad E_{mw}^{(k)} = \sum_{x \in S_k} \|(x - m_k)(I - w_k w_k^t)\|^2. \tag{5}$$

The minimum of $E_{mw}^{(k)}$ can be reached at $m_k = m_k^* = \frac{1}{\# S_k} \sum_{x \in S_k} x$ and $w_k = w_k^*$ which is the principal component vector of S_k, that is, it is the eigenvector of $\Sigma_k = \frac{1}{\# S_k} \sum_{x \in S_k} (x - m_k^*)(x - m_k^*)^T$ corresponding to the largest eigenvalue.

Suppose X is initially divided into exclusive subsets $\{S_k\}_1^{N_c}$, the following iterative algorithm gives a code set MW such that E_{MW} reaches a local minimum:

Step 1 Take $x \in X$, assume it is currently in S_i, we check the possible changes $\Delta_i = -\Delta E_{MW}^{(i)}$ and $\Delta_j = \Delta E_{MW}^{(j)}$, $j \neq i$ that may result in if we remove it from S_i to S_j for $j \neq i$. The changes can be calculated according to eq.(5) via the new centers $m_i^{new} = \frac{\# S_i m_i^{old} + x}{\# S_i - 1}$, $m_j^{new} = \frac{\# S_j m_j^{old} + x}{\# S_j + 1}$, and the new prinicpal vectors w_i^{new}, w_j^{new} of $S_i - \{x\}$ and $S_j \cup \{x\}$.

Step 2 Find S_r with $\Delta_r = \min_j \Delta_j$, put x into S_r. Then, goto Step 1. The iteration stops untill no change for all $x \in X$.

When $w_k = 0$, the algorithm degenerates into the conventional *hard* K-mean algorithm and thus has the similar disadvantages encoutered by the later one that discussed in Xu & Jordan (1993). One way to improve it is to modify eq.(3) into

$$E_{MW} = \sum_{k=1}^{N_c} \sum_{x \in X} \omega_k(x)\{\|x - m_k - w_k s(w_k^t[x - m_k])\|^2 + \ln \omega_k(x)\}, \quad s.t. \sum_{k=1}^{N_c} \omega_k(x) = 1. \tag{6}$$

It is minimized by an alternative descent iterative algorithm:
Step 1 Fix all the m_k, w_k's, and to update all the $\omega_k(x)$'s by

$$\omega_k(x) = \frac{e^{-\|x-m_k-w_k s(w_k^t[x-m_k])\|^2}}{\sum_{j=1}^{N_c} e^{-\|x-m_j-w_j s(w_j^t[x-m_j])\|^2}} \tag{7}$$

Step 2 Fix all the $\omega_k(x)$'s, and update all the m_k, w_k's by one step move of gradient descent on the first term given in eq.(6).

The algorithm will converge to a local minimim of E_{MW} as long as Step 2 ensures decreasing or non-increasing of E_{MW}, which is always possible by a very small moving at Step 2. For the special linear case $s(y) = y$, eq.(6) becomes

$$E_{MW} = \frac{1}{\#X} \sum_{k=1}^{N_c} \sum_{x \in X} \omega_k(x)\{\|(x - m_k)(I - w_k w_k^t)\|^2 + \ln \omega_k(x)\}, \tag{8}$$

and Step 2 can be done by directly solving $\frac{\partial E_{MW}}{\partial w_k} = 0$, and $\frac{\partial E_{MW}}{\partial m_k} = 0$, that is, we have

Step 2' $m_k^{new} = \frac{1}{\#X} \sum_{x \in X} \omega_k(x)x$, and w_k^{new} is the solution of $\Sigma_k^{new} w_k = \lambda_{max} w_k$, i.e., the eigenvector of $\Sigma_k^{new} = \frac{1}{\#X} \sum_{x \in X} \omega_k(x)(x - m_k)(x - m_k)^T$ with the largest eigenvalue λ_{max}.

This modified algorithm is guarantted to converge sine Step 2' ensures the decreasing or non-increasing of E_{MW}. This algorithm is a special case of the *Multi-sets Modeling Learning* (Xu, 1994a) with each set being a line passing m_k and directing w_k. It is also a kind of *soft* local PCA (Xu, 1994b&c).

