Average Competitive Learning Vector Quantization

Luis Armando Salomon, Jean-Claude Fort, Li-Vang Lozada Chang

Average Competitive Learning Vector Quantization

Luis A. Salomón[a,c,*], Jean-Claude Fort[b], Li-Vang Lozada Chang[c]

[a] *Institut de Mathématiques de Toulouse, 118 route de Narbonne, F-31062. Toulouse, France*
[b] *Université Paris Descartes, 45 rue des Saints Pères, 75006. Paris, France*
[c] *Universidad de La Habana, Facultad de Matemática y Computación, San Lázaro y L, CP-10400. La Habana, Cuba*

Abstract

We propose a new algorithm for vector quantization:*Average Competitive Learning Vector Quantization*(ACLVQ). It is a rather simple modification of the classical *Competitive Learning Vector Quantization*(CLVQ). This new formulation gives us similar results for the quantization error to those obtained by the CLVQ and reduce considerably the computation time to achieve the optimal quantizer. We establish the convergence of the method via the Kushner-Clark approach, and compare the two algorithms via the central limit Theorem. A simulation study is carried out showing the good performance of our proposal.

Keywords: Optimal quantization, Stochastic algorithms, Asymptotic convergence, Kushner-Clark method.
2000 MSC: 94A29, 62L20, 60F15, 90C51, 60F05, 65D30

1. Introduction

The quantization problem comes from the theory of signal processing in electrical engineering in the late 1940's. More precisely the quantization of probability distributions is related to the best approximation of a d-dimensional probability distribution $\mathbb{P}$ by a discrete probability distribution with finite support. The mathematical aspects of quantization are treated

*Corresponding author. Phone: +5352939272
Email addresses: algoritmo.cu@gmail.com (Luis A. Salomón), fort43@gmail.com (Jean-Claude Fort), livang@gmail.com (Li-Vang Lozada Chang)

extensively in **?**]. This theory has been applied in several areas such as cluster analysis, pattern recognition, finance and numerical probability. For a detailed exposition on the applications of vector quantization we refer the reader to **?**], **?**], **?**], **?**] and references therein.

Searching an optimal quantizer is one of the main interest on this framework. Only for a few cases the close form is known. It is posible to compute the optimal quantizer at level n in one dimension as the solution of a stationarity equation either by a zero search method (Newton-Raphson gradient descent) or a fixed point procedure (like the Lloyd I procedure). In higher dimensions, deterministic gradient descent methods become intractable and one uses Lloyd I procedure and/or stochastic procedures to compute optimal quantizers (see for instance **?**]).

The basic stochastic approximation algorithms was introduced in the 1950s by the works of Robbins and Monro. The stochastic gradient method (i.e. a method of minima searching using the gradient) is based on the integral representation of the criterion to be optimized and can be seen as a particular case of the Robbins-Monro algorithm (see **?** **?**] for instance).

The *Competitive Learning Vector Quantization* (CLVQ) is an on-line algorithm that can be seen as a particular case of the "stochastic gradient method with decreasing step". A good reference here is **?**]. We also refer the reader to **?**] where the authors give a detailed exposition of the application of the CLVQ. In **?**], **?**], **?**] and references therein we can found several variations around the CLVQ method.

The classical formulation of the CLVQ method has two phases: *competitive* and *learning*. The first one is the most time consuming, because it uses at each step a closest neighbor search to find the nearest point to the random vector generated at each iteration of the method. The second phase only actualizes the value of the quantizer. The performance of a vector quantization system depends on the composition of the codebook, the criteria to find an optimal quantizer and the calculations performed in order to find the best solution.

We propose a close algorithm to the CLVQ: *Average Competitive Learning Vector Quantization*. The advantage of using our proposal lies in the fact that we reduce the computation time for the obtention of optimal or stationary quantizers. We emphasize that our proposal attains similar results regarding the quantization error to those obtained by the CLVQ. Our method introduce a slight Lloyd modification in *competitive* phase to find the "winning index" which allows to reduce considerably the time and amount of calculations.

The rest of article is organized as follows. Next section describes the CLVQ and ACLVQ algorithms. There we recall briefly the approach of Kushner-Clark in the framework of stochastic algorithms. Section 3 is devoted to the proof of the convergence of ACLVQ algorithm. Section 4 shows the performance of our proposal by numerical experiments. Finally in Section 5 the conclusions are presented.

2. The Average Competitive Learning Vector Quantization.

2.1. Quadratic vector quantization.

Let us denote by $\mathbb{P}$ the target probability distribution and $\mathcal{O} = \{x \in (\mathbb{R}^d)^n, \forall i \neq j,\ x_i \neq x_j\}$. We assume that $\mathbb{P}$ is diffuse, with a moment of order $2+\delta, \delta > 0$. This assumption ensures that $\mathbb{P}(x \in \mathcal{O}) = 1$. Throughout this work we will assume that our algorithms lives in $\mathcal{O}$.

Let X be a square integrable $\mathbb{R}^d$-valued random vector with distribution $\mathbb{P}$ defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. For $n \in \mathbb{N}$, let $\mathcal{Q}_n$ be the set of all Borel measurable maps $f : \mathbb{R}^d \to \alpha = \{x_1, \cdots, x_n\} \subset \mathbb{R}^d$ with $|f(\mathbb{R}^d)| \leq n$ (where $|A|$ denote the cardinal of the set A). The elements of $\mathcal{Q}_n$ are called n-quantizers.

