Vector quantization and fuzzy ranks for image reconstruction

Stefano Rovetta [a], Francesco Masulli [b]

[a] *University of Genova, Italy – Department of Computer and Information Sciences – Via Dodecaneso, 35 I-16146 Genova (Italy)*

[b] *University of Pisa, Italy – Department of Computer Science – Largo B. Pontecorvo, 3 I-56127 Pisa (Italy)*

Abstract

The problem of clustering is often addressed with techniques based on a Voronoi partition of the data space. Vector quantization is based on a similar principle, but it is a different technical problem. We analyze some approaches to the synthesis of a vector quantization codebook, and their similarities with corresponding clustering algorithms. We outline the role of fuzzy concepts in these algorithms, both in data representation and in training. Then we propose an alternative way to use fuzzy concepts as a modeling tool for physical vector quantization systems, Neural Gas with a fuzzy rank function. We apply this method to the problem of quality enhancement in lossy compression and reconstruction of images with vector quantization.

Key words: Vector quantization; Neural Gas; Fuzzy ranks.

1 Introduction

The problem of clustering [1] is often addressed with the partitive, centroid-based approach of the *c*-Means procedure and many other derived algorithms. In this approach clustering is viewed as *finding the reference vectors (centroids) which best explain the input data distribution according to some cost criterion*. This goal is best achieved with a small or moderate number of centroids (clusters), to obtain a model for the data under study which is as simple and understandable as possible.

Vector quantization [2,3] is a different technical problem, which can be stated as follows: *find the reference vectors (codevectors) which approximate with the minimum error the input data according to some distortion criterion*. Usually the problem is also constrained by some resource limits. This is the rationale for the so called Rate-Distortion theory. In this case, the number of codevector is to be maximized within the allowed constraints, to keep distortion as low as possible.

The typical application of vector quantization is in signal and image processing, although alternative applications have been proposed [4,5]. Vector quantization provides a form of lossy block coding [6], whereby blocks of samples in a sequence or blocks of pixels in an image, represented as vectors in a suitably dimensioned space, are approximated by replacing them with a single codevector, selected to minimize a given distortion (quantization error) measure. Block coding is more efficient than sequential coding. Due to the higher correlation between nearby samples, in bidimensional signals (images) the technique is even more effective, and accordingly it is often used in moderate quality, very low bit rate applications.

In this work, we analyze some approaches to the synthesis of a vector quantization codebook, and their similarities with corresponding clustering algorithms. We outline the role of fuzzy concepts (such as membership in more than one Voronoi polyhedron) in the performance of these algorithms. Then, we propose an alternative use of the fuzzy paradigm in the vector quantization training algorithm by Martinetz *et al.*, the "Neural Gas" [7]. The techniques and concepts discussed will be applied in the proposed formulation of Neural Gas vector quantizer design.

2 Clustering, vector quantization, and fuzzy concepts

2.1 The goals of clustering and vector quantization

We have outlined some differences between clustering and vector quantization, yet the synthesis of a codebook for vector quantization is often approached with algorithms derived from *c*-Means (a standard clustering technique). This is the approach introduced by Linde, Buzo and Gray [8] and is therefore often referred to as the LBG approach. One usual feature of the LBG-style techniques with respect to clustering is that, since vector quantization is typically adopted for large-sized training sets and reference vector sets, minimization is performed by stochastic gradient descent (*online* training) [9] rather than by batch algorithms. This is because the curse of local minima is worsened by the moderate-to-large dimensionality and larger codebook size. Stochastic optimization [10,11] helps escaping local minima by adding errors (due to random sampling of patterns) to the current estimate of the cost function. Therefore there is a nonzero probability of taking steps in directions other than that of the "closest" local minimum.

The similarity between the clustering and quantization problems is of a geometrical nature. In both cases, the input space is partitioned by a *Voronoi tessellation* [12], representing regions of data sharing similar properties by means of a single reference point or site or, in the respective jargons of vector quantization and clustering, *codevector* or *centroid*. Furthermore, in both cases the reference points obey the principle of being the barycenter of all points included in a given cluster or Voronoi region.

This is a necessary condition for optimality in the case of a squared error measure of

distortion:

$$D = \frac{1}{n} \sum_{k=1}^{n} ||\mathbf{x}_k - Q(\mathbf{x}_k)||^2 \tag{1}$$

where the data point $\mathbf{x}_k$ is compared to reference points $\mathbf{y}_j$, $j \in \{1, c\}$ and its nearest reference point is denoted by $Q(\mathbf{x}_k)$.

However, there is a distinction in the goal to be achieved. Generally speaking, in clustering we want to define a cluster as the largest group of data that can be reasonably gathered in a single group: clusters should be as few as possible to enable understanding the structure of data. However, in vector quantization, points in a region must be so similar that the approximation error obtained by substituting these data with the nearest codevector is negligible. Thus in the case of vector quantization codevectors should be as many as possible, within the resource limits imposed by the overall system design.

2.2 Why fuzzy versions?

When designing a fuzzy algorithm, for instance the fuzzy version of an existing crisp technique, the technical problems we want to address are different in the case of clustering and vector quantization.

Several clustering algorithms have been modified in the direction of incorporating fuzzy concepts (starting with the Fuzzy *c*-Means algorithm [13]). A review of fuzzy concepts in clustering is provided in [14,15]. In the large majority of cases, fuzziness means that any point can belong to more than one cluster, to different degrees.

The introduction of a fuzzy membership has a twofold meaning in clustering. On one side, data can be partially belonging to more than one cluster, and this has a conceptual interpretation: it is possible to analyze and quantify whether points are clearly clustered or there is any ambiguity in cluster attribution. On the other side, fuzziness is a way to fight local minima during optimization.

