A METHOD TO OBTAIN BETTER CODEBOOKS FOR VECTOR QUANTIZERS THAN THOSE ACHIEVED BY THE GENERALIZED LLOYD ALGORITHM

Eyal Yair* and Kenneth Zeger**

* IBM Israel Scientific Center, Technion City, Haifa 32000, Israel
** Dept. of EE, University of Hawaii, Honolulu, HI 96822, USA

Abstract

The most widely used technique for designing a vector quantizer (VQ) is the *Generalized Lloyd Algorithm* (GLA), which is an iterative down-hill search in which some distortion function is being minimized. The major drawback of the GLA is that a local minimum of the distortion function, which heavily depends on the initial guess, is generally found. Unfortunately, in many practical applications, repeating the GLA using several different initial guesses generally yields only marginal improvement of the outcome codebook and does not always justify the increase in computation. In this paper we introduce an alternative approach to VQ design that produces better codebooks than the GLA (in the sense of lower distortion). The method is based on an on-line design technique (similar to the Kohonen learning scheme) which incorporates principles of stochastic relaxation. It is called the *Soft Competition Scheme* (SCS). The SCS is a deterministic on-line design procedure. In contrast to the Kohonen *winner-take-all* competition carried out between the candidate codevectors, it performs a "soft" competition where each codevector is assigned a *winning score* and all the codevectors are updated simultaneously according to these scores. A temperature schedule is used to control the speed of convergence versus the quality of the codebook.

1. Introduction

Vector quantization (VQ) design is a minimization problem of a multi-dimensional distortion function which, in most practical applications, contains multiple local minima. The outcome of this minimization is a set of prototype vectors called *codevectors* which are used to represent data vectors from the source being quantized. The collection of all the codevectors is called a *codebook*. At present, the most widely used algorithm for VQ design is the *generalized Lloyd algorithm* (GLA) [1] . The GLA is a descent algorithm in which the distortion function monotonically decreases towards a local minimum. The GLA is easy to use and is guaranteed to converge. However, as in most descent algorithms, its major drawback is that there is no control on the quality of the local minimum achieved after convergence. Once the initial guess has been chosen, the final solution to which the iterations converge has also been determined. There are many heuristic ways as how to choose the initial guess (e.g., [2]) but unfortunately, none of which can guarantee a good performance of the final codebook. A possible remedy to this limitation of the algorithm might be to repeat the GLA with several initial guesses, hoping that one of them will yield a satisfactory solution. However, in practice, this technique yields only marginal improvement due to the large number of local minima of the distortion function. Another solution is to insert some kind of stochastic perturbations in order to allow up-hill transitions on the distortion landscape. Up-hill transitions allow the search to escape from local minima, but should also be controlled so that the search will finally converge. Several types of stochastic perturbations were suggested for VQ design (e.g., [1], [3], [4]) where in most of them an improvement over the GLA could be obtained at the expense of a significant increase in complexity. It turns out that stochastic methods, while capable of finding good solutions, require exhaustive computation time to supply such solutions.

Another type of VQ design method was introduced by Kohonen [5] and in a slight different form by Chang and Gray [6]. It is an on-line method in which, at each time instant, a single training vector is presented, and the closest codevector migrates towards it with a diminishing step-size. The update of the closest codevector can be regarded as if there is a competition between the codevectors and only the winner of that competition is updated. The on-line method is not a descent procedure but rather an LMS type algorithm in which the direction of the instantaneous gradient (sometimes called the stochastic gradient [6]) is taken at each update. Convergence is achieved when the migration step-size reduces to zero. However, as was shown in [7], care must be taken in reducing the step-size in order to ensure convergence to a local minimum of the distortion function. The on-line algorithm was found to yield similar performance to the GLA for VQ applications (e.g., [6], [8]).