3 Hierarchical Implementation and Image Coding

In the following, we propose a sequential binary tree algorithm for the hierarchical implementation of the new VQ scheme Eqs.(2)(3), as shown in Fig.1, a node n_S in the binary tree corresponds to a set S of data. We compute its mean $m = \frac{1}{\#S} \sum_{x \in S} x$ and find an w that minimizes $e(w, m) = \sum_{x \in S} \|x - m_k - w_k s(w_k^t[x - m_k])\|^2$. When $s(y)$ is nonlinear, this can be done by adaptive rule

$w(t+1) = w(t) + \alpha\{s(y)(x - m - s(y)w(t)) + s'(y)(y - \|w(t)\|^2 s(y))(x - m)\}$,
which is actually the special case of the one layer LMSER rule given in Xu (1991; 1993). The converged $w(t)$ is used as the solution w.

When $s(y)$ is linear with $s(y) = y$, w can be obtained by explicitly solving

$$\Sigma w = \lambda_{max} w, \quad \Sigma = \frac{1}{\#S_k} \sum_{x \in S_k} (x - m)(x - m)^T. \tag{9}$$

After obtaining m, w and $e(w, m)$ as above, we check whether $e(w, m)$ is below a prespecified bound e_0. If yes, no son will be created for n_S, otherwise we create its two sons n_{S_1} and n_{S_2} which correspond to two exclusive subsets of :

$$S_1 = \{x : (x - m)^T w \geq 0, x \in S\}, \quad S_2 = \{x : (x - m)^T w < 0, x \in S\}. \tag{10}$$

Then, the same procedure will be repeated on S_1 and S_2, as well as their sons sequently (if any). The root of this tree corresponds to the original data set X. The tree construction will be completed when it is a balanced binary tree with a given depth d or when no node needs to be splitten under the given error bound e_0.

The binary tree T_b can be built in either the *Depth First* or the *Breadth First* , as well as *Best Frist* search (Pearl, 1984). In sequel, we use the binary tree for image coding.

The general procedure of VQ based image coding consists of two phases. At the first phase, a set of training images are transformed, one block by one block (e.g., with size $r \times r$, $r = 4$ or 8), into vectors. An VQ algorithm is used on the training vectors to obtain a set of N_c codevectors, which are also sent to the receiver. At the second phase, each $r \times r$ block on a testing image is turned into a vector $\mathbf{x}$ which is then coded by $[log_2 N_c]$ bits. The bits are transmitted to the receiver to find one codevector to decode the bits to a vector $\hat{\mathbf{x}}$.

In our case, the codebook T_b is sent to the receiver in advance with the parameter pair $\{\mathbf{m}, \mathbf{w}\}$ of each leaf as a codevector. At the sender, a test vector $\mathbf{x}$ is assigned to a leaf of T_b indexed by a binary number I. Hence, $\mathbf{x}$ can be coded by the index value I. if T_b is of depth d with leaves $N_f \leq 2^{d-1}$, the index value I for each leaf can be coded by $[log_2 N_f]$ bits or d bits, which are transmitted to locate the leaf codevector $(\mathbf{m}_k, \mathbf{w}_k)$ of T_b at the receiver. This part is same as that for the conventional VQ. However, here we have one additional job: we still need to code the scalar y so that $\mathbf{x}$ can be re-built at the receiver end by

$$\hat{\mathbf{x}} = \mathbf{m}_k + \mathbf{w}_k y, \quad y = s(\mathbf{w}_k^t[\mathbf{x} - \mathbf{m}_k]) \; or \; y = \mathbf{w}_k^t[\mathbf{x} - \mathbf{m}_k]. \tag{11}$$

The best way to code y is to quantize it into n_y numbers $\{b_j\}_1^{n_y}$ with each being a center of n_y bins so that y falls in each bin with equal probability, i.e., let $y = b_j$ if y falls in the bin centered at b_j. Thus, the set $\{b_j\}_1^{n_y}$ should be appended to a codevector for being sent to the receiver in advance. Also, for a test vector x, the extra $[log_2 n_y]$ bits need to be transmitted so that $\hat{\mathbf{x}}$ can be built by eq.(11).

One special case deserving a particular mention is that $n_y = 1$. In this case, all the y is quantized to the only center $b_1 = E(y)$. Now, Eq.(11) is simplifed into $\hat{\mathbf{x}} = \mathbf{m}_k$, and it returns back to implement the conventional VQ scheme. However, this algorithm is different from the existing algorithms in VQ literature. The parameters on $\mathbf{w}$ are still necessary for building up the binary tree T_b and for getting the final codevectors on $\mathbf{m}_k$ at all the leaf nodes.