Among all Borel functions $f \in \mathcal{Q}_n$ the one with the best properties is that obtained by the rule of closest neighbour:

$$f(X) = \sum_{i=1}^{n} x_i \mathbf{1}_{C_i(X)}(X),$$

where $\{C_i(X)\}_{1 \leq i \leq n}$ is a Voronoï partition for $\mathbb{R}^d$ satisfying

$$C_i(X) \subset \{x \in \mathbb{R}^d : ||x - x_i|| = \min_{1 \leq j \leq n} ||x - x_j||\}.$$

Every n-quantizer induce an error.

Definition 2.1. *The n-th quadratic quantization error for X is defined by*

$$e_n(X) \ = \ \inf \left\{ (\mathbb{E} \min_{a \in \alpha} ||X - a||^2)^{1/2} : \alpha \subset \mathbb{R}^d,\ |\alpha| \leq n \right\}.$$

We said that α is the optimal n-quantizer for X if

$$e_n(X) = \left(\mathbb{E} \min_{1 \leq j \leq n} ||X - x_j||^2 \right)^{1/2}.$$

Finding an optimal quantizer could be a hard task for some distributions.

Definition 2.2. ***Stationary quantizers.*** *A n-quantizer $\widehat{X}^\alpha$ is called stationary if any of the two following conditions are satisfied:*

1. *If $\widehat{X}^\alpha$ is a nearest neighbour projection that satisfies*

$$\widehat{X}^\alpha = \mathbb{E}(X|\widehat{X}^\alpha). \tag{1}$$

2. *If $\widehat{X}^\alpha$ is a critical point of the distortion, i.e.:*

$$\nabla e_n(\widehat{X}^\alpha) = 0.$$

Obviously an optimal n-quantizer is always n-stationary.

2.2. The classical Competitive Learning Vector Quantization

In applications, one generally prefers recursive algorithms, owing to their relative computational simplicity. Each successive estimate is obtained as a simple function of the last estimate and the current observation. *Competitive Learning Vector Quantization* is a classical example of these recursive methods. The general formulation of the CLVQ algorithm can be described briefly as follows.

CLVQ scheme.

1. Generate the initial n-quantizers $x_0^1, \cdots, x_0^n$.
2. Generate (independently) a random vector ξ with distribution $\mathbb{P}$.
3. Find the winning index (*competitive phase*)

$$i_0(k+1) = \arg\min_i \|x_k^i - \xi\|_2.$$

4. Update the quantizers (*learning phase*)

$$\begin{cases} x_{k+1}^i = x_k^i - \gamma_{k+1}(x_k^i - \xi), & i = i_0(k+1) \\ x_{k+1}^i = x_k^i, & i \neq i_0(k+1) \end{cases}.$$

5. Repeat steps 2-4 until to satisfy some convergence criteria.

As we mentioned before this method is a "stochastic gradient method with decreasing step". Thus the previous scheme could be written under that theory.

Definition 2.3. ***Stochastic gradient method.*** *Let g be a twice differentiable function from E to $\mathbb{R}$ such that dg has an integral representation on E with respect to $\mathbb{P}$, i.e.:*

$$dg(x) = \int_{\mathbb{R}^d} dG(x,\xi)\mathbb{P}(d\xi),$$

with $dG : E \times \mathbb{R}^d \to \mathbb{R}$, $dG(x,\cdot) \in \mathcal{L}^1(\mathbb{P})$ for every $x \in E$. The stochastic gradient method is defined by a triplet of sequences $((X_k)_{k\geq 0}, (\xi_k)_{k\geq 1}, (\gamma_k)_{k\geq 1})$ with values respectively in E, $\mathbb{R}^d$ and $\mathbb{R}_+$ satisfying for every $k \geq 1$ that

$$X_{k+1} = X_k - \gamma_{k+1} dG(X_k, \xi_{k+1}),$$

where $(\xi_k)_{k\geq 1}$ are i.i.d random variables with distribution $\mathbb{P}$ and $(\gamma_k)_{k\geq 1}$ is a decreasing positive deterministic sequence going to zero such that $\sum \gamma_k = +\infty$.

The theorem which ensures the convergence of such algorithms reads as follows.

Theorem 2.1. ***Convergence a.s.*** *Let $g : E \to \mathbb{R}_+$ be a continuously differentiable function whose differential dg admits an integral representation on E with respect to $\mathbb{P}$*

$$dg(x) = \int_{\mathbb{R}^d} dG(x,\xi)\mathbb{P}(d\xi).$$

Assume that dg and dG satisfy

$$\lim_{|x|\to+\infty} g(x) = +\infty \text{ and } dg \text{ is Lipschitz continuous,}$$

$$dg(x) = O(g(x)), \text{ when } |x| \to +\infty.$$

Let $((X_k)_{n\geq 0}, (\xi_k)_{k\geq 1}, (\gamma_k)_{k\geq 1})$ be a stochastic gradient method with a positive gain parameter sequence satisfying

$$\sum_{k\geq 1} \gamma_k = +\infty, \qquad \sum_{k\geq 1} \gamma_k^2 < +\infty.$$

Then $g(X_k)$ a.s. converges to some nonnegative random variable g_∞ and X_k a.s. converges toward some random connected component χ^* of $\{dg = 0\} \cap \{g = g_\infty\}$. In particular, if $\{dg = 0\}$ is a finite set, with probability 1 it exists an $x^* \in \{dg = 0\}$ such that:

$$X_k \to x^* \text{ when } k \to +\infty.$$

In general we can not ensure that the algorithm converges to a global minimum. In higher dimension, uniqueness of stationary quantizers clearly often fails (see **?**]), so that the CLVQ reaches, a priori, only locally optimal quantizers. Even if using a procedure of Lloyd the numerical precision can be improved, there is no reason to suppose that the obtained quantizers are the optimal ones.