In this paper we suggest a method which is a mixture of the on-line method and the stochastic relaxation techniques. It is somewhat more complex than the GLA but produces better codebooks. Moreover, better codebooks are obtained even if repetitions of the GLA (starting from different initial guesses) are allowed, such that the overall computation time is identical to that of the suggested method. The new method, called the *soft competition scheme* (SCS), is an on-line deterministic method. At each time instant a training vector is presented and, in contrast to the on-line Kohonen method in which only the closest codevector was updated, all the codevecotrs are simultaneously updated, each codevector with its own step-size. In order to determine the step-size for each codevector, a soft competition is performed between them, and a *winning score* is assigned to each codevector, depending on its proximity to the training vector presented. The winning score, representing the probability of the codevector to win the competition, is similar to the local Gibbs probability of the Gibbs Sampler introduced by Geman and Geman [3], in which the codevector to be updated was chosen probabilistically according to this local probability distribution. The step-size for each codevector is then factored by these winning scores, resulting in graded migration steps, where the codevectors closer to the training vector presented migrate with large steps and codevecotrs far from that training vector migrate with small steps. A temperature schedule, similar to that of simulated annealing algorithms (e.g., [9]), is used to shape the width of the winning probability distribution such that as the temperature approaches zero, the SCS approaches the on-line Kohonen scheme and the algorithm converges. The SCS provides better codebooks than the GLA since the large migration steps taken while the temperature is high enable the algorithms to find better partitions of the training set than the partition originally dictated by the initial guess. The SCS can be also utilized in conjunction with the GLA, where it is used as a mechanism to find a good initial guess for the GLA.

2. The on-line Kohonen update formula

In the on-line approach, the training vectors are presented one at a time, and the codebook is updated upon each presentation of a training vector. We start from an initial guess for the codebook and iterate until convergence while cycling through the training set. Each full cycle over the training set is called a sweep. The basic on-line update formula at the n-th iteration is as follows. A training vector $\mathbf{x}(n)$ is presented and the distances between $\mathbf{x}(n)$ to all the codevectors of the current codebook are evaluated. The codevector closest to $\mathbf{x}(n)$, denoted by $\mathbf{y}_c(n)$, is then updated according to

$$\mathbf{y}_c(n) = \mathbf{y}_c(n-1) + a_c(n)\,[\,\mathbf{x}(n) - \mathbf{y}_c(n-1)\,] \tag{2.1}$$

where all the remaining codevectors are left unchanged, i.e., $\mathbf{y}_j(n) = \mathbf{y}_j(n-1)$ for all $j \neq c$. The quantity a_c is called the step-size of $\mathbf{y}_c$. It was shown [7] that it is best to hold a counter n_i for each of the codevectors $\mathbf{y}_i$ $(i = 1,...K$, where K is the number of codevectors) which is incremented after each update of $\mathbf{y}_i$, and to set the step-size for each codevector equal to $a_i(n) = 1/n_i(n)$.
Let us define for every codevector $\mathbf{y}_i$ (for $i = 1, ..., K$) and a training vector $\mathbf{x}$ a selection function $S_i(\mathbf{x})$ which obtains the value "1" for $i = c$ and "0" for $i \neq c$ (where c indicates the index of the codevector which is closest to $\mathbf{x}$). Then, the update formula for the whole codebook at the n-th iteration can be written as

$$\mathbf{y}_i(n) = \mathbf{y}_i(n-1) + a_i(n)\,S_i(\mathbf{x}(n))\,[\,\mathbf{x}(n) - \mathbf{y}_i(n-1)\,] \tag{2.2}$$

for $i = 1,...,K$, and the counter update formula is given by

$$n_i(n) = n_i(n-1) + S_i(\mathbf{x}(n)) \quad . \tag{2.3}$$

The procedure can be regarded as a competition between the codevectors where only the winner is updated towards the training vector presented to it, and after each update its counter is incremented (resulting in a reduction of the step-size).

3. The SCS algorithm

The concept of a winner-take-all competition leads to an unrobust procedure with high sensitivity to the initial guess. For example, some of the codevectors may never win for the entire procedure and will represent empty quantization cells in the final codebook. On the other hand, a codevector that is attracted to a high density region of the training data tends to solely represent all the data in this region since, because it is always the sole winner in this region, it prevents other codevectors to approach the region. In spite of these limitations, on-line methods were found to produce equivalent quality to the GLA method for speech and image VQ applications [6], [8].