In vector quantization, the first aspect is irrelevant, since at the end of training a crisp decision must always be made. The other aspect is more important, since in the typical vector quantization application local minima are a serious issue.

3 Codebook design methods

In the following we briefly review how typical algorithms for the synthesis or "training" of vector quantization codebooks introduce fuzzy concepts in the minimization procedure, and what is their effect. We will assume that N training points (individually denoted with $\mathbf{x}$) of dimensionality d are used to design a codebook $\{\mathbf{y}_1, \ldots \mathbf{y}_c\}$ of c reference points.

The distorsion assumed is the squared Euclidean distance $d_j = ||\mathbf{x} - \mathbf{y}_j||^2$ yielding the squared-error distortion already introduced.

3.1 Lloyd's and MacQueen's methods

The classical approach is Lloyd/MacQueen's method [16,17,8], the standard *c*-Means clustering procedure. The *k*-th input vector is attributed to the Voronoi polyhedron defined by reference vector $\mathbf{y}_j$ if $u_{jk} = 1$, where u_{jk} is a membership indicator, a crisp value which is 1 if $d_{jk} = \min\{d_{1k}, \ldots, d_{Nk}\}$ and 0 for all other reference vectors, so that $Q(\mathbf{x}_k) = \sum_{j=1}^{c} u_{jk}\mathbf{y}_j$. The closest reference vector $\mathbf{y}_j = Q(\mathbf{x}_k)$ for a data point wil be called the "winner" for that point, The updating rule is:

$$\mathbf{y}_j^{(t+1)} = \frac{\sum_{k=1}^{N} \mathbf{x}_k u_{jk}}{\sum_{k=1}^{N} \mathbf{x}_k} \tag{2}$$

This rule defines a minimization by Picard iterations, in which at each step a necessary minimum condition is satisfied. This algorithm finds the minimum of a cost function based on the mean square error as a distortion criterion. Its well-known drawback lies in the huge number of local minima (for practical d and N).

Note that membership indicators u_{jk} obey the following normality condition:

$$\sum_{j=1}^{c} u_{jk} = 1 \ \ \forall k \in \{1, n\}, \tag{3}$$

that is, each point can belong only to one cluster.

The on-line version of *c*-Means training is due to MacQueen. It transforms the Picard iteration of the standard version in a stochastic optimization process. Input vectors are randomly selected, adding noise to the cost function, now optimized on the average. The updating rule is therefore:

$$\mathbf{y}_j = \mathbf{y}_j^{(t)} + \eta^{(t)} u_{jk} \left(\mathbf{x}_k - \mathbf{y}_j\right) \tag{4}$$

where t indexes the training steps, $\eta^{(t)}$ is an updating coefficient, and k is a random function of t.

Convergence is usually much slower, although this may not be true for very large and redundant data sets. However, the advantage is that local minima are escaped thanks to the "statistical" behaviour of the updating procedure, which does not necessarily reduce the cost at each step and therefore does not necessarily get trapped into sub-optimal basins.

The law for varying $\eta^{(t)}$ to ensure convergence (annealing schedule) has been studied in [18] for the Gibbs sampler. MacQueen [17] adopts an individual coefficient for every

reference vector, equal to $1/t_j$ where t_j is the number of updates for reference vector $\mathbf{y}_j$ so far, thus retaining the exact equivalence between the online and batch versions of *c*-means. Ritter *et al.* [19] propose instead a faster exponential decay rate $\eta^{(t)} = \eta_\mathrm{i} \left(\eta_\mathrm{f}/\eta_\mathrm{i}\right)^{t/t_\mathrm{max}}$ from η_i to η_f in t_max steps. This law has been used also in the Neural Gas algorithm.

3.2 Fuzzy c-Means

The most popular algorithm for clustering in the fuzzy framework, the "Fuzzy *c*-Means" [13] or "Fuzzy ISODATA" [20], has no direct counterpart in the vector quantization practice. Here the standard (crisp) *c*-Means membership is replaced by a fuzzy membership defined as a function of the point-prototype distance:

$$u_{jk} = \left[\sum_{l=1}^{c} \left(\frac{d_{jk}}{d_{jl}} \right)^{1/(m-1)} \right]^{-1} \tag{5}$$

In this case, it turns out that membership values are no longer 0 or 1: we have instead $u_{jk} \in [0,1] \subset \mathbb{R}\ \forall j,k$.

The centroids are still computed according to the barycenter principle, although in this case we have actual weights instead of indicators (binary values):

$$\mathbf{y}_j^{(t+1)} = \frac{\sum_{k=1}^{N} \mathbf{x}_k u_{jk}}{\sum_{k=1}^{N} \mathbf{x}_k} \tag{6}$$

and the memberships still obey the normality condition (3), which in this case it is also termed "probabilistic constraint" since it makes membership values equivalent to a set of probabilities for mutually exclusive events.

The parameter $m \in [1,\infty] \subset \mathbb{R}$ is a fuzziness index which has a direct influence on the actual values obtained for the memberships. When $m = 1$, the partition is crisp, whereas for $m \to \infty$ the memberships tend to have all the same value of $1/nc$. This index may be set to incorporate a-priori knowledge on the problem, but in the absence of such knowledge there is no established way to assess its value, although many approaches are possible (e.g., by cluster validation).

