On the Importance of Sorting in "Neural Gas" Training of Vector Quantizers

Fabio Ancona, Sandro Ridella, Stefano Rovetta, and Rodolfo Zunino

DIBE - Department of Biophysical and Electronic Engineering - University of Genoa
Via all'Opera Pia 11a - 16145 Genova - Italy - Email: zunino @ dibe.unige.it

Abstract— **The paper considers the role of the sorting process in the well-known "Neural Gas" model for Vector Quantization. Theoretical derivations and experimental evidence show that complete sorting is not required for effective training, since limiting the sorted list to even a few top units performs effectively. This property has a significant impact on the implementation of the overall neural model at the local level.**

1. Introduction

Vector Quantization (VQ) represents an important paradigm for information representation from both a theoretical point of view [1] and an applicative perspective [2]. This is mainly due to the possibility of easy interpretation of training outcomes and to the remarkable performances attained by VQ-based compression.

The "Neural Gas" (NG) model [3] provides an effective algorithm for adaptive training of vector quantizers by a simple principle of operation: neurons are iteratively adjusted, each unit moves proportionally to its similarity to training samples. As compared to k-means clustering [4], NG seems to be less subject to getting trapped in local minima. Theory proves [3] that the iterative NG algorithm implements a stochastic gradient descent of an analytical cost function, as opposed to Kohonen's Self-Organizing Maps [5], for which no such property has been established. Moreover, NG involves a matching-based rather than a topology-driven ordering of neurons: although a dynamic version of NG has been proposed for flexible-topology networks [6,7], the basic NG model does not assume any interconnection schema among neuron units. This simplifies hardware implementations, and motivates the interest in the NG algorithm, as the research presented here aims to a hardware support of VQ training: both a digital [8] and an analog [9] approach yield a "VQ-coprocessor" for reducing huge computational training loads.

Most of the neural model can be supported at the local level, that is, each neuron operates independently of the others; network-wide inter-neuron communications only take place in ordering the matching scores of neurons. The analysis presented here addresses the relevance of sorting to the algorithm's performance, with a twofold goal: from a a theoretical point of view, the investigation clarifies important features of the basic training model; from a practical perspective, simplifying sorting without affecting overall performance can reduce computational costs dramatically and facilitate hardware realizations, as well. A formal analysis first characterizes the sorting process by measuring the relevance of each position in the list. Relative importances decrease monotonically from "top" positions in the list to "bottom" ones: in other words, identifying the list head accurately is more important than the tail. Moreover, as a result of the annealing process, the gap in importance between non-top list positions progressively shrinks during training, and eventually only the best-matching neuron is really important.

Complete, exact sorting is a basic condition for the validity of the original NG theoretical model; nevertheless, the present analysis shows that *in practical implementations* complete sorting may be avoided, and determining only a few "top" positions in the neuron list is sufficient. The practical impact of such derivations is greatest on large codebooks, since detecting the best k-out-of-N units is simpler than sorting a long list completely; this is true both in software (by cutting computational costs) and in hardware, where the circuitry for exhaustive sorting may be quite complex [10-12]. When considering that NG optimization supports a fuzzy clustering strategy [3], partial sorting can be regarded as implementing a sort of constrained version of k-means clustering, in which only k units are active at any iterations.

A set of experimental results support theoretical predictions. Experiments include tests on the synthetic data set used by Fritzke [6] to demonstrate VQ algorithms, and tests on a real application involving VQ-based image compression [2,13]. Experiments aim to assess the importance of the sorting process quantitatively, by measur-

0-7803-4122-8/97 $10.00©1997 IEEE

ing the effect of incomplete or incorrect sorting on the algorithm's distortion performance; the statistical validation of obtained results required a massive amount of independent runs. Experimental evidence yields two conclusions: first, partial, exact sorting performs better than complete but noisy sorting; second, even a few units in partial sorting is sufficient to attain a final distortion equivalent to that attained by ideal NG.