In this section we present an improved on-line scheme which overcomes some of the major drawbacks of the winner-take-all method at the expense of some increase in complexity (namely, a slower convergence). In this scheme, which we call the *soft competition scheme* (SCS) we relax the restriction of a sole winner for each competition and instead attach a winning score to each of the codevectors. This winning score is the probability of the codevector to win, based on some probability model, and hence the competition becomes "soft". We note that in the framework of a *winner-take-all* competition of (2.2), the selection function $S_i(.)$ can be regarded as the probability of the codevector $\mathbf{y}_i$ to win while this probability is either "1" or "0". This is a very simple model for expressing the winning probabilities based on proximity measurements. In the SCS algorithm, we thus replace the binary functions $S_i(\mathbf{x}(n))$ by a continuous probability expression, $P_i(n)$, which represents the probability of $\mathbf{y}_i$ to win upon presentation of $\mathbf{x}(n)$ and satisfies

$$\sum_{i=1}^{K} P_i(n) = 1 \tag{3.1}$$

Hence, at each iteration, all the codevectors are *simultaneously* updated, each codevector with its own step-size, according to

$$\mathbf{y}_i(n) = \mathbf{y}_i(n-1) + a_i(n)\,P_i(n)\,[\,\mathbf{x}(n) - \mathbf{y}_i(n-1)\,] \tag{3.2}$$

and the counters are modified according to

$$n_i(n) = n_i(n-1) + P_i(n) \tag{3.3}$$

The winning score $P_i(n)$ factors the step-size $a_i(n)$ such that the larger the probability of $\mathbf{y}_i$ to win, the larger its step-size is, and vice-versa. The probability model for proximity is taken as the local Gibbs distribution, as was suggested by Geman and Geman for stochastic relaxation algorithms [3] where is was proven that if the winning codevector is chosen probabilistically with the Gibbs distribution, the global minimum of the distortion function can be achieved (provided that the annealing schedule for reducing the temperature of the distribution is sufficiently slow). Denoting by $d(\mathbf{x}, \mathbf{y})$ an appropriate distance measure between the vectors $\mathbf{x}$ and $\mathbf{y}$, the probability score $P_i(n)$ is thus given by

$$P_i(n) = \frac{1}{Z_n}\, e^{-\beta_n\, d(\mathbf{x}(n),\, \mathbf{y}_i(n-1))} \tag{3.4}$$

where the normalization factor Z_n is given by

$$Z_n = \sum_{k=1}^{K} e^{-\beta_n\, d(\mathbf{x}(n),\, \mathbf{y}_k(n-1))} \tag{3.5}$$

The quantity β_n is the reciprocal of the physical temperature of the Gibbs distribution at the n-th iteration, and it satisfies

$$\lim_{n\to\infty} \beta_n = \infty \ , \tag{3.6}$$

where the specific increment procedure of β_n is called the *annealing schedule.*

Note that when $n \to \infty$, the probability factor $P_i(n) \to S_i(\mathbf{x}(n))$ and the soft competition becomes the winner-take-all competition. However, for small values of β (high temperature values) the codevectors migrate in the space and slowly improve its partition into quantization cells. If the annealing schedule of (3.6) is slow, the algorithm is not sensitive to the initial guess of the codebook. The step-size for each codevector is still $a_i(n) = 1/n_i(n)$, but it should be occasionally reset, otherwise, the convergence might be too slow.

4. Practical Considerations

We now address some of the practical aspects regarding the implementation of the SCS algorithm. The most critical aspect is the implementation of the annealing schedule of (3.6). To better understand the role of the temperature (or its reciprocal β) let us examine the behavior of the algorithm in the two extreme values, $\beta \to 0$ and $\beta \to \infty$. When β is small, the probability distribution of (3.4) approaches a uniform distribution, which means that all the codevectors migrate with about the same step-size for each training vector that is presented to them. The high temperature environment is thus a melting stage which causes the codebook to disconnect itself from the initial guess. However, spending a long time in high temperatures causes the codevectors to approach each other and thus a long time will also be required to separate them again. On the other hand, when β is high, the probability distribution becomes a delta function around the winning codevector, in which case only this winner is updated and the algorithm becomes the