3.3 Maximum Entropy approach (the Deterministic Annealing method)

The maximum entropy approach of the Deterministic Annealing technique by Rose [21] builds on a different concept. Here a fuzzy membership in clusters is introduced by substituting the "min" selection criterion, by which a single reference vector is selected for

updating on a minimum-distance basis, with a "softmin" criterion:

$$u_i = \frac{e^{-d_i/\beta}}{\sum_{j=1}^{c} e^{-d_j/\beta}} \tag{7}$$

The parameter β governs the fuzziness of this criterion; for $\beta \to 0$ it turns back into the crisp "min" criterion. The Deterministic Annealing approach is a sequence of minimizations (made by Picard iterations), with β decreasing at each minimization. Therefore the first minimizations are done with a high degree of fuzziness, that is, high β (with few local minima), whereas the last minimizations, with $\beta \to 0$, are potentially subject to local minima, but they take advantage of the good initialization provided by previous steps.

Memberships are again subject to (3), and this is justified in this case by the explicit treatment of memberships as formal probabilities. Points are viewed as abstractions of physical particles. The approach is based on minimization of a cost functional which includes entropy of the partitions as a cost term, and energy of partitions as a constraint. By gradually lowering energy, a simulated annealing procedure is obtained. However, since the energy is updated only at convergence of the previous optimization step, the procedure is termed "deterministic annealing". The fuzziness parameter here is interpretable as a formal temperature, and it is the responsible for fixing the energy level of each step.

The specific form of memberships in (7) is derived from necessary conditions for minimum of the cost function just described, with the addition of the probabilistic constraint (3).

3.4 Possibilistic approach

Another popular fuzzy clustering approach which is not commonly used in vector quantization practice is the Possibilistic Approach by Krishnapuram and Keller [22]. We cite it here for completeness.

In the possibilistic case, a higher level of fuzziness is introduced by relaxing the requirement of memberships to all prototypes for each point summing up to 1 (3), which is enforced in all other methods. This changes considerably the principle of operation of the method, and is not compatible with vector quantization goals. The possibilistic approach is aimed at data understanding, featuring robustness properties with respect to outliers [23].

The memberships in this case are subject to the following set of weak constraints:

$$u_{jk} \in [0,1] \; \forall j \; \forall k \tag{8}$$

$$0 < \sum_{k=1}^{n} u_{jk} < n \quad \forall j \tag{9}$$

$$\forall k \; \exists j : u_{jk} > 0 \tag{10}$$

which only imply that no cluster be empty and each pattern be assigned to at least one cluster.

In principle, a point can now be attributed to more than one cluster with a high level of membership, although additional penalty terms in the cost function may impose a bias toward the normality condition. However, this is not a hard constraint, and it can be violated. Therefore it may not be possible to perform a final de-fuzzification of the resulting memberships, and a single best approximating codevector may not be found.

3.5 The Neural Gas algorithm

The Neural Gas algorithm by Martinetz *et al.* [7] combines fuzzy membership in partitions with stochastic minimization. This algorithm has the interesting feature that membership in a Voronoi polyhedron is not defined as a direct function of the distance from the data point to the reference vector, as in previously cited methods. Rather, it is a function of its rank with respect to the list of distances from all reference vectors. Distance d_i has the rank ρ_i in the set $\{d_1, \ldots, d_N\}$ when ordered decreasingly with respect to values, and this value can be written in an algebraic fashion as:

$$\rho_i = \sum_{j=1}^{c} \theta\left(d_i - d_j\right) \tag{11}$$

$\theta(x)$ is the Heaviside step function, taking on the values 0 for $x < 0$, 1 for $x > 0$, and 0.5 for $x = 0$. This extension is needed in the case of ties, very uncommon if the distances are real numbers; however this is the standard way to deal with ties in rank tests (such as Spearman's rank correlation or Kendall's rank correlation and coefficient of concordance). Notice that $\rho_{\text{winner}} = 0$ rather than 1, so $\rho_i \in \{0, \ldots, c-1\}\ \forall i \in \{1, \ldots, c\}$.

The membership of the data point to the i-th encoding polyhedron is:

$$u(\mathbf{x}) = e^{-\rho_i/\lambda} \tag{12}$$

where λ is a parameter which is annealed (made smaller) during training, thereby progressively reducing the extent to which reference vectors, other than the nearest (the "winner"), are included in the updating process.

The annealing of the two parameters (λ, influence of prototypes other than the "winner", and learning coefficient) can be interpreted from the standpoint of learning machine capacity. When vectors other than the winner get updated a correlation is introduced between reference vectors, thus effectively reducing the learning capacity of the vector quantizer. As the annealing proceeds, the range of the correlation shrinks gradually, and the capacity is correspondingly increased; however, at the same time the learning coefficient is reduced, so that it is progressively more difficult to fall into local minima. (To relate vector quantization, an approximation procedure, to the theory of learning capacity it is necessary to adopt a threshold-based criterion. This analysis is introduced in [24].)

3.6 Kohonen's Self-Organizing Maps

In this review, we must also mention Kohonen's Self Organizing Maps [25], in which fuzziness is of the same nature as in the Neural Gas technique, that is, in the influence of non-winners on the update of the winner. This method is neither a clustering algorithm nor a vector quantization algorithm; it is rather conceived as a multidimensional scaling technique, but we cannot avoid mentioning it due to its importance and influence on the subject, and particularly on many vector quantization algorithms, including Neural Gas itself.

3.7 Interval Vector Quantization

A different way to include fuzziness into vector quantization is making the *codevectors themselves* fuzzy. Although adopting this formulation can lead to a computationally inefficient algorithm, this approach can be simplified by representing uncertainty by means of interval values. This has been done in [26]. Uncertain codevectors can be defuzzified by applying some additional criterion (for instance, regularity or smoothness of the overall reconstructed image), which acts as a constraint and helps obtaining better perceived quality.