2. Sorting in "Neural Gas" training

"Neural Gas" iteratively adjusts neurons' positions according to their similarities to input patterns. In order to coordinate adjustments without a fixed topological schema, the training process uses Euclidean distances from training samples for sorting neurons. A nonlinear time-controlled scheduling weights the reward for each neuron's ranking; thus training evolves from a distributed-activation pattern, in which several neurons are adjusted at the same time, to a truly Winner-Take-All (WTA) schema, in which only one neuron is affected by a training sample. Theory [3] shows that the asymptotical positions of neurons in the data space span a uniform coverage over data. In the following, a set of neurons $\{\mathbf{w}_i,\ i=1,...,N\}$ are positioned in a d-dimensional domain space, X; the terms 'neuron' and 'codevector' will be used as synonyms. The NG algorithm can be outlined as follows:

For iterations t =0 to T :

1. *Draw* at random a training sample, $\mathbf{x} \in X$;
2. $\forall$ neuron, w_i, *compute* the Euclidean distance:
$$d_i = \|\mathbf{w}_i - \mathbf{x}\| \quad \forall i; \tag{1}$$
3. *Sort* neurons in order of increasing $d(i)$; let $k(i) = 0,...N$ denote the rank of the i-th neuron in the sorted list;
4. *Adjust* neurons according to their positions in the sorted list:
$$\mathbf{w}_i^{(t+1)} = \mathbf{w}_i^{(t)} + \eta(t) \cdot h[k(i),t] \cdot \left(\mathbf{w}_i^{(t)} - \mathbf{x}\right) \quad \forall i \tag{2}$$

The rewarding function, $h()$, balances the distribution of weight updates during training and drives the training process. An implementation of $h()$ is given in [3], and the overall scheduling functions can be written as:

(3)

$$\eta(t) = \eta_0 \left(\frac{\eta_T}{\eta_0}\right)^{\frac{t}{T}}; \quad \lambda(t) = \lambda_0 \left(\frac{\lambda_T}{\lambda_0}\right)^{\frac{t}{T}}$$

$$h[k(i),t] = \exp\left(-\frac{k(i)}{\lambda(t)}\right);$$

Due to the satisfactory results obtained, implementation (3) will be adopted as a default throughout the paper. The settings for the annealing schedule — $\lambda_0 = 10$, $\lambda_T = 0.01$, $\eta_0 = 0.65$, $\eta_T = 0.05$ — proved effective in all experiments. NG as a training algorithm is interesting from several viewpoints. As to distortion performance, empirical evidence (reported in [3] and verified in the present research) indicates that, on average, NG yields smaller distortion than other algorithms; in addition, overall convergence typically requires a smaller number of iterations (training samples). As to implementation, the model's features meet hardware-related requirements very well, especially when considering that the neural model does not involve any topological structure. Most of a neuron's activity performs independently of the other units. As distance computations (1) and weight adjustments (2) only involve local information, the bulk of the computational load can be supported by neuron-embedded circuitry [9].

Sorting is the only process requiring inter-neuron circulation of information for evaluating list positions $k(i)$; in a full impementation of NG, complete, exact sorting demands network-wide communications. In training huge networks for a real application (e.g., image compression), this may prove very expensive in terms of hardware connectivity and layout design [10-12]. Thus one might wonder how accurate and complete the sorting process must be to ensure effective training; the overall goal is to reduce the information flow through the network.

Two basic questions arise: the first concerns sorting *accuracy*: can each unit be allowed just to estimate its own list ranking by using limited information? Conversely, the second question concerns sorting *completeness*: can proper training be attained by identifying (exactly) a subset of the total sorted list of neurons? Both problems lead o an analysis of the sensitivity of the training process to sorting errors. Weight update (2) evolves as: $\varepsilon(k,t) = \eta(t) \cdot h(k,t)$ where the neuron index, i, is omitted for simplicity. "Sensitivity" will be expressed by :

$$S(k,t) = \frac{\partial}{\partial k}\varepsilon(k,t) = -\frac{\eta_0}{\lambda_0}\left(\frac{\eta_T \lambda_0}{\eta_0 \lambda_T}\right)^{\frac{t}{T}} \exp\left(\frac{-k}{\lambda(t)}\right) \tag{4}$$

Thus sensitivity measures how much a variation from the k-th list position at time t affects the weight update of the associated neuron. Normalized sensitivity, a trivial simplification of (4), better fits practical purposes:

$$S_n(k,t) = \left(\frac{\eta_T \lambda_0}{\eta_0 \lambda_T}\right)^{\frac{t}{T}} \cdot \exp\left(-\frac{k}{\lambda(t)}\right) \tag{5}$$