One crucial fact for this kind of algorithms is the choice of the gain sequence. Usually some heuristic methods are made attending to the distribution $\mathbb{P}$ to quantize (see **?**]). For instance, a gain sequence $(\gamma_k)_{k\geq 1}$ usually used is

$$\gamma_k = \gamma_0 \frac{a}{a + (\gamma_0 bk)^c},$$

for some constants γ_0, a, b and c. That is the gain sequence used in the numerical experiments presented in Section 4.

When we know the $a.s.$ convergence of the algorithm toward a minimum, we may look for the speed of convergence. It is given by the following central limit theorem (see $e.g$ [**?**]):

Theorem 2.2. ***Central Limit Theorem*** *The assumptions are those of the theorem(2.1), plus:*

- *g is C^2 and $S = d^2g(x^\star)$ is definite positive, with smallest eigenvalue λ_1.*
- *Denoting $\varepsilon_{k+1} = dG(X_k, \xi_{k+1}) - dg(X_k)$, we assume that it exists a limiting variance for ε_k:*

$$\lim_{k\to\infty} \mathbb{E}(\varepsilon_k \varepsilon_k^t | \mathcal{F}_k) = V,$$

  where V is definite positive. Here $\mathcal{F}_k$ is the σ-algebra of the past, generated by $(X_0, (\xi_l)_{1\leq l\leq k})$.

- *for some $\delta > 0$, $\sup_k \|\varepsilon_k\|^{2+\delta} < \infty$.*

Then on the event X_k converges toward x^*, $\sqrt{\gamma_k}(X_k - x^*)$ weakly converges toward a $\mathcal{N}(0, \Sigma)$, a normal distribution with mean 0 and variance Σ. The matrix Σ is the solution of the following equation:

if $\gamma_k \sim \frac{\alpha}{k+\beta}$ with $2\alpha > \frac{1}{\lambda_1}$,

$$(S - \frac{1}{2\alpha} I)\Sigma + \Sigma(S - \frac{1}{2\alpha} I)^t = V, \tag{2}$$

if $\gamma_k = o(\frac{1}{k})$

$$S\Sigma + \Sigma S^t = V. \tag{3}$$

2.3. The Average Competitive Learning Vector Quantization

The CLVQ only changes one quantizer at each iteration, and thus needs a large number of simulations to achieve good results. Our proposal is a modification of the *competitive* phase of the CLVQ method. This new procedure introduce a sort of small Lloyd in this phase. The ACLVQ generates a set of N random vectors ξ instead of one as the CLVQ method do. Our proposal requires less time to achieve quantization errors of the same order despite it use the same number of simulations needed by the CLVQ. The underlying idea could be described as follows

ACLVQ scheme.

1. Generate the initial n-quantizers $x_0^1, \cdots, x_0^n$.
2. Assign $x^i = x_0^i$ for all i.
3. Generate a set of *i.i.d.* random vectors $\xi = (\xi_1, \cdots, \xi_N)$ (with the same distribution $\mathbb{P}$).
4. Calculate the distance matrix between the n-quantizer and the random vectors (*competitive phase*).
5. Identify the "Voronoï region" for each x^i in the step k using ξ, i.e.:

$$\mathcal{I}_k^i = \{j : \|\xi_j - x_k^i\| < \|\xi_j - x_k^m\|,\ \forall\, m \neq i\}, \quad i = 1, \cdots, n,\ j = 1, \cdots, N.$$

6. Calculate the mass centroid of the "Voronoï region" associated to each $\mathcal{I}_k^i$

$$\widetilde{\xi}_i = \frac{\sum_{i \in \mathcal{I}_k^i} \xi_i}{\mathcal{N}_k^i} \mathbf{1}_{\mathcal{I}_k^i \neq \emptyset},$$

   with $\mathcal{N}_k^i = |\mathcal{I}_k^i|$, where $|A|$ denotes the cardinal of the set A.
7. Update the quantizers (*learning phase*)

$$\begin{cases} x_{k+1}^i = x_k^i - \gamma_{k+1}(x_i^k - \widetilde{\xi}_i), & I_k^i \neq \emptyset \\ x_{k+1}^i = x_k^i, & I_k^i = \emptyset \end{cases}$$

   where $\gamma_k > 0$ is the gain sequence as usual.
8. Repeat steps 2-7 until reach some convergence criteria.

It is worth pointing out that this new method is not a stochastic gradient method with decreasing step as the CLVQ, even if it is very close to. The convergence theorem stated for this kind of algorithms can not be used directly. For that reason we need to write our proposal in the Kushner-Clark settings to obtain the asymptotic behavior of the ACLVQ solution.

The Kushner-Clark theorem has been used to describe the behavior of stochastic algorithms. It is also called *method of ordinary differential equation* and was introduced in **?**]. The most important results on $a.s.$ convergence of stochastic algorithms is due to the mentioned theorem. The main idea comes from the link between the behavior of the solution of the ODE ($\dot{x} = -h(x)$) and the convergence of each sample path of the stochastic algorithm.

Let us consider the general recursive algorithm:

$$\begin{cases} X^0 \in \mathbb{R}^d, \\ \\ X_{k+1} = X_k - \gamma_{k+1} f(X_k, \xi_{k+1}), \end{cases} \tag{4}$$

where $k \in \mathbb{N}$, $(\xi_k)_{k\geq 1}$, $(X_k)_{k\geq 0}$ are two sequences on $\mathbb{R}^d$, γ_k is the gain of the method (i.e. $\gamma_k \searrow 0$) and H some general function, defined as $f(X_k, \xi_{k+1}) = H(X_k, \xi_{k+1}) - \eta_{k+1}$ where η_k is a noisy term such as $\eta_k \to 0$ when $k \to +\infty$.