4 Image compression by vector quantization

When performing the specific task of image compression, the most basic procedure is as follows. The description is for single channels of an RGB image or for gray/level images. For images encoded according to other formats (e.g. HSV or composite video) there may be additional processing steps to take advantage of the more meaningful structure of pixel representation.

First, the image is split into square blocks of a given size, usually 4, 8, or 16 pixels. Then, each block is preprocessed to reduce inter-block variability. Usual preprocessings include subtracting the block average (which will be stored and encoded on its own), trimming the extreme values to predefined limits (on the basis of the consideration that details in very light or very dark areas are not as distinguishable as those in the middle of the intensity range), normalizing the values into a given range.

Each block is finally rasterized, and its linearized version thus obtained is regarded as a data vector. Therefore we may have typical vector sizes of 16, 64, or 256 (for 4, 8, and 16 pixel block sizes respectively). These data vectors are compared to codevectors in a codebook (which is tailored on the specific signal statistics). Each data block is encoded (approximated) with the best matching vector in the codebook. This results in vector indexes to be transmitted in place of whole data vectors.

At this phase, the compression ratio attained is expressed in terms of block size, N, bits per pixel b_p in the original image, and bits per codevector index b_c in the compressed image:

$$R_{\mathrm{VQ}} = \frac{N b_p}{b_c}. \tag{13}$$

Note that, of course, $b_c = \lceil \log_2 c \rceil$ where c is codebook cardinality, and on the practical side this mandates the use of numbers of codevectors which are powers of 2 to avoid wasting code space.

We must add for completeness that there is usually an additional step consisting of channel encoding (e.g., Huffmann compression), and also the block averages should be transmitted, so that the actual compression ratio is not given simply by the ratio data size/codebook index size.

Image reconstruction is performed by retrieving each indexed codevector from the codebook, for use in place of the image blocks they approximate. Each preprocessing step should be reversed (e.g., the respective block averages must be summed again to block values) and the resulting blocks are then displayed.

5 A fuzzy model of the ranking function

5.1 Fuzzy ranks

The performance of the Neural Gas algorithm is remarkably good, as found in previous research by the present and other authors. This is probably due to the combination of fuzzy membership, stochastic optimization and robust evaluation through ranking. Therefore it is not surprising that this algorithm has been used as the basis for improvements [27,28] as well as hardware implementations [29]. In the case of analog hardware implementations, other algorithms either perform worse, as we have reviewed, or imply very complex circuit structures. The Neural Gas seems the best choice in view of this trade-off, also because the sorting step can be simplified with little performance loss [30].

In a fuzzy perspective, it is more natural to define the relation "larger" among two (conventional) numbers as a degree to which one number is larger than another. We should mention that the problem of ranking fuzzy quantities has been reviewed for instance by Bortolan and Degani [31], and, more recently, by Wang and Kerre [32,33]. However, we are not dealing with fuzzy quantities, but with a *fuzzy evaluation* of crisp quantities. This approach is reasonable in very common situations such as presence of noise or other uncertainties in the measure of signals. In this case, two values which are very close cannot be reliably ranked, and a statement such as "a is larger than b" is more naturally expressed in fuzzy terms.

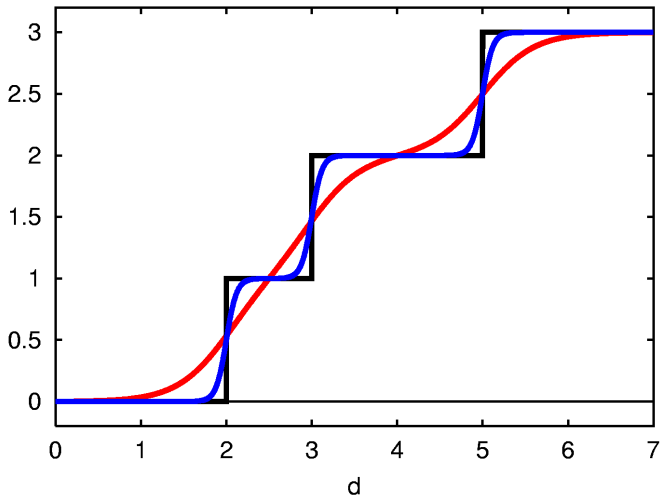

Fig. 1. Comparing crisp and fuzzy rank functions.

As a numerical illustration, suppose that we are to compare distances in two cases: (a) $d_1 = 3$ with $d_2 = 4$, and (b) $d_1 = 3$ with $d_2 = 3.01$. Clearly in both case (a) and case (b) we can rightfully say that $d_2 > d_1$, but it is also clear that in (a) this is "more true" than in (b). With a given level of uncertainty (for instance due to a given quantity of additive noise) it is also possible to quantify how much (a) or (b) are "true".

Therefore, we can make the following substitution:

$$\theta\left(d_j - d_i\right) \approx \frac{1}{1 + e^{(d_j - d_i)/\beta}} \tag{14}$$

and

$$\frac{1}{1 + e^{(d_j - d_i)/\beta}} \underset{\beta \to 0}{\longrightarrow} \theta\left(d_j - d_i\right) \tag{15}$$

so the computation of fuzzy rank can be expressed as

$$\rho_j = \sum_{i=1, j \neq i}^{n} \frac{1}{1 + e^{(d_j - d_i)/\beta}} \tag{16}$$

The parameter β here acts as a fuzzification parameter, such that for large β the ranking function is definitely fuzzy, while for $\beta = 0$ we obtain the original, crisp ranking function.