From the perspective of the scalar quantization for coding y, we get also an interesting insight on the role of nonlinear $s(.)$. After quantizing y into n_y equal-probability-bins represented by $b_1 < b_2 < \cdots, < b_{n_y}$, in fact, $y = \mathbf{w}_k^t[\mathbf{x} - \mathbf{m}_k]$ becomes

$$y \doteq S(\mathbf{w}_k^t[\mathbf{x} - \mathbf{m}_k]), \quad S(r) = b_1 + 2\sum_{i=2}^{n_y} c_i U(r - c_i), \quad c_i = (b_i - b_{i-1})/2, \tag{12}$$

where $U(r - c) = 1$, when $r \geq c$, otherwise $U(r - c) = 0$. $S(r)$ is a stair function as shown in Fig.1(b). When y is bounded and $n_y \to \infty$, $S(r)$ tends to a sigmoid nonlinear function. Let $S_n(r) = b_{n_y} - b_1 + 2S(r)/(b_{n_y} - b_1)$, $-1 \leq S_n(r) \leq 1$, it falls within the same range as $s(r) = tanh(r)$. We can regard that the role of sigmoid nonlinear $s(.)$ is to help equal-probabilistic quantization of $s(\mathbf{w}_k^t[\mathbf{x} - \mathbf{m}_k])$. Furthermore, let $s(r) = S_n(r)$, $\mathbf{w}_k^* = \sqrt{0.5(b_{n_y} - b_1)}\mathbf{w}_k$ and $\mathbf{m}_k^* = \mathbf{m}_k$ to be the solutions for the case of nonlinear $s(.)$, we can observe that the procedure of letting $\mathbf{m}_k = \frac{1}{\#S}\sum_{\mathbf{x} \in S} \mathbf{x}$, and $\mathbf{w}_k$ being given by Eq.(9) for the linear case $s(y) = y$ actually is equivalent to a nonlinearr case if we divide $\{y : y = \mathbf{w}_k^t[\mathbf{x} - \mathbf{m}_k], \; \mathbf{x} \in S\}$ into n_y equal-probability-bins centered at $b_1 < b_2 < \cdots, < b_{n_y}$.

Fig.2 gives some experimental results. It is interesting that only one image is used as the training image and the image is significantly different from the testing LENA image and an other testing image Fig.2(d). It is instructive to compare these reconstructed images with those reconstructed images obtained by I/RTVQ provided in Chapter 12 of Rabbani& Jones(1991),which outperforms the classical VQ and other variants. There, eight 512×512 different images were used as the training set with each 4×4 block as an 16 dimensional vector. Roughly Figs.(b)&(c) are comparable to the results of I/RTVQ that used the bite rate 0.5 *bpp*, 0.69 *bpp* respectively. However, Figs.(b)&(c) have used the bit rates with a further compression by 20% - 30%. The Fig. 2(d) shows a good result obtained on a different testing image. These results suggest that the new scheme has superior abilities on both generalization and data compression.

4 Remarks

One key feature of the new scheme is that number of free variables in the whole codebook is significantly less than that of the convectional VQ scheme, where a code vector of dimension N_d represents only one point in R^{N_d}. Its rate of points per variable is $1/N_d$. However, for the new scheme, a pair $(\mathbf{w}_k, \mathbf{m}_k)$ with scalar qantization n_y levels can represent n_y points in R^{N_d}. The rate of points per variable is $\frac{n_y}{2N_d+n_y}$ which is larger than $1/N_d$ as long as $n_y > 2/(1-1/N_d), N_d > 1$. In other words, to use the same number of points in R^{N_d} as codevectors, the conventional VQ scheme needs $\frac{n_y N_d}{2N_d+n_y}$ times more free variables.

A codebook actually sets up an image model. The more the number of free variables in this model, the smaller the error on a training set, but the worse its generalization ability and the larger the error on a testing set, and the more the number of training samples are needed as well. In other words, the new scheme will have the advantages of better generalization and smaller testing error over the conventional VQ scheme. This is why better results are obtained in Figs.(b)-(e) by using only one training image and even smaller bit rates.

The key feature also means that the conventional VQ scheme needs $\frac{n_y N_d}{2N_d+n_y}$ times more bits than the new scheme need for coding a codebook of the same representation ability. For many practical applications such as visible telephone, a fixed codebook that obtained on a pre-determined training set is not adequate to produce a good coding for on line transmission. A considerably reduced size of codebook by the new scheme makes it possible to adaptively update the codebook and send it to the receiver for each certain period. Furthermore, the binary tree algorithm provides a fast training which makes the adaptive modification of codebook more possible.

In addition, the tree structure of the codebook T_b makes the coding and decoding very fast, which will help to considerably speed up the data transmission a lot too.