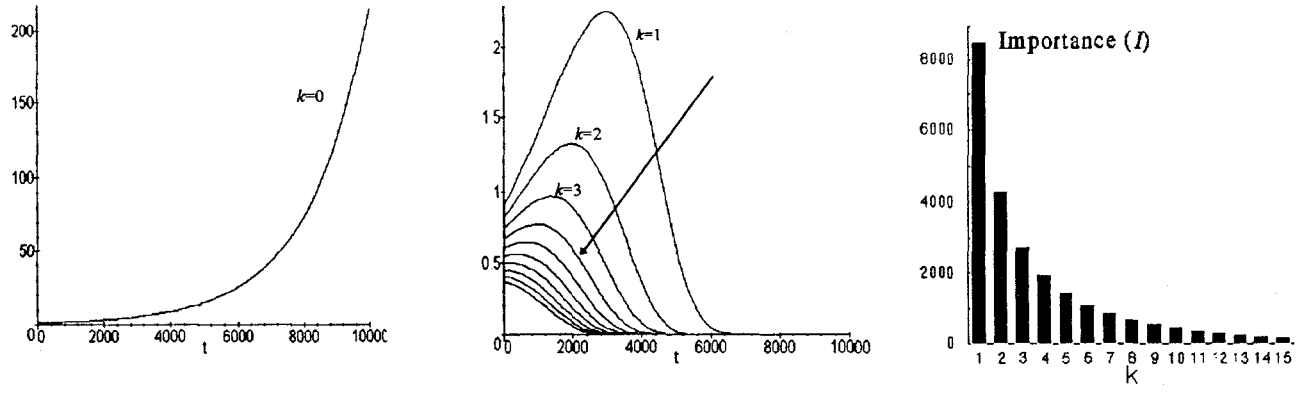


Fig.1 - Sensitivity analysis for different positions in the sorted list of neurons
a) - Normalized sensitivity at top position, k=0
b) - Normalized sensitivity at general positions, $k \geq 1$
c) Overall Importance at general positions, $k \geq 1$

Analyzing the family of curves (5) parameterized by k points out interesting features of the training progress.

First of all, the graphs in Fig.1a and Fig.1b show that normalized sensitivity for the top position in the list (the best-matching neuron with k=0) is substantially different from all others (note the different scale on the graphs).

Second, the importance of the "winning" unit increases while training proceeds, whereas the importances of all other positions decrease. This is analytically derived by verifiying that:

$$S_n(k,T) = \frac{\eta_T \lambda_0}{\eta_0 \lambda_T} \cdot \exp\left(-k/\lambda_T\right); \qquad (6)$$

$$\lim_{t\to\infty} S_n(0,t) = \infty; \qquad \lim_{t\to\infty} S_n(k,t) = 0 \ \ \forall k \geq 1$$

The limits (6) are well approximated in practice; with the parameter settings above cited (e.g., λ_T = 0.01), final sensitivity $S_n(k,T)$ is very large when k=0 and virtually vanishes even for small positive values of k.

Finally, comparing curves shows that sensitivity decreases from higher to lower ranks (i.e., $k' < k'' \Rightarrow S_n(k',t) > S_n(k'',t)$). A quantitative "overall importance", $I(k)$, of the k-th list position is given by:

$$I(k) = \int_0^T S_n(k,t)dt \qquad (7)$$

Importances (7) give relative weights of the various list positions throughout the entire training process. Integrals can be computed numerically: results for the first 15 positive values of k are reported in Fig.1c; the importance value for k=0 far exceeds any other (by at least 50 times) and has not been included. Relative importance clearly drops to insignificant values even for small values of k. This strongly supports partial sorting.

To sum up, the theoretical analysis of the sorting process in NG training lead to a few basic conclusions:

- Correct identification of the best-matching neuron becomes more and more important while training proceeds; conversely, sensitivity for non-top positions tends to a null value during training.
- Sensitivity is not uniformly distributed among list positions. More precisely, it decreases when k increases; however, the gaps among different curves shrink when k increases.
- The specific importances of positions in the list (weighted over the entire training process) indicate that a few top positions convey most of the overall importance; partial sorting is acceptable in practice.

3. Experimental results

This sections discusses practical methods to take maximum advantage of the conclusions drawn by the above theoretical analysis. In particular, two crucial issues are considered: 1) if partial sorting is supported, how many positions in the list must be computed exactly to attain an acceptable distortion? 2) what is the best processing strategy for the positions not included in the partal list? Due to the complexity of a theoretical analysis to this purpose, an empirical approach has been adopted and different domains have been considered to validate conclusions. The procedure used in all tests involves two experimental set-ups:

1. a few (top) list positions are computed exactly and the corresponding neurons adjusted accordingly; uniform noise affects the sorting process for other neurons, whose list positions are somehow offset. In the simulations, noise injection is controlled by how much a rank can be misplaced.

2. a few (top) list positions are computed exactly and the corresponding neurons adjusted accordingly; all other neurons remain unaffected.