The two expressions (11) and (16) for the rank function $\rho(\cdot)$ are compared in a simple example, illustrated in Figure 1, where the following set of values is used: $\{d, 2, 3, 5\}$. The diagram is a plot of $\rho(d)$ (in the two expressions, crisp and fuzzy) for d in the range $[0, 7]$. Two plots are shown for the fuzzy expression, one for $\beta = 0.05$ and another for $\beta = 0.25$ (smoother).

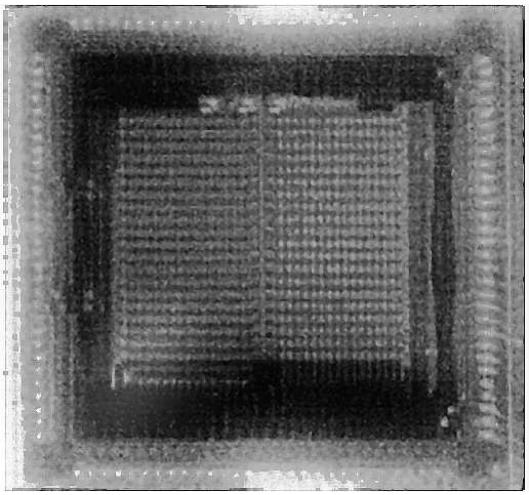

Fig. 2. An analog VLSI realization of a vector quantization encoder.

5.2 Implications of fuzzy ranking

Analyzing the diagram in Figure 1, we may gain insight on the effect of fuzzy ranks, as proposed above.

The main consequence of introducing $\beta > 0$ consists in points which are closer (in terms of distance) having converging rank values, with the limit for $\beta \longrightarrow 0$ being the average of the two corresponding crisp ranks.

Note that this limit reproduces the usual choice of rank statistics to resolve ties by averaging the ranks, so that, for instance, if we have two data points at ranks 3 and 4 with the same value, instead of assigning arbitrarily the two points, we should use the rank 3.5 for both. From this viewpoint, fuzzy ranking represents a generalization of the concept of *tie*, whereby with growing β points are more and more likely to be deemed equal in a fuzzy sense.

Conversely, the fuzzy ranking scheme does not have any effect of points which are sufficiently far away. When ranks are used *instead of original values* to exploit the robustness inherent in the rank operator, as for instance in rank correlation analysis, the introduction of fuzzy ranks does not influence points which are sufficiently far away, and only modifies ranks for points which are close to each other.

The fuzziness parameter β is related to the desired resolution, in that it establishes a soft form of thresholding for deciding whether points are close (fuzzy ties) or distant.

5.3 A note about hardware vector quantization

Since the authors have published some works about an implementation of Neural Gas in analog hardware, based on the VLSI chip shown in Figure 2, we comment about the implementation of the proposed fuzzy ranking function in the case of analog circuitry.

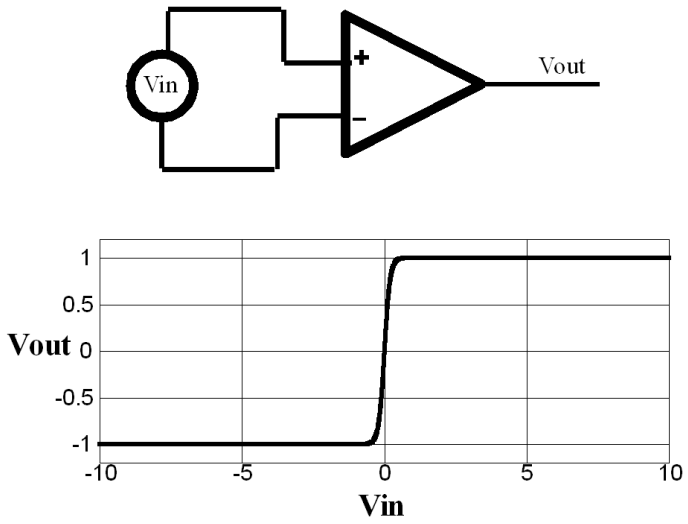

Fig. 3. A low-performance operational amplifier implements an approximate step function.

In analog hardware, when the functions implemented are non-ideal there can be a variable effect on training performances. In particular, the rank function (11) often uses the Heaviside step as a crisp distance comparison.

The step function in analog hardware is simply built by means of a saturating amplifier with large gain, which means typically an open-loop operational amplifier. However, Equation (11) has a c^2 space complexity, so circuit topologies should be made very inexpensive in terms of silicon area. This means very simple topologies (typically two stages). Consequently, the operational amplifier will feature a finite gain which implies a deviation from the ideal behavior.

The mid-frequency input-output relationship of an operational amplifier is a hyperbolic tangent saturating (approximately) at the $+$ and $-$ power voltages.

The fuzzy ranking function described is directly implemented by the op-amp-based circuitry outlined above. The fuzzification parameter is the inverse of the amplifier gain (the crisp and fuzzy version coincide for gain $\to \infty$ or for $\beta \to 0$). Therefore the fuzzy Neural Gas is simply a realistic model for the hardware implementation of the algorithm, without requiring any circuit modification.

6 Fuzzy ranks in image reconstruction

The fuzzy model for the ranking function makes it possible to enhance the reconstruction step in the process of image compression as described in Section 4. The procedure we are going to describe is similar to the one introduced in [34], whose main drawback was due to its crisp nature.

6.1 Multiple codevectors

When dealing with a limited codebook, it is often the case that blocks cannot be reconstructed with high accuracy due to the absence of a codevector with sufficiently good match, i.e., whose appearance is sufficiently similar to the block to be approximated.

In principle, as done for instance in Singular Vector Decomposition image reconstruction, this problem could be attenuated if several elementary patterns could be combined to obtain an output block as their average.

