Fuzzy concepts in vector quantization training

Francesco Masulli
Department of Computer Science,
University of Pisa, Italy,
INFM - National Institute for the Physics of Matter
E-mail: masulli@di.unipi.it

Stefano Rovetta
Department of Computer and Information Sciences,
University of Genoa, Italy,
INFM - National Institute for the Physics of Matter
E-mail: rovetta@disi.unige.it

Abstract

Vector quantization and clustering are two different problems for which similar techniques are used. 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 the performance of these algorithms, and propose an alternative way to use fuzzy concepts as a modeling tool for physical vector quantization systems, Neural Gas with a fuzzy rank function.
Keywords: Vector quantization; Neural Gas; Fuzzy clustering.

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*.

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: Rate/Distortion or Distortion/Rate approaches.)

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" [4].

2 Clustering, vector quantization, and fuzzy concepts

We have outlined some differences between clustering and vector quantization, and yet the synthesis of a codebook for vector quantization is often approached with algorithms derived from *c*-Means (a standard clustering technique). One usual difference is that, since vector quantization is typically adopted for large-dimensional, large-sized training sets, minimization is performed by stochastic gradient descent (*online* training) rather than by deterministic (or *batch*) algorithms. This is because the curse of local minima is worsened by the large dimensionality and codebook size.

Several clustering algorithms have been modified in the direction of incorporating fuzzy concepts (starting with the Fuzzy *c*-Means algorithm [5]). A review of fuzzy concepts in clustering is provided in [6, 7].

The relationship between the clustering and quantization problems is of a geometrical nature. In both cases, the input space is partitioned by a *Voronoi tessellation* [8], representing regions of data sharing similar properties by means of a single reference point or site or (in vector quantization jargon) codevector. However, in clustering, data points belonging in a single region should be the largest group of data that can

be reasonably gathered in a single cluster (clusters should be as few as possible to enable understanding the structure of data), whereas in vector quantization points in a region must be so similar that the approximation error obtained by substituting data with codevectors is negligible (thus codevectors should be as many as possible, within the resource limits imposed by the overall system design).

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. 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 (training)

In the following we review how typical algorithms for the synthesis 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 $\{\mathbf{y}_1, \dots \mathbf{y}_c\}$ of c reference points. The distorsion assumed is the Euclidean distance $d_i = ||\mathbf{x} - \mathbf{y}_i||$.

The classical approach is Lloyd/MacQueen's method [9][10][11], the standard c-Means clustering procedure; MacQueen's method is the online version. The k-th input vector is attributed to the Voronoi polyhedron defined by reference vector $\mathbf{y}_i$) if $u_{ik} = 1$, where u_{ik} is a crisp membership value which is 1 if $d_{ik} = \min\{d_{1k}, \dots, d_{Nk}\}$ and 0 for all other reference vectors. The closest reference vector for a data point wil be called the "winner" for that point. The updating rule is:

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

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). The on-line version transforms the Picard iteration of the standard version, in which at each step a necessary minimum condition is satisfied, 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:

$$\mathbf{y}_i^{(t+1)} = \mathbf{y}_i^{(t)} + \eta^{(t)} u_{ik} (\mathbf{x}_k - \mathbf{y}_i) \tag{2}$$

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, but 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 [12]. MacQueen [10] adopts an individual coefficient for every reference vector, equal to $1/t_i$ where t_i is the number of updates for reference vector $\mathbf{y}_i$ so far, thus retaining the exact equivalence between the online and batch versions of c-means. Ritter *et al.* [13] propose instead an exponential decay rate $\eta^{(t)} = \eta_\mathrm{i} (\eta_\mathrm{f}/\eta_\mathrm{i})^{t/t_\mathrm{max}}$ from η_i to η_f in t_max steps. This law has been used also in the Neural Gas algorithm.

The maximum entropy approach of the Deterministic Annealing technique by Rose [14] 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:

$$u_i = \frac{e^{-d_i/\beta}}{\sum_{j=1}^{c} e^{-d_j/\beta}} \tag{3}$$

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 deterministic 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.

The Neural Gas algorithm by Martinetz *et al.* [4] 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, but rather as 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:

$$\rho_i = \sum_{j=1}^{c} \theta(d_i - d_j) \tag{4}$$

$\theta(x)$ 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-order statistics. 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:

$$u(\mathbf{x}) = e^{-\rho_i/\lambda} \tag{5}$$

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.

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.

4 A fuzzy model of the ranking function

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 [15] as well as hardware implementations [16]. 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.

In analog hardware, when the functions implemented are non-ideal there can be a variable effect on training performances. In particular, the rank function (4) 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 (4) has a c^2 space complexity, so circuit topologies should be made very inexpensive in terms of silicon area. Consequently, the operational amplifier will feature a finite gain which implies a deviation from the ideal behavior.

The input-output relationship of an operational amplifier at middle frequencies is a hyperbolic tangent saturating (about) at the $+$ and $-$ power voltages. This suggests a very natural fuzzy model for the non-ideal rank function.

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 [17]. However, we are not dealing with fuzzy quantities, but with a *fuzzy evaluation* of crisp quantities.

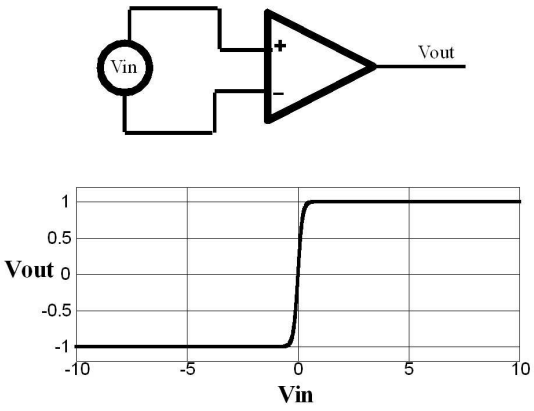

Figure 1: A low-performance operational amplifier implements an approximate step function.