References

Karhunen, J & Joutsensalo, J.J. (1994), *Neural Networks*, Vol.7, pp113-127.
Oja. E., et al.(1991), Proc. ICANN'91, Finland, June 1991, pp. 385-390.
Pearl J. (1984), *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley.
Rabbani, M., & Jones, P.W.(1991), *Digital Image compression Techniques*, SPIE Press.
Xu, L(1991), In *Proc. IJCNN-Singapore-91*, Nov. 1991, pp. 2362-2373.
Xu, L(1993), *Neural Networks*, vol. 6, pp. 627-648.
Xu, L., Krzyzak, A. & Oja, E.(1993), *IEEE Tr. Neural Networks*, Vol.4, No.4, pp636-649.
Xu, L & Jordan, M.I.(1993),) *Proc. of WCNN'93*, Portland, OR, Vol. II, 431-434.
Xu, L(1994a), *Prof. of 1994 IEEE ICNN*, Invited paper, Vol.II, pp1252-1257.
Xu, L(1994b), *Prof. of 1994 IEEE ICNN*, Invited paper, Vol.I, pp315-320.
Xu, L(1994c), *Proc. 1994 ICONIP*, Invited paper, Oct 17-20, Seuol, Korea, pp943-949.

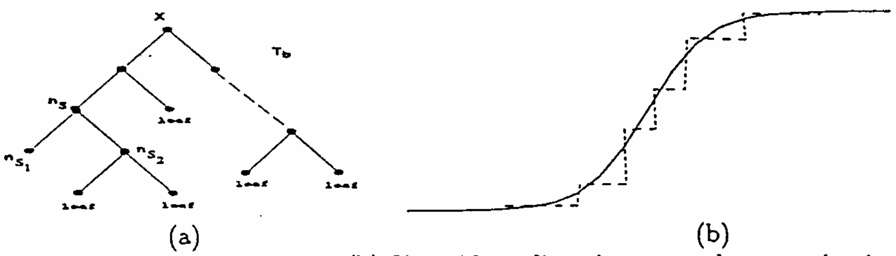

Figure 1 (a) A binary tree. (b) Sigmoid nonlinearity vs. scalar quantization.

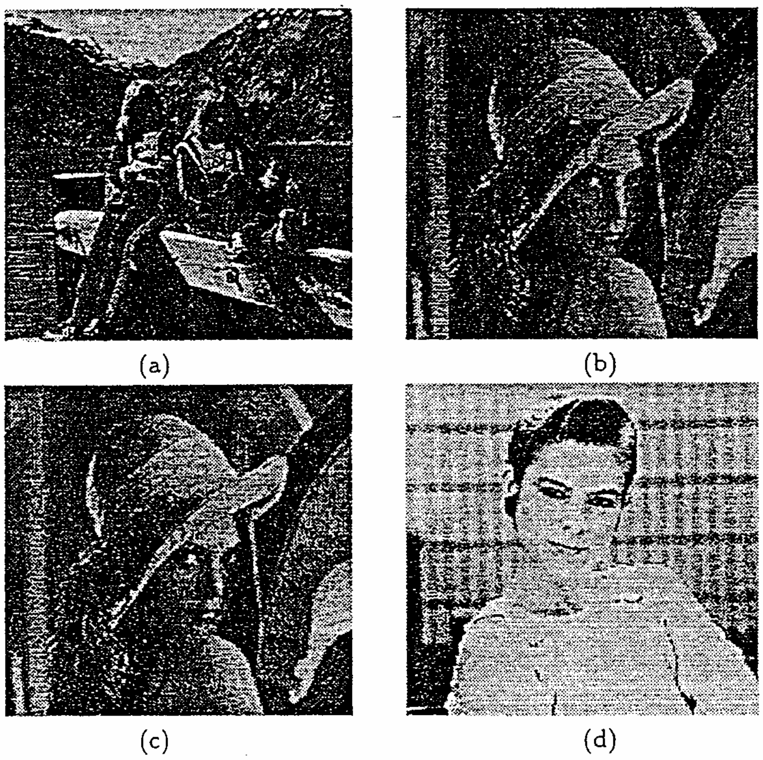

Figure 2 The experimental results on using the binary tree coding with $s(y) = y$ and $n_y = 4$. (a) The training image of size 512 × 512 with the bit rate 8 *bpp* (i.e., 256 grey levels). Each 4 × 4 blocks is represented by an 16 dimensional vector. (b) The reconstructed 512 × 512 LENA test image by a tree codebook of depth 5. Its bit rate is $(5 + \log_2 4)/16 = 0.4375$ *bpp*. (c) The residual image of Fig.2(b) scaled by a factor of 2 and biased by 127. (d) The reconstructed LENA image by T_b of depth 6 with the bit rate is 0.5 *bpp*. (e) The residual image of Fig.2(c). (f) The reconstructed another testing image of size 512 × 512.