The "Multibest" technique [34] combines a standard vector quantization procedure with the principle of combining patterns. The outline of the technique is very simple: when approximating an image, we do not use only the best matching codevector, but the set of n best matching codevectors.

A combination of these patterns may then be obtained by interpolation of these n vectors. There are several possible techniques for performing interpolation, but those which are more feasible for their efficiency can be simply approximated with codevector averaging.

The main strength of this technique is the ability to represent image blocks which are considerably different from those used for training, thus making the overall procedure both more robust with respect to the image set used for codebook training, and more performing in terms of quality on new images (generalization). The improved quality can be assessed by objective measures, such as MSE or RMSE, and was experimentally observed by inspection.

A drawback of the method as originally presented is that codewords to be transmitted or stored to encode blocks are not simply composed of a single index in the codebook, but of all n indexes, thus compromising the compression performance of the technique. This problem is related to the fact that the per-block overhead is fixed and equal to the number of additional indexes required for reconstruction. The compression ratio in this case is

$$R_{\mathrm{MB}} = \frac{Nb_p}{nb_c}. \tag{17}$$

6.2 Fuzzy combinations of multiple codevectors

The use of fuzzy ranks in Neural Gas codebook design provides a technique to overcome the fixed overhead problem in the multiple codevector technique.

The procedure is as follows. Let $u_j(\mathbf{x})$ be the membership of pattern (image block) $\mathbf{x}$ to Voronoi polyhedron (codevector) $\mathbf{y}_j$. As we have seen, Neural Gas defines this quantity as

$$u_j(\mathbf{x}) = e^{-\rho_j/\lambda}. \tag{18}$$

Standard Neural Gas has fixed, integer values for ρ_j, so memberships have the same values for each pattern (these values are only permuted over codevectors). In contrast, Neural Gas with fuzzy ranks have real values for ρ_j, so that different situations may be obtained.

In particular, we define a threshold q so that *all codevectors with membership values over* q are used in reconstruction. The value of q is in the range [0,1]; it should usually be close to 1, and the lower the value, the larger the number of codevectors used in reconstruction.

For blocks for which there is a clear best match, the list of codevectors includes only the best match. However, if more than one codevector has similar membership (when no good match is found this will be the case), these are selected for reconstruction.

In this way, the number of codevectors in the multiple best-matching technique is variable rather than fixed, and compression performance will be improved. The actual improvement, however, cannot be estimated in advance since the number of codevectors needed for encoding any block is variable with the nature of the image and the codebook.

7 Experimental performance

7.1 Experimental setup

The fuzzy model for the Neural Gas has been tested by comparison with the standard version on some problems, with both artificial and real datasets:

(1) Centers-only (toy problem, very trivial): place three codevectors on three points. For initial "consistency checks".
(2) Centers-plus-noise (toy problem): place three codevectors on a set of points generated by a superposition of three Gaussians plus 60% random points.
(3) Lena (real dataset). Vector quantization of the standard benchmark image "Lena", shown in Figure 4, with codebooks of size 16 and 256.
(4) Four images (real datasets). Vector quantization of more benchmark images, shown in Figure 5.
(5) Detail quality in reconstruction with the proposed technique, shown in Figure 7.

The training of both algorithms was performed with identical initialization parameters (scheduling of updating coefficient and of range of influence of non-winners) and starting values (prototypes at first iteration).

7.2 Experimental results

The first problem was used to ensure that the training steps were not too different, to validate the software (written in C). For $\beta = 0$ the two algorithms are indeed identical.

Fig. 4. The “Lena” image.

Fig. 5. Four benchmark images (from `http://links.uwaterloo.ca/bragzone.base.html`).

The second problem highlighted that, for low values of β, there are no significant differences in performance between the two algorithms. In some experiments the fuzzy version outperformed the standard version, but this is not a typical behavior.

The training on the Lena image was a test of these outcomes on a real problem. In Figure 6 is shown a typical training trace (mean square error versus training steps), put on a logarithmic scale to compensate for the (approximately) logarithmic decreasing in convergence with time. This shows that the two traces are different, but converge to the same solution. The thin trace is standard Neural Gas and the thick trace (only slightly different