Let us define the mean function h as $h(X_k) = \mathbb{E}(H(X_k, \xi_{k+1}))$. In the Kushner-Clark approach the convergence of $(X_k)_{k\geq 1}$ is "conditional $a.s.$" and related to the average ODE, which describes the mean behavior of the stochastic algorithm. For instance the limiting points x^* of the algorithm in (4) lies in $\{h = 0\}$.

Let us define $\mathbf{X} = (X^i)_{1\leq i\leq n}$, where $X^i \in \mathbb{R}^d$ for all $i = 1, \cdots, n$ and γ_k is the gain sequence as before. The general model under the Kushner-Clark settings can be written as:

$$X^i_{k+1} = X^i_k - \gamma_{k+1} H_i(\mathbf{X}_k, \boldsymbol{\omega}_{k+1}), \tag{5}$$

where $H_i(\mathbf{X}_k, \boldsymbol{\omega}_{k+1})$ is a function from $\mathbb{R}^d \times \mathbb{R}^{Nd}$ to $\mathbb{R}$, $(\boldsymbol{\omega}_k)_{k\geq 1}$, is a sequence of random vectors in $\mathbb{R}^{Nd}$: $\boldsymbol{\omega}_k = (\omega^1_k, \cdots, \omega^N_k)$, with $\omega^j_k \overset{i.i.d}{\sim} \mathbb{P}$ for all k and $j = 1, \cdots, N$, $N \in \mathbb{N}$. Therefore $\boldsymbol{\omega}_k \sim \mathbb{P}_N = \bigotimes_{j=1}^N \mathbb{P}$ and obviously $\mathbf{X}_k$ is of the same kind as $\mathbf{X}$.

The theorem which assures the convergence $a.s$ of the ACLVQ to an optimal or stationary quantizer reads as follows

Theorem 2.3. ***ACLVQ convergence.*** *If the sequence* $\mathbf{X}_k = (X_k^1, \cdots, X_k^n)$ *defined by the stochastic algorithm given in equation (5) lives in a compact subset of* $\mathcal{O}$*, then it converges* $\mathbb{P}$*-a.s to a limiting point* $x^\star = (x^{\star,1}, \cdots, x^{\star,n})$ *(a stationary quantizer) of the function* $\mathbf{h} = (h_1, \cdots, h_n)$*, where* $h_i(\mathbf{X}) = \mathbb{E}\left[H_i(\mathbf{X_k}, \boldsymbol{\omega}_{k+1})|\mathbf{X_k} = \mathbf{X}\right]$*.*

The proof of previous theorem is left to the next section.

3. ACLVQ convergence

We use the Kushner-Clark theorem to prove the asymptotic convergence of the solution of (5) toward one "optimal" or stationary quantizer of the distribution.

3.1. Preliminaries

Let us rewrite the general equation (4)

$$\begin{cases} X^0 \in \mathbb{R}^d, \\ \\ X_{k+1} = X_k - \gamma_{k+1} H(X_k, \xi_{k+1}) + \gamma_{k+1}\eta_{k+1}. \end{cases}$$

Defining $\Delta M_{k+1} = h(X_k) - H(X_k, \xi_{k+1})$ the previous formulation reads as follows

$$\begin{cases} X^0 \in \mathbb{R}^d, \\ \\ X_{k+1} = X_k - \gamma_{k+1} h(X_k) + \gamma_{k+1}(\Delta M_{k+1} + \eta_{k+1}). \end{cases} \tag{6}$$

We state the following assumption

$$(\mathbf{R}) \equiv \begin{cases} (i.) \sum_{s=1}^{+\infty} \gamma_s \Delta M_s < +\infty \\ \\ (ii.) \lim_{s\to\infty} \eta_s = 0 \end{cases}.$$

Theorem 3.1. ***Kushner-Clark.*** *If* x^* *is an equilibrium point of the ODE* $\dot{x} = -h(x)$*,* h *a continuous function, the sequence* $(X_k)_{k\geq 1}$ *in (4) is bounded and the assumption* $\mathbf{R}$ *holds, then*

$$X_k \to x^*, \qquad as \quad k \to +\infty,$$

on the event $A_K^{x^*} \stackrel{\triangle}{=} \{X_k \in K \subset G_{x^*}$ *infinitely often*$\}$*, where* K *is a compact set.*

The Kushner-Clark theorem needs the boundedness of the solutions $(X_k)_{k\geq 1}$ and the continuity of h. In some cases the choice of h leads to the boundedness of the solution. The continuity requirement for h could be relaxed using Lyapunov functional (see for instance **?**]).

3.2. Proof of the Theorem 2.3

Let us begin with the following lemma. We will denote $p_i(x) = \mathbb{P}(C_i(x)$ and $q_i(x) = (1 - (1 - \mathbb{P}(C_i(x)))^N)$.

Lemma 3.1. *The function $h = (h_1, \cdots, h_n)$ ($\mathbf{h}(\mathbf{x})$ is defined for $x = (x^1, \cdots, x^n)$, $x^i \in \mathbb{R}^d$ for all $i = 1, \cdots, n$) is $\mathbb{P}$-a.s. continuous and its zeros are exactly stationary quantizers for the distribution $\mathbb{P}$, since h writes:*

$$h_i(\mathbf{x}) = \frac{q_i(x)}{p_i(x)} dg_i(x). \tag{7}$$

here dg_i is the derivatives of g with respect to x_i.

