Codebooks for vector bit-plane encoding

L. Lovisolo and E.A.B. da Silva

In many vector quantisation applications, codebooks must be uniformly distributed over a hypersphere. Shells of regular lattices are often used, but they provide only a limited choice in terms of the number of vectors K and dimension N. The authors present a method for generating such codebooks with arbitrary K and N.

Introduction: Bit plane coding represents the state of the art in image compression schemes. It is used in the recently developed JPEG-2000 standard. In [1] the bit-plane concept is generalised to vectors using the so-called successive approximation vector quantisation (SA-VQ) technique. In SA-VQ, a vector X is approximated by a vector X_m as

$$X_m = \sum_{n=1}^{m} \alpha^n Y_{i_n} \quad \text{where } Y_{i_n} \in C_N = \{Y_1, Y_2, ..., Y_K\} \quad \text{and } 0 < \alpha < 1 \tag{1}$$

where C_N is a codebook composed of unit-norm vectors, K is the cardinality of the codebook and m is the number of steps used in the quantisation.

Such a representation is useful only if, as we increase the number of vectors m used to approximate X, the error in the approximation $e_m = \|X - X_m\|$ tends to zero. It has been shown in [1] that a sufficient condition for this is

$$\begin{cases} \alpha \geq [2\cos\Theta(C_N)]^{-1} & \text{for } \Theta(C_N) \leq \frac{\pi}{4} \\ \alpha \geq \sin\Theta(C_N) & \text{for } \Theta(C_N) \geq \frac{\pi}{4} \end{cases} \tag{2}$$

where $\Theta(C_N)$ is the maximum angle between any vector in $\mathbb{R}^N$ and its nearest neighbour in C_N.

In [1] it is shown that the error $e_m = \|X - X_m\|$ in the representation is bounded by $e_m \leq B\alpha^m$. From eqn. 2, the smaller the $\Theta(C_N)$ the smaller the α that grants the convergence of the algorithm. Therefore, in order to minimise the representation error, we should minimise $\Theta(C_N)$. One way to achieve this while keeping the number of vectors K and the dimension N fixed, and therefore the rate $R = (m\log_2 K)/N$, is to distribute the vectors of the codebook as uniformly as possible on the surface of an N-dimensional hypersphere. This is equivalent to optimum covering over a hypersphere [2, 3]. For a broader discussion of the meaning of uniformity in this case, see [4]. In summary, the problem of designing 'good' codebooks for SA-VQ, and thus efficient vector bit-plane decompositions, is equivalent to distributing uniformly K points over an N-dimensional hypersphere. This is a problem that has been studied for centuries, but for which there is no general solution [4].

For the purpose of distributing an arbitrary number of vectors uniformly on a hypersphere, we could envision three methodologies: a) the algebraic methodology [2, 3], where we use orientation codebooks formed from lattices that are solutions for the packing and covering problems. These were employed in [1] in the context of wavelet image coding using vector bit-planes. However, there is no solution for the general problem of distributing K points over an N-dimensional hypersphere. Hardin and Sloane [3] have found solutions for dimensions 3, 4 and 5; b) the geometric approach [4], where we distribute the points over the surface of the sphere using geometric considerations. In [4], good asymptotic approximations for dimension 3 were found, but there is no general method valid for higher dimensions; c) a stochastic method in which we use an LBG-like algorithm [5] in order to generate the desired codebook. The training set used is composed of vectors whose components are i.i.d. Gaussian random variables. Since in this case their probability density function depends only on their magnitude, by normalising the vectors we end up with a training set whose p.d.f is uniformly distributed over the surface of a hypersphere. This approach is in theory able to generate good codebooks for any dimension N. However, the size of the training set has to be reasonably large even for moderate values of the dimension N and number of vectors K to be allocated, which results in high computational demands.

In this Letter, we propose a method that borrows characteristics from the geometric and stochastic methods. We design the initial codebook and training set using a novel geometric method, valid for an arbitrary dimension N, and employ the LBG algorithm to improve the initial codebook.

Distributing points over hypersphere: We first introduce the proposed geometric procedure (GM), based primarily on heuristic arguments and approximations that are asymptotically valid. The GM is employed to generate initial codebooks (IC) and training sets (TS) that are used in an LBG-like algorithm.

We consider first the representation in spherical coordinates of a point in $\mathbb{R}^N$ over an N-dimensional hypersphere of unit radius. The point is given by $\{1, \omega_1, \omega_2, ..., \omega_{N-1}\}$, where $0 \leq \omega_i \leq \pi$ for $i \in \{1, ..., N-2\}$ and $0 \leq \omega_{N-1} \leq 2\pi$.