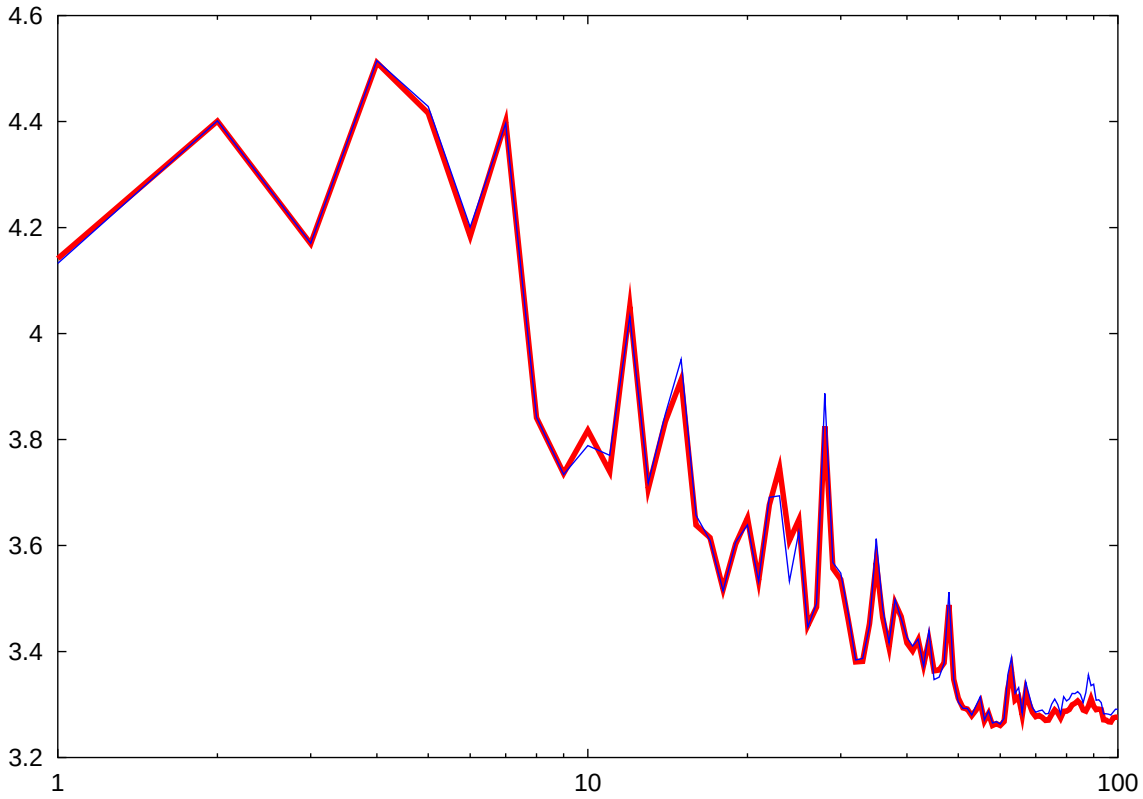

Fig. 6. Trace of mean square error during training on the "Lena" image. (Note that, for ease of visualization, the x-axis is logarithmic and the y-axis does not start at 0.)

in some locations) is the fuzzy modification.

The four additional images, which are greyscale and are of size 256x256, were used to confirm the previous results. The images were obtained from the "Waterloo Repertoire", available online at `http://links.uwaterloo.ca/bragzone.base.html`. Codevector sizes used are 16, 64 and 256. Results on the concordance of the two methods are outlined in Table 7.2. For each test, the maximum deviation of the fuzzy version over the standard version (in percentage of RMS error) is indicated. The final codebooks have always been found to be equal according to the following definition. Two codebooks A and B are considered equal if, for any codevector in codebook A, the closest codevector in codebook B is within a preselected distance threshold. This threshold has to be selected case by case, taking into account codebook cardinality and making it less than the minimum distance between two codevectors of any codebook.

The remarkable fact that final codebooks were always coincident is a confirmation of the good properties of the Neural Gas algorithm, which proves to be stable under the perturbations induced by reasonable values of β.

Finally, a quality verification on the "Lena" image was performed to compare the result of vector quantization compression and multiple codevector reconstruction on a complex detail (Figure 7, left). Neural Gas training was performed 10 times and the best cross-validation result was picked, for both the standard and the fuzzy-rank versions of the algorithm. A test set was obtained by extracting a small percentage of the image blocks from the training set (these blocks were deleted from the training set and did not take part

Table 1
Verification of convergence between crisp and fuzzy rankings in Neural Gas training

Test	Max. discordance in RMS Error
goldhill 16	0.8%
goldhill 64	0.9%
goldhill 256	1.3%
bridge 16	0.0%
bridge 64	0.5%
bridge 256	0.5%
bird 16	0.2%
bird 64	1.0%
bird 256	2.1%
camera 16	1.7%
camera 64	1.7%
camera 256	1.1%

in Neural Gas training).

Since a 8×8 block size was selected, and the image is 512×512 pixels, 4096 blocks were obtained, of which 4000 were in the training set and 96 in the test set. These blocks were selected randomly. This very small percentage was decided because of the small dataset size.

The value of β was also selected by evaluating cross-validation RMSE on this training/test set split, whereas q was arbitrarily set at 0.9. The codebook size was set at 32, which is a small value, to obtain lower quality images for which the improvement could be more easily appreciated.

The result can be observed in Figure 7. The left image is the original detail, an eye from the "Lena" image. The center image is the result of standard Neural Gas training. The right image is the result obtained with fuzzy ranks and multiple output vectors.

The main feature appearing from this demonstration is that the proposed technique is able to create more varied patterns with respect to standard, one-codevector version. Although this does not imply that the reconstruction is very faithful, both RMSE and visual inspection confirm the improved quality obtained with the technique.

7.3 Choice of parameters β *and* q

The acceptable value of β depends linearly on the difference between distances that has to be resolved. A method to select the acceptable β can be based on the distribution of the

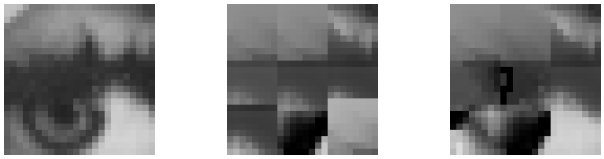

Fig. 7. Detail from the “Lena” image: left, original; center, standard VQ reconstruction; right, multiple-codevector training and reconstruction.

distance differences in the training set. The Δ_{ij}'s are tabulated and sorted; then a given tolerance is selected (for instance, 1%) and the corresponding quantile is identified on the table. This corresponds to a given value of Δ and therefore to the required acceptable value of β.

However, to relate the parameter value to actual performance, in general it is probably better to select it by cross-validation or other empirical procedures based on measuring the actual reconstruction performance. In the image reconstruction experiments the value of β has been assessed with this technique.

The value of q should be left up to the user, since it represents the degree of freedom which is always present in lossy compression techniques. It should be viewed as a “knob” to be turned for tuning the quality/compression tradeoff to suit the user's needs.