3.1. Tests on synthetic data

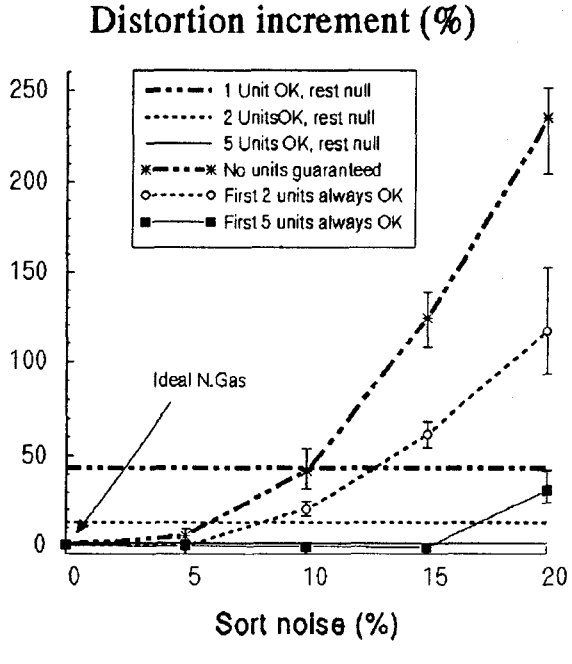
This two-dimensional synthetic domain was proposed by Fritzke to demonstrate VQ training algorithms visually [6,7]. In such tests, the number of neurons has been set to 30 after estimation with the Ying-Yang criterion [14]; the training set includes about 4,500 samples. Noise injection for inaccurate-sorting tests has been simulated by allowing each list position to randomly deviate from its original value by at most 5%, 10%, 15% and 20%, respectively. The algorithm's performance has been measured by its eventual distortion (mean square error) on training data; for each noise percentage, 40 independent training processes have been run in order to define statistical confidence intervals. Results on this testbed are presented in Fig.2a; the graph plots increments in distortion versus noise injection under experimental conditions A). Horizontal lines indicate the performances for experimental conditions B), which are noise-free.

Experimental evidence proves the effectiveness of partial sorting. In noisy sorting [exp. A)], performance degrades significantly when reducing from 5 to 1 exact list positions; confidence intervals enlarge accordingly. More importantly, the graph points out that disregarding subsequent list positions [exp. B)] is greatly preferable to noisy sorting. By determining just five positions distortions are equivalent to the ideal algorithm's performance.

3.2. Tests on image-compression data

The application of the presented analysis to VQ-based image compression [13] confirms and supports the results obtained on synthetic data. In this testbed, each gray-level 8bpp image includes 512x512 pixels and is split into 4,096 elementary square blocks; each block includes 8x8 pixels and represents a sample. In all tests, the codebook holds 256 neurons: the codebook size was determined by generalization-based criteria [15].

The experiments adopted the same procedure followed for synthetic data; again, sorting affected by increasing noise percentages is compared with partial exact sorting; distortion results are presented in Fig.2b. The differences between eventual distortions are less marked than those shown in Fig.2a; in other words, taking into account 1, 2, or 5 units, respectively, yields similar results. This phenomenon is due to the fat that the underlying data distribution is quite complex, and "natural" clusters are not well segregated: as a consequence, a quasi-uniform distribution of neurons in the data space makes sorting accu-



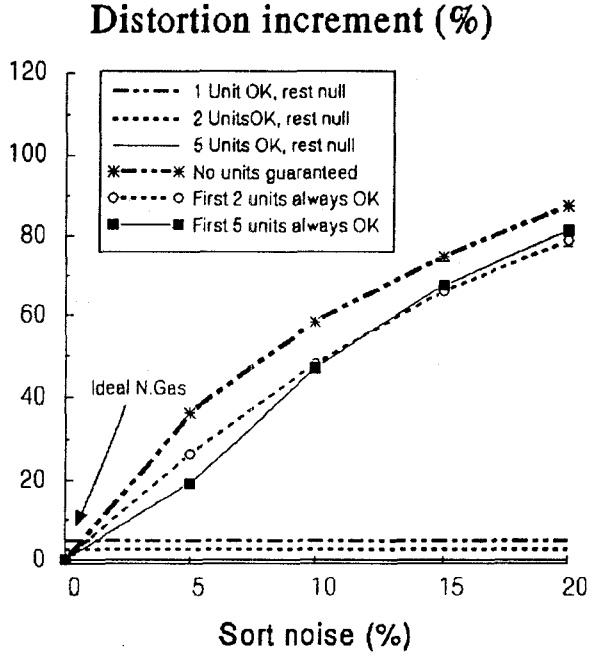


Fig.2 - Comparison results between noisy and partial sorting
Fig.2a - Fritzke's synthetic data **Fig.2b - Image compression**

racy less important. Nevertheless, experimental evidence still supports the advantage of using partial sorting, since the straight horizontal lines for noiseless partial sorting [exp. B)] indicate smaller distortions than their counterparts involving noisy sorting [exp.A)]. Indeed, the overall performance of partial sorting approximates the quality attained by ideal NG very closely; this result is even more relevant when considering that it is obtained in a complex, real-world experimental domain.