The main supposition in the GM is that, when the number of vectors K is large, they define a tiling of the N-dimensional hypersphere by identical hypercubes of dimension $N-1$. Considering that each hypercube of size δ is defined by small angle variations $\Delta\omega_1, \Delta\omega_2, ..., \Delta\omega_{N-1}$, we have

$$\delta = \Delta\omega_1 \text{ and } \delta = \Delta\omega_j \prod_{i=1}^{j-1} \sin\omega_i \quad \text{for } j = 2, ..., N-1 \tag{3}$$

Given δ and a point $\omega_1, ..., \omega_{N-1}$, the values of $\Delta\omega_1, \Delta\omega_2, ..., \Delta\omega_{N-1}$ follow from eqn. 3. From the above, an algorithm for allocating the points is as follows:

compute $\Delta\omega_1$ from eqn. 3
for $\omega_1 = \frac{\Delta\omega_1}{2}$ to π in increments of $\Delta\omega_1$ do
  compute $\Delta\omega_1$ from eqn. 3
  for $\omega_2 = \frac{\Delta\omega_2}{2}$ to π in increments of $\Delta\omega_2$ do
    ⋱
    compute $\Delta\omega_{N-1}$ from eqn. 3
    for $\omega_{N-1} = \frac{\Delta\omega_{N-1}}{2}$ to 2π
      in increments of $\Delta\omega_{N-1}$ do
      for $i = 1, ..., N-1$ do
        {compute $x_i = \sin\omega_1 ... \sin\omega_{i-1} \sin\omega_i$
      compute $x_N = \sin\omega_1 ... \sin\omega_{N-2} \cos\omega_{N-1}$

The value of δ is computed by considering that if the K hypercubes, each one of area δ_{N-1}, provide a tiling of the hypersphere, then $A_N = K\delta^{N-1}$. The area A_N of the N-dimensional hypersphere of unit radius is given by $A_N = (N\pi^{N/2})/((N/2)!)$ for N even and $A_N = (N2^N\pi^{(N-1)/2}(((N-1)/2)!))/N!$ for N odd [2].

Since eqn. 3 and the tiling are only asymptotically valid, the value of δ provided by the tiling may give rise to a number of points $K' \neq K$. This can be corrected by making $\delta = \delta'(K'/K)^{1/N-1}$ and applying the procedure iteratively until $K' = K$.

Since GM generates asymptotically valid approximations (large K), it is good enough to generate a training set (TS) for the LBG algorithm. This circumvents the problem of the very large TS required by the stochastic approach. We have also verified that it generates good initial codebooks (IC). In our experiments, the distortion metric used was the mean squared error. Since the centroid of a region over a hypersphere does not lie on it, in the LBG algorithm we had to scale the magnitude of the centroid in every iteration to keep it on the surface of the hypersphere.

In order to assess the effectiveness of GM to generate both the TS and the IC, we evaluated the performance of codebooks generated by the LBG algorithm for four combinations of TS and IC. They are referenced as: **MM** – initial codebook (IC) and training set (TS) by the geometric method (GM); **II** – IC and TS i.i.d. Gaussian (IIDG); **IM** – IC IIDG and TS by GM; **MI** – IC by GM and TS IIDG. We compared their performance in terms of the number of iterations for the LBG algorithm as well as the maximum, average and minimum angles between the two nearest vectors in the codebook. Two points were observed: (i) the **MM** type generates more uniform codebooks, with the smallest maximum and highest average angle; for example, for $N = 3$ and $K = 12$ ((3, 12) codebooks) and training set composed of 9600 vectors, the

maximum, average and minimum angles are respectively: 63.28°, 63.07°, 62.64° for **MM**; 61.79°, 60.33°, 58.09° for **MI**; 63.29°, 63.01°, 62.82° for **IM** and 62.07°, 60.36°, 58.16° for **II**; (ii) the **MM** type converges faster to the final codebook. This can be observed by the number of iterations required for the LBG to converge: the **II** type needs 114% more iterations than the **MM** type, the **IM** 20% more, and the **MI** 68% more. The **MM** type is therefore clearly the best choice, i.e. the geometric method was effective in generating the TS and IC. It should be noted that the choice of the TS is more critical to the quality of the final codebook, while the IC influences the speed of convergence. We also observed that as the size of the TS increases, the performance of the generated codebooks increases asymptotically. For example for (4, 24) codebooks, a TS 5000 times larger than the IC generates a codebook with performance very close to that in [3].

Table 1: Mean squared error for different codebooks, for unit variance input

(N, K)	Generation	Steps	α	MSE
(3, 12)	packing	9	0.59	0.000052
(3, 12)	method	9	0.58	**0.000039**
(3, 12)	packing	6	0.59	0.001331
(3, 12)	covering	6	0.58	0.001094
(3, 12)	method	6	0.58	**0.001094**
(3, 15)	packing	6	0.57	0.000954
(3, 15)	covering	6	0.56	0.000797
(3, 15)	method	6	0.56	**0.000797**
(3, 21)	packing	6	0.55	0.000561
(3, 21)	covering	6	0.55	0.000562
(3, 21)	method	6	0.55	**0.000564**
(4, 24)	packing	6	0.59	0.001490
(4, 24)	method	6	0.60	**0.001625**
(4, 50)	packing	6	0.56	0.000671
(4, 50)	method	6	0.50	**0.000667**
(5, 64)	packing	6	0.59	0.001326
(5, 64)	method	6	0.59	**0.001311**
(5, 100)	packing	6	0.57	0.000867
(5, 100)	method	6	0.57	**0.000833**

