STOCHASTIC RELAXATION ALGORITHM FOR IMPROVED VECTOR QUANTISER DESIGN

Indexing terms: Image processing, Algorithms, Speech processing, Quantisation

An easily implementable stochastic relaxation algorithm for vector quantisation design is given. It generalises the usual Lloyd iteration in codebook design by perturbing the computed centroids with independent multidimensional noise, whose variance diminishes as the algorithm progresses. A significant improvement is often achieved.

Introduction: The design of vector quantisers plays an important role in developing high-quality voice and image communication systems. The traditional approach, called the LBG algorithm or generalised Lloyd algorithm (GLA),[1] iteratively updates the encoder and decoder of a quantiser according to the well-known nearest neighbour (NN) and centroid conditions. Each iteration results in a strict decrease in average distortion over the previous iteration. The algorithm converges in a finite number of iterations (assuming a finite training set representation of the source) and the resulting codebook is recorded. This descent algorithm constantly moves in the direction of the local minimum in average distortion associated with the initially chosen codebook.

A common problem with the GLA is that there may be a significant difference between the resulting local distortion minimum and the global minimum. In Reference 2, a simulated annealing (SA) approach is utilised in the codebook design process to obtain performance superior to the GLA. The algorithm is, however, quite complex and requires large amounts of programming effort and CPU time. The basic theory of SA and its applications can be found in Reference 3.

We introduce a conceptually simple but effective modification of the standard GLA that yields quantiser performance equal or superior to that in Reference 1, but with much smaller running time and a negligible increase in programming effort over the GLA. It utilises a simple form of stochastic relaxation (SR), a probabilistic technique for finding globally optimal solutions to complex optimisation problems. The core of the idea is to add an element of zero-mean noise to each codevector following the centroid computations in each iteration of the GLA. The noise variance (or 'temperature') is then reduced monotonically as the algorithm progresses. Performance increases equivalent to doubling the codebook size have been observed using this method, which in many applications corresponds to a bit-rate saving of 10–15%.

Definitions: Let $\boldsymbol{R}^k$ denote k-dimensional Euclidean space and let $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ be a training set of vectors from a source random vector $\boldsymbol{X}$ in $\boldsymbol{R}^k$ with variance σ_x^2. A vector quantiser is a mapping $Q : \boldsymbol{R}^k \rightarrow \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$, where each $\boldsymbol{y}_i \in \boldsymbol{R}^k$ is called a codevector (for $1 \leq i \leq N$) and the set of codevectors is a codebook.

The criterion of performance (objective function) assumed throughout this paper is the mean-square distortion function, given by the expectation

$$D = E\|\boldsymbol{X} - Q(\boldsymbol{X})\|^2 \tag{1}$$

For positive integers $1 \leq i \leq N$ we let $S_i(\sigma)$ denote a collection of independent identically distributed k-dimensional zero-mean random vectors with equal component variance $\sigma^2 < \infty$.

Stochastic relaxation and codebook design: SR is a successful technique that attempts to achieve solutions to inherently complex combinatorial optimisation problems that are very close to the globally optimal solution. In a typical SR algorithm, each iteration consists of perturbing the state of the system in some random fashion before computing the next state, with the ultimate goal of minimising the objective function. The magnitude of the perturbations generally decreases with time, so that convergence results. The common element of all SR algorithms is that increases in the value of the objective function are possible at each iteration.

SA is a special case of SR that utilises an energy function E which is inherently related to the objective function (often identical), and a temperature T which controls the degree of randomness in the algorithm. At each iteration of an SA algorithm, a perturbation is accepted if it results in a net decrease in energy ($\Delta E < 0$). If $\Delta E > 0$ then the perturbation is accepted with probability $\exp(-\Delta E/T)$. The temperature T is generally held constant until a 'thermal equilibrium' results and then decreased, eventually to zero ('freezing'). An application of SA was applied to the codebook design process in Reference 2 by altering the repartitioning phase of the standard Lloyd iteration, and in Reference 4 by perturbing random codevectors one at a time in the decoder.

In this letter we present an SR algorithm which is related to the method used in Reference 4. Our algorithm, however, provides important improvements in terms of computational efficiency and ease of implementation. Instead of adopting the SA approach we choose a more straightforward implementation. Zero-mean vector-valued noise is added to every codevector following each centroid computation in the Lloyd iteration. The variance of the noise (i.e. the temperature) decreases ('cools') as the algorithm progresses. The critical difference between this algorithm and SA is that the codevector perturbations are accepted unconditionally, whereas in SA their acceptance depends probabilistically on the values of ΔE and T. This is significant because it completely eliminates the need to compute the change in energy ΔE at each iteration.