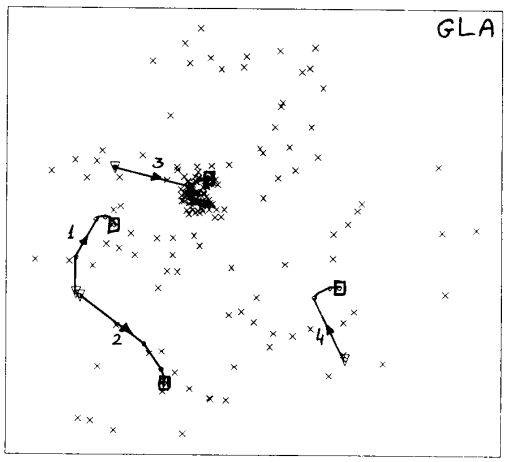

Figure 1. Codebook trace for a GLA search.

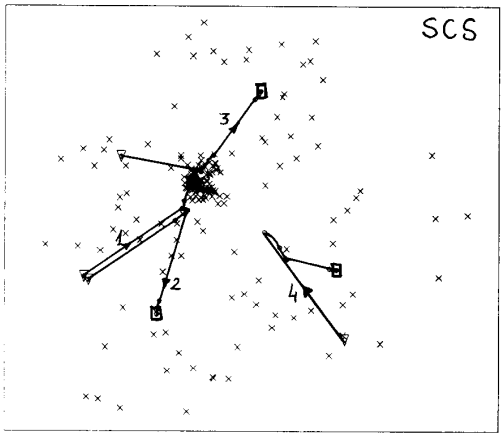

Figure 2. Codebook trace for an SCS search.

winner-take-all learning, which is an LMS type learning algorithm. The above observations of these two extreme situations suggest that the rate of annealing (i.e., the rate of increasing β) should vary in such a way that it is high for low β values and moderate for high values of β . Probably the optimal rate found for the stochastic relaxation algorithms (which was shown to be logarithmic, i.e., $\beta_n = \Gamma \ln(n)$ [3]) is also the best schedule to use for the SCS. In our experiments we have used exponential rate which is must faster and obviously not the best choice to use. In such a fast annealing, one must avoid the merging of the codevectors by bounding the initial value of β from below. By examining the use of β in the exponential terms of (3.4) and (3.5) we define:

$$\sigma_x^2 = \frac{1}{N} \sum_{n=1}^{N} d(\mathbf{x}(n), \bar{\mathbf{x}}) \tag{4.1}$$

where n is the number of training vectors, and $\bar{\mathbf{x}}$ is the sample mean of the training set. The quantity σ_x^2 is equivalent to the sample variance when the distortion measure is Eucledian. The initial value of β, denoted by β_0 is determined by: $\beta_0 = C/\sigma_x^2$ where C is a constant which is approximately unity. With exponential annealing schedules we have found $0.1 \leq C \leq 1$ to supply good results.

In applying practical annealing there are two basic approaches. The first one is the "continuous" approach in which β is incremented after each update, namely $\beta_n = \beta_0 \gamma^n$, where n is the iteration index. The second approach is to modify β occasionally, e.g., at the end of each sweep $\beta_m = \beta_0 \delta^m$, where m is the sweep index, and β is kept constant within the sweep. These two approaches correspond to the inhomogeneous and homogeneous annealing schedules of Markov chains in simulated annealing algorithms (see e.g. [10]). In our experiments we have found the "continuous" approach to yield better results with $1.001 \leq \gamma \leq 1.01$.

Another consideration should be given to the step-size update. It is advised to occasionally reset the codevector counters to some predetermined value η in order to accelerate the convergence. Typically, the time interval between two successive resets should increase with time. We have chosen a linear growth of these intervals by resetting every time the sweep index equals a perfect square. Typically, $1 \leq \eta \leq 10$.

Finally, there are several options to terminate the algorithm. A basic stopping condition checks the relative change of the distortion function in the last two sweeps and compares it to some threshold. Another option is to use two phases: a first phase which is terminated by a higher threshold, which means fewer iterations, which is followed by either a GLA or a winner-take-all phase. That is, a SCS phase is used to find a good initial codebook for the second, descent phase.