8 Conclusion

In this paper we have reviewed some uses of fuzzy concepts in vector quantization training. We have presented the novel concept of fuzzy ranks and applied it to Neural Gas codebook design algorithm. The experiments presented shows that, for reasonably chosen uncertainty levels (β), the Neural Gas algorithm is remarkably stable.

The technique has been applied to image reconstruction with a multiple codevector strategy, whereby codevectors to be used are selected according to a threshold over rank values. This produces a variable overhead strategy for which compression is not easy to assess in advance, but quality of reconstruction is improved.

Acknowledgments

This work was funded by the Italian Ministry of Education, University and Research under a “PRIN 2004” grant.

References

[1] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, New Jersey, USA: Prentice Hall, 1988.

[2] A. Gersho, "On the structure of vector quantizers," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 157–166, March 1982.

[3] R. Gray, "Vector quantization," *IEEE Acoustic, Speech and Signal Processing Magazine*, vol. 1, pp. 4–29, 1984.

[4] S. Rovetta and R. Zunino, "Vector quantization for license plate location and image coding," *IEEE Transactions on Industrial Electronics*, vol. 47, no. 1, pp. 159–167, February 2000.

[5] S. Ridella, S. Rovetta, and R. Zunino, "K-winner machines for pattern classification," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 371–385, March 2001.

[6] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–565, 1948.

[7] T. Martinetz, S. Berkovich, and K. Schulten, "'Neural gas' network for vector quantization and its application to time-series prediction," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 558–569, 1993.

[8] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizers design," *IEEE Transactions on Communications*, vol. COM-28, pp. 84–95, January 1980.

[9] J. Moody and C. Darken, "Fast adaptive k-means clustering: Some empirical results," in *Proceedings of the International Joint Conference on Neural Networks, IJCNN90*, vol. II. IEEE Neural Networks Council, 1990, pp. 233–238.

[10] C. D. Kirkpatrick, S. and Gelatt and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[11] A. Corana, M. Marchesi, C. Martini, and S. Ridella, "Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm," *ACM Transactions on Mathematical Software*, vol. 13, no. 3, pp. 262–280, 1987.

[12] F. Aurenhammer, "Voronoi diagrams-a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[13] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*. New York: Plenum, 1981.

[14] A. Baraldi and P. Blonda, "A survey of fuzzy clustering algorithms for pattern recognition. I," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 29, pp. 778–785, 1999.

[15] ——, "A survey of fuzzy clustering algorithms for pattern recognition. II," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 29, pp. 786–801, 1999.

[16] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982.

[17] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. L. Cam and J. Neyman, Eds., vol. I. University of California, January 1967, pp. 281–297.

[18] G. Geman and D. Geman, "Stochastic relaxation, gibbs distribution and bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721–741, 1984.

[19] H. Ritter, T. Martinetz, and K. Schulten, *Neuronale Netze*. München, Germany: Addison-Wesley, 1991.

[20] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, vol. 3, pp. 32–57, 1974.

[21] K. Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," *Proceedings of IEEE*, vol. 86, no. 11, pp. 2210–2239, November 1998.

[22] R. Krishnapuram and J. M. Keller, "A possibilistic approach to clustering," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 2, pp. 98–110, May 1993.

[23] O. Nasraoui and R. Krishnapuram, "A robust estimator based on density and scale optimizations and its application to clustering," in *FUZZIEEE96: Proceedings of the International Conference on Fuzzy Systems*. IEEE, New Orleans, USA, 1996, pp. 1031–1035.

[24] D. Cohn, E. A. Riskin, and R. Ladner, "Theory and practice of vector quantizers trained on small training sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 54–65, January 1994.

[25] T. Kohonen, *Self-Organizing Maps*. Springer, 2001.

[26] S. Ridella, S. Rovetta, and R. Zunino, "IAVQ–interval-arithmetic vector quantization for image compression," *IEEE Transactions on Circuits and Systems, Part II*, vol. 47, no. 12, pp. 1378–1390, December 2000.

[27] T. Hoffmann and J. M. Buhmann, "An annealed neural gas network for robust vector quantization," in *Proceedings of the International Conference on Artificial Neural Networks – ICANN96, Bochum, Germany*, 1996, pp. 151–156.

[28] S. Ridella, S. Rovetta, and R. Zunino, "Plastic algorithm for adaptive vector quantization," *Neural Computing and Applications*, vol. 7, no. 1, pp. 37–51, 1998.

[29] S. Rovetta and R. Zunino, "Efficient training of vector quantizers with analog circuit implementation," *IEEE Transactions on Circuits and Systems, Part II*, vol. 46, no. 6, pp. 688–698, June 1999.

[30] F. Ancona, S. Ridella, S. Rovetta, and R. Zunino, "On the importance of sorting in 'neural gas' training of vector quantizers," in *Proceedings of the 1997 International Conference on Neural Networks, Houston, USA*, June 1997, pp. 1804–1808.

[31] G. Bortolan and R. Degani, "A review of some methods for ranking fuzzy sets," *Fuzzy Sets and Systems*, vol. 15, pp. 1–19, 1985.

[32] W. Wang and E. Kerre, "Reasonable properties for the ordering of fuzzy quantities (I)," *Fuzzy Sets and Systems*, vol. 118, pp. 375–385, 2001.

[33] ——, “Reasonable properties for the ordering of fuzzy quantities (II),” *Fuzzy Sets and Systems*, vol. 118, pp. 386–405, 2001.

[34] S. Carrato, F. Passaggio, S. Rovetta, and R. Zunino, “Interpolation approaches to vector quantization for image compression,” in *NNSP–96, Proceedings of the 6th IEEE Workshop on Neural Networks for Signal Processing*, September 1996.