Proof. Firstly we prove the second statement of the lemma. We have that

$$H_i(\mathbf{X}_k, \boldsymbol{\omega}_{k+1}) = \begin{cases} X_k^i - \frac{\sum_{j=1}^N \mathbf{1}_{C_i(\mathbf{X}_k)}(\omega_{k+1}^j)\omega_{k+1}^j}{N_{k+1}^i} & \text{if } N_{k+1}^i > 0 \\ \\ X_k^i & \text{if } N_{k+1}^i = 0, \end{cases}$$

where $N_{k+1}^i = \sum_{j=1}^N \mathbf{1}_{C_i(\mathbf{X}_k)}(\omega_{k+1}^j)$, N_{k+1}^i is the number of times that a random vector lies in the Voronoï region of X_k^i. Furthermore

$$\begin{aligned} \mathbb{P}(N_{k+1}^i = 0) &= \mathbb{P}(\omega_{k+1}^j \notin C_i(\mathbf{X}_k)),\ \forall j = 1, 2, \cdots, N \\ &= (1 - \mathbb{P}(C_i(\mathbf{X}_k)))^N, \end{aligned}$$

thus $\mathbb{P}(N_{k+1}^i > 0) = 1 - (1 - \mathbb{P}(C_i(\mathbf{X}_k)))^N$.

We use $\boldsymbol{\omega}$ instead $\boldsymbol{\omega}_k$ in order to simplify the notations.

$$h_i(\mathbf{X}_k) = X_k^i \mathbb{P}(N_{k+1}^i > 0) - \int_{\{N_{k+1}^i > 0\}} \frac{1}{N_{k+1}^i} \sum_{j=1}^N \mathbf{1}_{C_i(\mathbf{X}_k)}(\omega_{k+1}^j)\omega_{k+1}^j \mathbb{P}_N(d\boldsymbol{\omega}).$$

Denoting

$$\Omega^r = \{\omega_{k+1}^1 \in C_i(\mathbf{X}_k), \cdots, \omega_{k+1}^r \in C_i(\mathbf{X}_k), \omega_{k+1}^{r+1} \notin C_i(\mathbf{X}_k) \cdots, \omega_{k+1}^N \notin C_i(\mathbf{X}_k)\},$$

and knowing that $\omega_k^j \overset{i.i.d}{\sim} \mathbb{P}$ it follows

$$\begin{aligned} h_i(\mathbf{X}_k) &= X_k^i \mathbb{P}(N_{k+1}^i > 0) - \sum_{r=1}^{N} \frac{1}{r} \binom{N}{r} \sum_{j=1}^{r} \int_{\Omega^r} \mathbf{1}_{C_i(\mathbf{X}_k)}(\omega_{k+1}^j) \omega_{k+1}^j \mathbb{P}_N(d\boldsymbol{\omega}) \\ &= \left[X_k^i - \frac{\mathbb{E}\mathbf{1}_{C_i(\mathbf{X}_k)}(\omega^1)\omega^1}{\mathbb{P}(C_i(\mathbf{X}_k))} \right] \mathbb{P}(N_{k+1}^i > 0). \end{aligned}$$

Set

$$U_i(\mathbf{X}_k) = \frac{\mathbb{E}\mathbf{1}_{C_i(\mathbf{X}_k)}(\omega)\omega}{\mathbb{P}(C_i(\mathbf{X}_k))} = \mathbb{E}\left[\mathbf{X} | \mathbf{X} \in C_i(\mathbf{X}_k)\right], \qquad \forall i = 1, 2, \cdots, n,$$

where $\omega \sim \mathbb{P}$. We have

$$h_i(\mathbf{X}_k) = \mathbb{E}\left[H_i(\mathbf{X}, \boldsymbol{\omega}_{k+1}) | \mathbf{X} = \mathbf{X}_k\right] = \mathbb{P}(N_{k+1}^i > 0)(X_k^i - U_i(\mathbf{X}_k)).$$

This gives formula (7), and it is quite obvious that $h_i(\mathbf{X}_k) = 0$ if and only if $X_k^i = U_i(\mathbf{X}_k)$. Which means that the zeros of $\mathbf{h} = (h_1, \cdots, h_n)$ are exactly stationary quantizers.

The first assertion of the lemma is clear since $\mathbb{P}$ is supported by $\mathcal{O}$ ($\mathcal{O} \subset \mathbb{R}^d$ is an open set in $\mathbb{R}^d$): the functions $\int_{C_i(\mathbf{X}_k)} \mathbb{P}(d\omega), \int_{C_i(\mathbf{X}_k)} \omega \mathbb{P}(d\omega)$ are continuous on $\mathcal{O}$ and the first one is positive on $\mathcal{O}$.

□

Proof of Theorem 2.3. By Lemma 3.1 the proof is a direct consequence of the Kushner-Clark theorem.

□

4. About the asymptotics of the CLVQ and ACLVQ

The Central Limit Theorem (theorem 2.2) straightforwardly applied to $CLVQ$. For $ACLVQ$ we need again a theorem a bit more general since it is only close to a stochastic gradient algorithm. Nevertheless, the assumptions and result are identical to that of theorem 2.2, replacing dg by $\mathbf{h}$ and $d^2g(x*)$ by the derivatives of $\mathbf{h}$ (see [**?**]). The Hessian d^2g has been computed by Pollard and Fort-Pagès (see ([**?**]),([**?**]).