Performance of generated codebooks: We evaluated the generated codebooks using the SA-VQ algorithm. The input data was composed of vectors statistically uniformly distributed over the surface of a hypersphere. Each codebook was evaluated by the average distortion after m steps. We generated codebooks in dimensions 3, 4 and 5 and compared the results obtained with the codebooks from [3]. Table 1 lists the results for codebooks that are the best-known solutions for the covering and packing problems [3] for various K and $N \in \{3, 4, 5\}$. Together we list the results of the codebooks generated by the **MM** method for the same (N, K). The mean squared error (MSE) shown for a unit variance input is for the value of α for which it is minimum. We quantised the input data using SA-VQ for each pair (N, K). Our method consistently generates codebooks with good performance.

We also tested the generated codebooks using the successive approximation wavelet vector quantisation (SA-W-VQ) algorithm [1] for the Lena image at 0.5 bit/pixel for different values of α. In this case, we obtained approximately the same performance, within a few hundredths of dB in terms of the peak signal-to-noise ratio (PSNR), which is also an indication of the effectiveness of the proposed method.

Conclusions: A method to allocate an arbitrary number of points almost uniformly over the surface of a hypersphere has been proposed. In cases where there are known solutions to the packing and covering problems, we have seen that the proposed method generates codebooks as good as those. The proposed method is therefore suited to generating codebooks for use in any vector bit-plane coding scheme, of arbitrary dimension and cardinality.

5 April 2000
Electronics Letters Online No: 20000737
DOI: 10.1049/el:20000737

L. Lovisolo and E.A.B. da Silva (*PEE/COPPE/DEL/EE/Universidade Federal do Rio de Janeiro, Cx. P. 68504, Rio de Janeiro, RJ, 21945-970, Brazil*)

References

1 CRAIZER, M., DA SILVA, E.A.B., and RAMOS, E.G.: 'Convergent algorithms for sucessive approximation vector quantization and applications to wavelet image compression', *IEE Proc. Vis., Image Signal Process.*, 1999, **146**, (3), pp. 159–164
2 CONWAY, J., and SLOANE, N.: 'Sphere packings lattices and groups, A series of comprehensive studies in mathematics' (Springer Verlag, New York, New York 10010, USA, 1993), 2nd edn.
3 SLOANE, N.J.A., HARDIN, R.H., and SMITH, W.D.: 'Neil J. A. Sloane: Home Page'. http://www.research.att.com/~njas/, February 2000 (and subpages on covering and packing)
4 SAFF, E.B., and KUIJLAARS, A.B.J.: 'Distributing many points on a sphere', *Mathematical Intelligencer*, 1997, **19**, (1), pp. 5–11
5 GERSHO, A., and GRAY, R.M.: 'Vector quantization and signal compression, Engineering and Computer Science' (Kluwer Academic Publishers, Boston/Dordrecht/London, 1992) 1st edn.

Deterministic weight evolution algorithm for solving local minima problem

Sin-Chun Ng, Chi-Chung Cheung and Shu-Hung Leung

A new algorithm is proposed to enable local minima to be escaped from by changing the weights of a multi-layer neural network in a deterministic way. Simulation results show that the new algorithm always outperforms other traditional methods in achieving global convergence.

Introduction: A local minimum is a suboptimal equilibrium point at which the system error is nonzero and the hidden output matrix is singular [1, 2]. The existence of local minima is a problem in the supervised learning of multi-layered feedforward networks. To solve this problem, we can change the hidden output matrix to a non-singular matrix by using evolutionary algorithms. In [3], we introduced a random perturbation to solve the problem of local minima. However, it can sometimes spend too much time in searching the wrong descent paths before achieving the correct one, and hence its computational complexity will be quite high. In this Letter, we propose a new weight evolution algorithm with deterministic perturbation which can solve the problem of local minima. We first select the worst input-output pair, i.e. the worst set of outputs with its corresponding input pattern. By considering its expected output, we then reverse the computation of the forward pass of back-propagation learning with some approximations to determine their corresponding weights. We then update the weights accordingly and continue the adaptation until the local minimum has been escaped from.

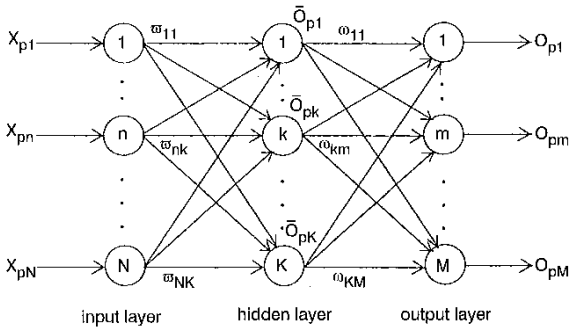

Fig. 1 *Basic stucture of two-layer feedforward network*