Codebook design algorithm:

```
(a) Codevector initialisation: $y_1^{(1)}, \ldots, y_N^{(1)}$:
    $m = 1$
    $D_0 = \infty$
(b) Nearest neighbour repartition ($1 \leq i \leq M$):
    $j = \operatorname{argmin}\{\|x_i - y_l\| : 1 \leq l \leq N\}$
    Let $x_i \in R_j$
    $D_m = D_m + \|x_i - y_j^{(m)}\|^2$
(c) Stopping criterion:
    If $(D_{m-1} - D_m)/D_m < \varepsilon$ stop
    $m = m + 1$
(d) Centroid computation ($1 \leq i \leq N$):
    $y_i^{(m)} = 1/|R_i| \sum_{x_i \in R_i} x_i$
(e) Codevector jiggling ($1 \leq i \leq N$):
    $y_i^{(m)} = y_i^{(m)} + S_i(T_m)$
    Goto (b)
```

Perturbation noise: The type of perturbation applied to the codevectors has very little effect on the outcome of the algorithm. For simplicity we chose the noise random vector S_i to be uniformly distributed over a k-dimensional hypercube (i.e. scalar uniform PDF in each component). A random variable with a uniform PDF of width Δ has variance equal to $\Delta^2/12$. Hence, the width of the PDF depends quadratically on the variance. This allows us to decrease the PDF width directly in terms of the temperature T_m at each iteration.

Temperature schedules: There are many different temperature schedules that can be used to cool the system. Generally in SA algorithms the slower the cooling, the closer one gets to the global optimum. If the number of iterations is constrained then it is generally better to cool the system relatively fast at the beginning and then gradually approach freezing with slow temperature decreases.

Experimentally we have found that a reasonable initial temperature is on the order of the input variance σ_x^2. Three classes of cooling functions were investigated, one of which explicitly depends on a bound I on the total number of iterations allowed. Various choices for the parameters p and α can be utilised. The dependence of the algorithm's performance on the temperature schedule was significant, though the best schedule often varied somewhat depending on which source was used. The various cooling schedules are given below:

(*a*) $$T_m = \sigma_x^2\left(1 - \frac{m}{I}\right)^p$$

(*b*) $$T_m = \frac{\sigma_x^2}{(m+1)^p}$$

(*c*) $$T_m = \sigma_x^2 \alpha^m$$

In (*a*) schedules with p taking on the values 1, 2 or 3 were tested, and in (*b*) with p equal to 2, 1 or $\frac{1}{2}$. In (*a*) the best results occurred with $p = 3$ and in (*b*) with $p = \frac{1}{2}$. In (*c*) we chose $\alpha = 0{\cdot}95$, though other values also yield good results. The best schedule overall, of those we tested, appeared to be (*a*) with $p = 3$. Fig. 1 shows a comparison of the performance of the above algorithm and the GLA for a speech signal sampled at 8 kHz and scalar-quantised with a codebook of size 32. The two functions plotted are the signal-to-noise ratios of the quantisers as a function of the algorithms' iteration number. In this example, a 0·8 dB gain was achieved over the GLA using SR. In the context of speech coding this improvement is significant and often yields a perceivable quality improvement.

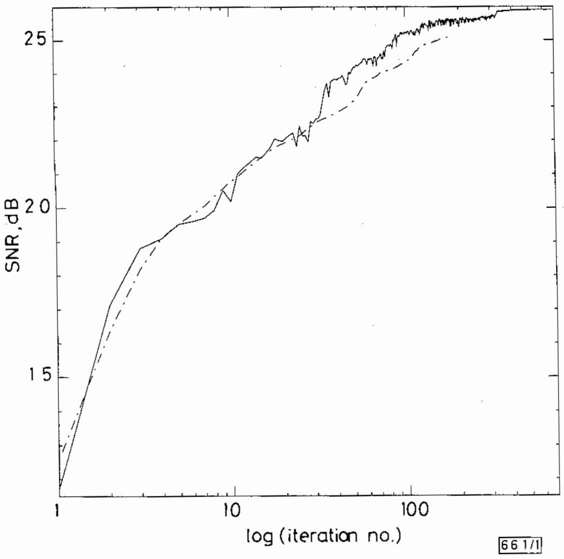

Fig. 1 *Performance comparison between stochastic relaxation and GLA*
——— GLA-SR   – – – GLA

Acknowledgment: This work was sponsored by the Bell Communications Research Inc., Rockwell Corporation, Bell-Northern Research Inc., and the State of California MICRO programme.

K. ZEGER
A. GERSHO

Department of Electrical & Computer Engineering
University of California
Santa Barbara, CA 93106, USA

17th March 1989

References

1 LINDE, Y., BUZO, A., and GRAY, R. M.: 'An algorithm for vector quantizer design', *IEEE Trans. Commun.*, 1980, **COM-28,** (1), pp. 84–95

2 VAISEY, J., and GERSHO, A.: 'Simulated annealing and codebook design'. Proc. Int. Conf. Acous., Speech, and Sig. Proc., New York City, April 1988, pp. 1176–1179

3 VANLAARHOVEN, P., and AARTS, E.: 'Simulated annealing: theory and applications' (D. Reidel Publishing Co., Dordrecht, Holland, 1987)

4 CETIN, A. E., and WEERACKODY, V.: 'Design vector quantizers using simulated annealing', *IEEE Trans. Circ. Syst.*, 1988, **CAS-35,** (12), p. 1550