By noticing that $dg(x^*) = 0$ we easily obtain $d\mathbf{h}(x^*)$:

$$d\mathbf{h}(x^*) = diag(\frac{q_i(x^*)}{p_i(x^*)})d^2g(x^*).$$

Now in order to compare the asymptotics of the two algorithms, we need a rescaling. Indeeed the ACLVQ use N times more samples at each iteration. So we will compare CLVQ at step k with ACLVQ at step $[k/N]$, where $[u]$ is the entire part of $u \in \mathbb{R}$. Of course the gains γ are to be rescaled too: if the gain is γ_k at iteration k for CLVQ, it will be γ_{kN} at iteration k for ACLVQ. Denoting for convenience the ACLVQ X^N we thus obtain the following asymptotics.

For CLVQ : $\sqrt{\gamma_k}(X_k - x^*)$ weakly converges toward a $\mathcal{N}(0, \Sigma)$.

For ACLVQ $\sqrt{\gamma_{[\frac{k}{N}]N}}(X^N_{[\frac{k}{N}]} - x^*)$ weakly converges toward a $\mathcal{N}(0, \Sigma_N)$.

As $\gamma_{[\frac{k}{N}]N} \sim \gamma_k$, it remains to compare the two covariance matrices Σ and Σ_N. Let us denote V the asymptotic variance matrix of the noise for CLVQ (notice that V is a diagonal matrix), and $N^i, 1 \leq i \leq N$ a multinomial random variable with parameter $(p_i(x^*)_{1\leq i\leq N})$. As $x*$ is a stationary quantizer, one can verify that the asymptotic variance of the noise for ACLVQ is:

$$V_N = diag(\mathbb{E}\frac{\mathbf{1}_{N^i} > 0}{p_i(x^*)N^i})V.$$

The equations (2) and (3) remain valid, but $d\mathbf{h}(x^*)$ is no more symetric. Then, for instance in the case of equation (3), the matrix Σ_N satisfies:

$$diag(\frac{q_i(x^*)}{p_i(x^*)})d^2g(x^*)\Sigma_N + \Sigma_N[diag(\frac{q_i(x^*)}{p_i(x^*)})d^2g(x^*)]^t = diag(\frac{1}{p_i(x^*)}\mathbb{E}\frac{\mathbf{1}_{N^i} > 0}{N^i})V.$$

Now we notice that $E\frac{\mathbf{1}_{N^i>0}}{N^i} \leq \mathbb{P}(N^i > 0) = q_i(x^*)$, and setting $D = diag(\frac{q_i(x^*)}{p_i(x^*)})$ we obtain:

$$Dd^2g(x^*)\Sigma_N + \Sigma_N d^2g(x^*)^t D \leq DV,$$

as semi definite matrices. Introducing $D^{\frac{1}{2}}$, using equation (3) for Σ we deduce that the matrix:

$$M = D^{\frac{1}{2}}(d^2g(x^*)\Sigma_N - d^2g(x^*)\Sigma)D^{-\frac{1}{2}} + (D^{\frac{1}{2}}d^2g(x^*)\Sigma_N - d^2g(x^*)\Sigma D^{-\frac{1}{2}})^t,$$

is semidefinite negative. From this we conclude that $d^2g(x^*)\Sigma_N - d^2g(x^*)\Sigma$ has negative eigenvalues, which yields $\Sigma_N \leq \Sigma$ as semi definite matrices.

We now conclude that ACLVQ with the same number of samples is asymptotically more accurate than CLVQ.

This result as to be softened by the fact that when N is large, the asymptotic of ACLVQ is reached much longer after CLVQ. So choosing N too large would cause ACLVQ to fail with respect to CLVQ.

5. Numerical results

In this section, numerical comparisons between the classical CLVQ and our proposal are carried out in order to investigate the accuracy and computation time of each method. Our main interest is to illustrate the advantage of the ACLVQ regarding the computation time.

We carry out numerical experiments with distributions in dimension one and two. In dimension one it is not practical using the ACLVQ because determinist algorithms like the Newton one perform very well (see **?**]). However, these runs let us to evaluate the performance of the ACLVQ and the classical CLVQ for the standard Gaussian distribution. In dimension 2 we make the major portion of the numerical simulations. In this setting we consider 4 different distributions. Only for the uniform distribution in the unit cube for any dimension and particular sizes of the quantizer it is known the exact value of the distortion error. Therefore we consider these distributions in both dimensions concerned. More precisely, for $n = k^2$, in **?**] is proved that

$$e_n^2(U([0,1]^d)) = n^{-2/d}\frac{d}{12}.$$

In Table 1 we show the choice of the parameters of the probability distributions concerned in the study in dimension two.

Distribution	Parameters
$N(0, I_2)$	
$N(0, \Sigma_2)$	$\Sigma_2 = (\sigma_{ij})_{1\leq i,j\leq 2}$: $\sigma_{11} = 7.1$, $\sigma_{22} = 1.7$ and $\sigma_{ij} = 0$ if $i \neq j$
$U([0,1]^2)$	
$Exp(\Psi_2)$	$\Psi_2 = (\mu_{ij})_{1\leq i,j\leq 2}$: $\mu_{ii} = 1$ and $\mu_{ij} = 0$ if $i \neq j$

Table 1: Probability distributions in dimension 2 used in the study.

In the *competitive* phase we use five different values of $N > 1$, the number of random vector generated in each iteration of the method. The value $N = 1$ corresponds to the CLVQ. For each, distribution, n and N we run 100 repetitions.