4. Concluding Remarks

Studying the role of sorting in a VQ training algorithm aims to simplify implementation issues by reducing the exchange of information throughout a network. This is especially attractive in the NG model, whose nature inherently stresses a local-level approach, since neurons operate independently of one another in the crucial, computation-intensive steps of the training process. The paper considered the sorting subprocess in NG from a formal perspective, by deriving analytical expressions for the importances of specific positions in the list of neurons.

The strongest conclusion is that the best-matching unit deserves some special attention and requires accurate evaluation - this is indirectly confirmed by the existence of algorithms using only "winning" nodes. When dealing with the specific NG model, the analysis indicates the validity of partial sorting, i.e., not all positions in the list need exact evaluation. The accuracy and completeness issues in sorting have also been addressed from an experimental perspective, involving both synthetic and real-domain data. Empirical evidence supports theoretical predictions, and indicates that a few (typ. five-ten) "top" positions in the list are sufficient to attain almost ideal results in terms of distortion. Such results open views over current research about simplified circuitry for partial sorting, which appears simpler and more effective than classical hardware architectures for exhaustive sorting.

5. References

[1] *IEEE Trans. Inf.Theory*, Spec.Issue on Vector Quantization, March 1982, vol.IT-28, No.2

[2] Nasrabadi N, King R "Image Coding Using Vector Quantization:A Review", *IEEE Trans. Commun.*, Aug. 1988, pp. 957-971

[3] Martinetz TM, Berkovich SG, Schulten KJ " "Neural Gas" network for vector quantization and its application to time-series prediction", *IEEE Trans. on Neur.Net.*, 1993, vol.4, No.4, pp.558-569

[4] Linde Y, Buzo A, Gray RM "An Algorithm for vector quantizater design", *IEEE Trans. Commun.*, Vol. COM-28, pp. 84-95, Jan. 1980

[5] Kohonen T, *Self-organization and Associative Memory 3rd Ed.*, 1989, Springer Verlag

[6] Fritzke B, "A growing Neural Gas network learns topologies", in Tesauro G, Touretzky DS, and Leen TK (Eds.), *Advances in Neural Information Proc. Sys. 7 NIPS-7*, MIT Press Cambridge MA, 1995

[7] Fritzke B, "Growing Grid - a self-organizing network with constant neighborhood range and adaptation strength" *Neural Proc. Letters*, 1995, vol.2, No.5, pp.1-5

[8] Ancona F, Rovetta S, Zunino R "Hardware architectures for Vector Quantization in very low bit-rate image coding", *Int. Workshop on HDTV HDTV'96* - Los Angeles - October 1996

[9] Oddone G, Rovetta S, Uneddu G, Zunino R "WTA circuit with proportional output", *World Congr. Neural Networks WCNN'96* - SanDiego USA - September 1996, pp.1376-1379

[10] K. Tsang and B.W.Y. Wei, "A VLSI architecture for a real-time code book generator and encoder of a vector quantizer", *IEEE Transactions on VLSI Systems*, vol. 2, no. 3, pp 360–364, Sept. 1994

[11] Blair96 G.M. Blair, "Low cost sorting circuit for VLSI," *IEEE Trans. Circ. Syst. I*, vol. 43, no. 6, Jun. 1996

[12] Fang Y, Cohen MA, Kincaid TG "Dynamics of a Winner-Take-All network", *Neural Networks*, vol.9, No.7, 1996, pp.1141-1154

[13] Carrato S, Passaggio F, Rovetta S, Zunino R "Interpolation approaches to Vector Quantization for image compression", *IEEE Workshop Neur.Net. in Signal Proc. NNSP'96* - Kyoto - September 1996, pp.391-400

[14] Xu L "Bayesian-Kullback coupled YING-YANG machines: unified learnings and new results on vector quantization", *Int.Conf.on Neur.Inf.Proc.*, Bejing, Oct.1995, pp.977-988

[15] Ridella S, Rovetta S, Zunino R "Generalization-based approach to plastic vector quantization", *World Congress on Neural Networks WCNN'95*, Washington, July 1995, vol. I, pp.505-508