In most of the cases we have examined we found that best solutions were obtained by concluding the SCS phase with a few (1-3) GLA iterations. Our experiments included both Gauss-Markov and speech sources, with vector dimensions ranging from 2 to 8, and codebook sizes of $64 \leq K \leq 256$. In all these test, the SCS approach consistently produced better codebooks than the GLA (in term of better SNR values) even when the GLA was allowed to start from several different initial guesses in each test, such that the total computation time allotted for both algorithms was equal.

4.1. Example

Let us conclude with a simple example which illustrates some of the practical considerations discussed above. Consider two Gauss-Markov sources in the 2-dimensional Eucledian space given by

$$\begin{aligned} \mathbf{x}_1(n) &= 0.9\mathbf{x}_1(n-1) + \mathbf{w}_1(n) \\ \mathbf{x}_2(n) &= 0.7\mathbf{x}_2(n-1) + \mathbf{w}_2(n) \end{aligned} \tag{4.2}$$

where $\mathbf{w}_1$ and $\mathbf{w}_2$ are i.i.d. Gaussian processes with variances $\sigma_1^2 = 1$ and $\sigma_2^2 = 36$ respectively. The training set is composed of equal number of samples from these processes (Figures 1,2). In Figure 1 the GLA algorithm was used yielding an SNR of 4.60dB in 5 iterations. The figure illustrates the trace of a codebook of four codevectors. The initial guess, chosen randomly in the square is marked by triangles. The final codebook is marked by squares. The crosses indicate the location of the 200 training vectors. In Figure 2 an SCS algorithm was applied to the same training set, starting from the same initial codebook. The resultant SNR, obtained after 7 sweeps was 5.33dB. Note that each SCS sweep is equivalent in computation to a single GLA iteration. The parameters used for this example were $C = 0.25$ and $\gamma = 1.001$. Applying a winner-take-all competition to the final codebook of the SCS (which means an infinite value of β) resulted in an SNR of 5.41dB and 3 more sweeps. Applying the GLA to the final codebook of the SCS (instead of the winner-take-all competition phase) resulted in the same solution of 5.41dB but with a single GLA iteration (i.e., total of 8 sweeps to reach this solution).

References

[1] Y. Linde, A. Buzo, and R.M. Gray, **An algorithm for vector quantizer design,** *IEEE Trans. on Communications,* Vol. COM-28, 84-95, 1980.

[2] R.M. Gray, **Vector quantization,** *IEEE ASSP Magazing,* Vol. 1, 4-29, 1984.

[3] S. Geman and D. Geman, **Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images,** *IEEE Trans. on Patt. Anal. and Mach. Intel.,* Vol. PAMI-6, 721-741, 1984.

[4] K. Zeger, J. Vaisey, and A. Gersho, **Globally optimal vector quantizer design by stochastic relaxation,** *IEEE Trans. on Acoustics, Speech, and Signal Processing,* to appear in 1991.

[5] T. Kohonen, **Self organization and associative memory,** *Springer-Verlag,* Berlin, 1984.

[6] P.C. Chang and R.M. Gray **Gradient algorithms for designing predictive vector quantizers,** *IEEE Trans. on Acoustics, Speech, and Signal Processing,* Vol. ASSP-34, 679-690, 1986.

[7] E. Yair, K. Zeger, and A. Gersho, **Competitive learning and soft competition for vector quantizer design,** *IEEE Trans. on Acoustics, Speech, and Signal Processing,* to appear in 1991.

[8] N.M. Nasrabadi and Y. Feng, **Vector quantization of images based upon the Kohonen self organization feature maps,** *Proc. of the 2nd ICNN Conf.,* Vol. I, 101-105, 1988.

[9] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, **Optimization by simulated annealing,** *Science,* Vol. 220, 671-680, 1983.

[10] P.J.M. van Laarhoven and E.H.L. Aarts, **Simulate Annealing: Theory and Applications,** *D. Reidel Publishing Company,* Dordrecht Holland, 1987.