To make a fair comparison between both methods we assure generate similar number of random vectors for each case. More precisely, we compare the runs of the CLVQ having total number of iteration K with those of the ACLVQ of N simulations of the random vector by generation and total number of generations K/N. Hence, both methods use the same "total information": K copies of the random vector. We set the same initial vector x_0 for each n and all N. For the gain sequence

$$\gamma_k = \gamma_0 \frac{a}{a + (\gamma_0 N b k)^c}$$

we set $a = 1$, $b = 0.1$ and $\gamma_0 = 0.2$. In Table 2 we present the parameter values chosen in the study. All computations were made with `MATLAB 7.11.0.584 R2010b` in a computer with the following characteristics: `CPU Pentium(R) Dual Core T4500 2.30 GHz` and `3Gb RAM`.

$n = 10$			$n = 50$		
N	K^{10}	c	N	K^{50}	c
1	10^6	0.51	1	4×10^6	0.51
10	10^5	0.45	50	8×10^4	0.35
20	5×10^4	0.45	100	4×10^4	0.3
50	2×10^4	0.45	250	1.6×10^4	0.28
100	10^4	0.35	500	8×10^3	0.15
200	5×10^3	0.3	1000	4×10^3	0.09

Table 2: Parameter values used in the simulation study in dimension 2.

5.1. Quantization error

First of all, we point out that the computation of the quantization error in all the cases was made by Monte Carlo simulation using a dataset of 10^6 random vector copies.

In **?**] the authors reported quantization errors for the n-quantizer of the standard Gaussian distribution for $n = 2, 3, \cdots, 8$. Our algorithm reaches results quite similar to those. The simulation study shows that our method

exhibits a small standard deviation (of order of 10^{-4}) over the 100 simulation for each distribution and each value of N. Similar behavior was observed for the one dimensional uniforme distribution on $[0, 1]$. These results support our assumption that the ACLVQ has a good accuracy.

In Figure 1 and Figure 2 we display the boxplots of the quantization errors after 100 repetitions of the experiment for each of values of N chosen and $n = 10$ and $n = 50$ respectively. In Table 3 we shows the rest of the results obtained. As can be observed the results obtained by the CLVQ ($N = 1$) and by the ACLVQ are close. The standard deviation is small. Moreover, the quantizers obtained for each case had the same spatial configuration (graphs not shown).

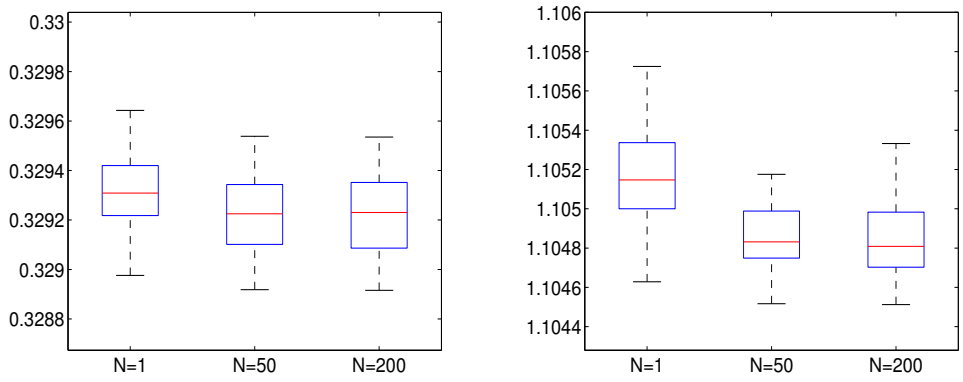

Figure 1: Boxplots of quantization error for $n = 10$ obtained after 100 runs: $N(0, I_2)$ (left) and $N(0, \Sigma_2)$ (right).

5.2. Computation time

Now, we turn our attention to the computation time. The major advantage of our proposal is the reduction of the computation time to obtain optimal quantizers.

The results regarding the computation time can be seen in Table 4. As we can see the ACLVQ reduces considerably the computation time when

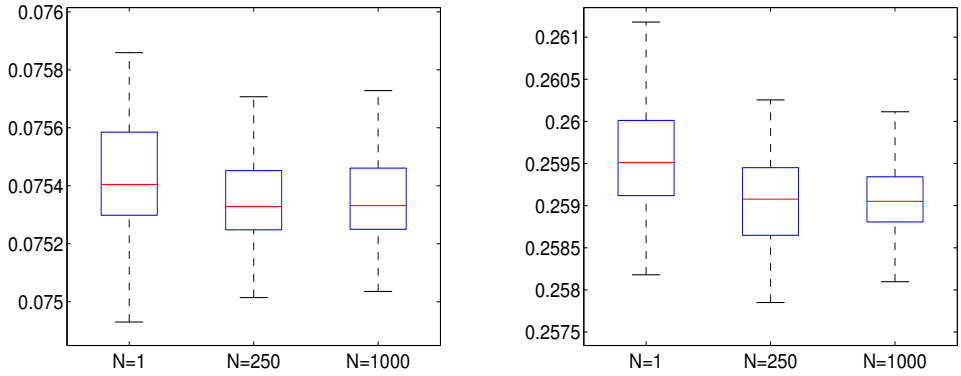

Figure 2: Boxplots of quantization error for $n = 50$ obtained after 100 runs: $N(0, I_2)$ (left) and $N(0, \Sigma_2)$ (right).