For instance, suppose that we are to compare (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).

Therefore, we can make the following substitution:

$$\theta(d_j - d_i) \approx \frac{1}{1+e^{(d_j-d_i)/\beta}} \qquad \text{and} \qquad \frac{1}{1+e^{(d_j-d_i)/\beta}} \underset{\beta\to 0}{\longrightarrow} \theta(d_j - d_i) \tag{6}$$

so the computation of fuzzy rank can be expressed as

$$\rho_j = \sum_{i=1, j\neq i}^{n} \frac{1}{1+e^{(d_j-d_i)/\beta}} \tag{7}$$

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 (4) and (7) for the rank function $\rho(\cdot)$ are compared in a simple example, illustrated in Figure 2, 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).

The fuzzy ranking function 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 a realistic model for the hardware implementation of the algorithm.

5 Experimental performance

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

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.

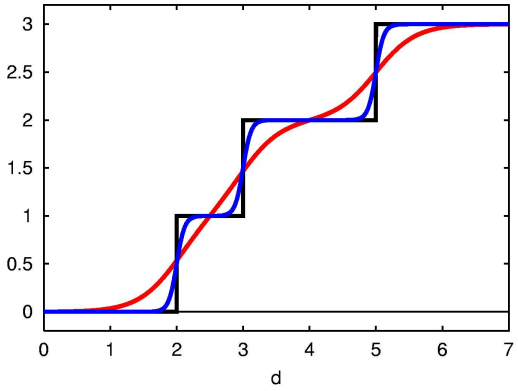

Figure 2: Comparing crisp and fuzzy rank functions.

Figure 3: The "Lena" image.

3. Lena (real problem). Vector quantization of the standard benchmark image "Lena", shown in Figure 3, with codebooks of size 16 and 256.

4. Four images (real problem). Vector quantization of more benchmark images, shown in Figure 4.

The training of both algorithms was performed with identical initialization parameters (for the scheduling of updating coefficient and of range of influence of non-winners,

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.

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 5 is shown a typical training trace (mean square error versus training steps), put on a logarithmic scale to show 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 in some locations) is the fuzzy modification.

Figure 4: Four benchmark images (from `http://links.uwaterloo.ca/bragzone.base.html`).

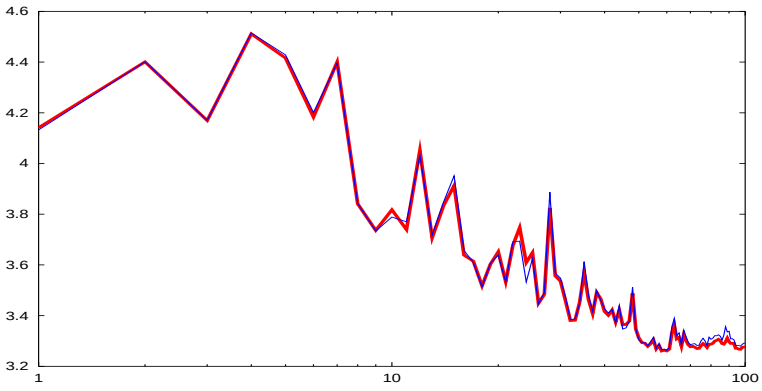

Figure 5: Trace of mean square error during training on the "Lena" image.

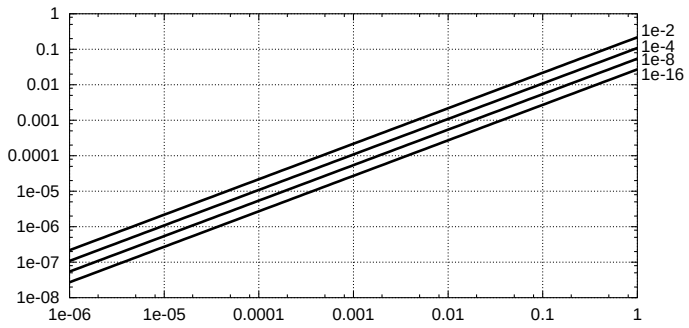

Figure 6: Nomogram for the calculation of β.

Finally, the four additional images (greyscale 256x256) were used to confirm the previous results. Codevector sizes used are 16, 64 and 256. Results on the concordance of the two methods are outlined in the following table. 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.

Test	Max. discordance in RMS
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%

The acceptable value of β depends linearly on the difference between distances that has to be resolved. The nomogram in Figure 6 is a plot of β versus $d_i - d_j$ for different values of the accepted error (annotated on the right margin).

6 Conclusion

In this paper we have reviewed some uses of fuzzy concepts in vector quantization training. The main application is to enhance convergence, but we have also proposed using fuzzy ranking for the modeling of hardware implementations.

Acknowledgments

This work was funded by the Italian National Institute for the Physics of Matter (INFM) and by the Italian Ministry of Education, University and Research under a "PRIN - Cofin2002" grant.

References

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

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

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

[4] T.M. Martinetz, S.G. Berkovich, and K.J. 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.

[5] James C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*, Plenum, New York, 1981.

[6] 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.

[7] A. Baraldi and P. Blonda, "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.

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

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

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

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

[12] 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.

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

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

[15] 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.

[16] Stefano Rovetta and Rodolfo 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.

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