$\mathbb{P}$	n	N	$\overline{e}^2_{n,N}$	$s_{n,N}$
$U([0,1]^2)$	10	1	0.0170	0.3523×10^{-4}
		50	0.0170	0.3505×10^{-4}
		200	0.0170	0.3682×10^{-4}
	50	1	0.0033	0.1471×10^{-4}
		250	0.0033	0.0953×10^{-4}
		1000	0.0033	0.0938×10^{-4}
$Exp(\Psi_2)$	10	1	0.2492	0.1134×10^{-2}
		50	0.2494	0.1706×10^{-2}
		200	0.2494	0.1577×10^{-2}
	50	1	0.0531	0.3213×10^{-3}
		250	0.0517	0.2856×10^{-3}
		1000	0.0517	0.2117×10^{-3}

Table 3: Mean and standard deviation of the quantization after 100 runs.

N takes large values. For instance, the ACLVQ for $n = 10$ and $N = 200$ achieves the optimal quantizer 30 times faster than the CLVQ.

One could think that increasing N is all what one need to do in order to improve the performance of algorithm. However, in experiments we do not show here it was observed a decline of the accuracy of the algorithm. This is mainly due to the fact that the ACLVQ can be trapped more easily in local minima. Similar behavior is observed in the Lloyd's method I when the

$n = 10$					$n = 50$				
N	D_1	D_2	D_3	D_4	N	D_1	D_2	D_3	D_4
1	66,80	66,62	74,65	138,63	1	286,57	286,70	321,36	579,50
10	32,79	32,74	32,91	39,62	50	115,41	114,72	112,98	120,56
20	16,61	16,58	16,65	20,15	100	61,29	60,92	59,82	64,65
50	6,87	6,85	6,84	8,36	250	30,30	30,17	29,60	31,85
100	3,73	3,72	3,71	4,54	500	20,92	20,87	20,45	22,05
200	2,19	2,19	2,17	2,64	1000	17,00	16,92	16,67	17,75

Table 4: Mean of computation times (in seconds).

initial set is far from the global minimum [**?**]. Actually, the choice of N is a tradeoff between accuracy and time consuming.

The gain in time is due to two factors: the first one and the most important is related with the number of computations that take place on the *learning* phase: the ACLVQ is able to update more quantizers at each iteration than the CLVQ and, at the same time, in the overall process the ACLVQ update less times. The number of iterations needed by the ACLVQ is smaller. This fact relies on the Lloyd modification of the ACLVQ. The computation time needed for this modification is negligible for the whole process. It is important to emphasize that the number of random vector simulations generation for both algorithms is similar. The second aspect concerns the procedure used for the computation of the distance matrix in the *competitive* phase (see Appendix A) which exploits very efficiently the concept of vectorizing algorithms of `MATLAB`.

6. Some considerations in conclusion

The Average Competitive Learning Vector Quantization attains quite similar results to those obtained by the CLVQ in the searching of the optimal quantizers in $\mathbb{R}^d$ regarding the distortion error. The generation of set of random vectors in the competitive phase instead of one allows to the algorithm updating more than one quantizer at each iteration. This improvement leads to an important dismiss of consuming of computation time.

Our proposal gains in interest into problems in where the time consumption is a crucial constraint. Another important advantage of the ACLVQ is that the competitive phase is parallelizable what is not possible in the CLVQ because the strong recursive nature of this algorithm.

There are natural few ways to improve our algorithm. In a forthcoming paper we study some of them. They concern, principally, an adaptive scheme to select the number of simulations N at each competitive phase of the ACLVQ. We focus in the choice of N as a tradeoff between accuracy and computation time.

Appendix A. Euclidean distance matrix: a MATLAB procedure

We present a method to compute the Euclidean distance matrix of two sets of vectors in dimension greater than 2. This method is based in product of matrices and componentwise matrix multiplication which permits speed up computing in MATLAB.

Let us define $Q = (Q_{i,\,j})_{1\leq i\leq d,\,1\leq j\leq n}$ the matrix of quantizers and $\xi = (\xi_{i,\,k})_{1\leq i\leq d,\,1\leq k\leq N}$ the matrix of random vector with $N \in \mathbb{N}$.

To calculate the norm of $n \cdot N$ vectors: $D_{j,k} := \|Q_{\cdot,\,j} - \xi_{\cdot,\,k}\|$ for all j and all k, we propose to decompose

$$\begin{aligned} D_{j,k}^2 =& \|Q_{\cdot,\,j} - \xi_{\cdot,\,k}\|^2 = \|Q_{\cdot,\,j}\|^2 + \|\xi_{\cdot,\,k}\|^2 - 2\langle Q_{\cdot,\,j}, \xi_{\cdot,\,k}\rangle \\ =& \langle Q_{\cdot,\,j}, Q_{\cdot,\,j}\rangle + \langle \xi_{\cdot,\,k}, \xi_{\cdot,\,k}\rangle - 2\langle Q_{\cdot,\,j}, \xi_{\cdot,\,k}\rangle. \end{aligned}$$

The vectors

$$Q^\star = \left(\sum_{i=1}^{d} Q_{i,1}^2; \cdots; \sum_{i=1}^{d} Q_{i,n}^2,\right)_{1\times n}, \quad \xi^\star = \left(\sum_{i=1}^{d} \xi_{i,1}^2; \cdots; \sum_{i=1}^{d} \xi_{i,N}^2\right)_{1\times N}$$

are easily computed using elementwise self-multiplication of matrix Q and ξ respectively, and summing by rows. Set $D = \left(D_{j,k}^2\right)_{j,k}$ it follows

$$D = {}^tQ^\star\,(1,\cdots,1)_{1\times N} + {}^t(1,\cdots,1)_{n\times 1}\,\xi^\star. - 2\,{}^tQ\xi.$$

The code is available at http://profesores.matcom.uh.cu/~salomon.