3 The Effect of Finite Sample Size in On-line K-Means

Abstract- The *asymptotic* convergence of on-line algorithms when the number of training samples tend to infinite is well understood from a theoretical point of view (Benveniste et al., 1990)(Bottou, 1998)(Ljung & Söderström, 1983). However, much less is known about the real convergence of these algorithms when the data samples size is finite. In this chapter we address the study of the real convergence of the popular K-means algorithm (McQueen, 1967) when it deals with finite data resources.

Index Terms- K-means algorithm, Asymptotic convergence, Online gradient descent, Finite-sample properties, Vector Quantization, Data-dependent partitioning classifiers.

1. Introduction

On-line algorithms are one of the simplest optimization processes in the learning phase of artificial adaptive systems. This feature makes them attractive to handle large (eg. real-world) training sets using a moderated computational effort. In these cases, since the number of examples (N) is usually *fixed*, the learning algorithm stores the whole training data in memory and passes (e.g. cyclically) though them over and over until a stopping criteria is met.

In some cases this learning scenario is unavoidable since the derivation of the algorithm is intrinsically on-line (e.g. supervised LVQ algorithms; (Kohonen, 1996)). In other cases, since the algorithm is related to a particular cost function, its use is supported by the empirical evidence that the convergence speed is accelerated over batch versions if the data is redundant, a typical situation in real-life data (p.264 (Bishop, 1995), (Bengio, 1991)). Another typical argument in favor of on-line versions is that they can be considered as a 'noisy' versions of batch algorithms so they could escape from a local minimum easier.

Although the use of on-line learning algorithms when there is a cyclic or random access to a fixed set of examples is a common practice, there is no guarantee of convergence. From a theoretical point of view, convergence is only guaranteed when N tends to infinite (This holds either in the case of using a constant or decreasing step size (Benveniste et al., 1990).) Besides, the theory that studies theses systems usually does not provide any hints about the practical convergence rate. Hence, to get more insight of the finite-sample properties of these algorithms, the theoretic analysis must be always complemented by simulation studies.

This chapter addresses the study of the finite-sample convergence of the popular clustering K-means algorithm (McQueen, 1967). This algorithm is widely employed in vector quantization (Gersho & Gray, 1992), as initialization for other more powerful learning systems (like Radial Basis functions (Moody & Darken, 1989) or Kohonen’s LVQ algorithms (Kohonen, 1996)) and in data-dependent partitioning classifiers (Devroye, 1996). We study this on-line learning when it uses a cyclic or any other presentation of the training data and a *constant or variable step size*. Emphasis is done in comparing the cyclic on-line version over batch and infinite-sample on-line versions. For the study of the learning algorithm we will make a non-statistical analysis mainly based on the use of the discrete-time dynamical systems theory (Devaney, 1989) (Oppenheim & Schafer, 1989) (Wiggins, 1991).

The organization of this chapter is as follows. Section 2 reviews on-line and batch versions of the K-means algorithm. In section 3, we introduce our study of the finite-sample convergence introducing an asymptotic first-order model of the on-line K-means. To give a complete view, we also include convergence studies of the other versions. Section 4 briefly accounts the generalization performance of these algorithms. Next, experimental results are presented in order to validate the proposed model of the finite-sample on-line K-means. Finally, some discussion and conclusions are given.

2. K-means

2.1.1. Optimal On-line K-means

We want to design a codebook $\mathbf{C}_{ul}$ (or set of prototypes of size K) for a vector quantizer VQ. A VQ of dimension p and size K is defined as a mapping from a p-dimensional Euclidean space, $\Re^p$, into a set or codebook $\mathbf{C}=\{\mathbf{w}_i, i=1,...,K\}$. Associated with every codevector $\mathbf{w}_i$ exist a region of influence R_i where VQ maps any input vector that falls in it to $\mathbf{w}_i$ Since we use a nearest neighbor quantizer, R_i is defined by

$$R_i = \left\{ \mathbf{x} \,\middle|\, \left\| \mathbf{x} - \mathbf{w}_i \right\| = \min_{j=1,\dots,K} \left\| \mathbf{x} - \mathbf{w}_j \right\| \right\} \tag{1}$$

Thus, VQ($\mathbf{x}$) can be expressed as

$$\mathrm{VQ}(\mathbf{x}) = \sum_{j=1}^{K} 1(\mathbf{x} \in R_j)\mathbf{w}_j \qquad \text{where } 1(\mathbf{x} \in R_j) = \begin{cases} 1 & \text{si } \mathbf{x} \in R_j \\ 0 & \text{si } \mathbf{x} \notin R_j \end{cases} \tag{2}$$

Once VQ is defined, the principal goal is to find a codebook $\mathbf{C}_{ul}$ that will minimize a measure of performance considering the total sequence of input patterns to be quantized. This overall performance can be expressed in terms of a statistical criterion. In optimal K-means, the statistical criteria is the expected error of the vector quantizer given by the functional

$$I[\mathrm{VQ}] = \frac{1}{2} \int_{\Re^p} \sum_{i=1}^{K} 1(\mathbf{x} \in R_i) \left\| \mathbf{x} - \mathbf{w}_i \right\|^2 f_X(\mathbf{x}) d\mathbf{x} \tag{3}$$

where f_X is the probability density function of the random vector (RV) X.

We must apply the gradient descent optimization method over I[VQ] to derive the learning algorithm:

$$\mathbf{w}_j[n+1] = \mathbf{w}_j[n] - \alpha[n]\frac{\partial I[VQ[n]]}{\partial \mathbf{w}_j[n]} \quad j = 1 \ldots K \tag{4}$$

If we can exchange the integrator and differentiator operators, e.g. f_X is smooth enough (see the technical details in (Bottou, 1998)), the learning equation yields:

$$\mathbf{w}_j[n+1] = \mathbf{w}_j[n] + \alpha[n]P(X \in R_j[n])\left(E_{X|R_j[n]} - \mathbf{w}_j[n]\right) \quad j = 1 \ldots K \tag{5}$$

So finally the optimal K-means algorithm can be stated as follows:

Optimal K-means Algorithm (*K*, f_X, *α[n]* **)**

K	Number of prototypes or codevectors
f_X	Probability density function of RV X
$\alpha[n]$	Step size function

```
1. n=0
2. Initialize C[0]={$\mathbf{w}_j[0]$, j=1,...,K}
3. Compute $P(X \in R_j[n])$ and $E_{X|Rj[n]}(\mathbf{x}|R_j[n])$ for j=1,...,K
4. Apply the following update equation:
       $\mathbf{w}_j[n+1] = \mathbf{w}_j[n] + \alpha[n]P(X \in R_j[n])\left(E_{X|R_j[n]} - \mathbf{w}_j[n]\right) \quad j = 1,\ldots,K$
5. n=n+1
6. If *stop criteria is not met* go to 3
7. End
```

2.2. Batch K-means

If f_X is unknown, we cannot apply the optimal K-means algorithm. In these cases, we build an empirical estimator of I[VQ], $I_{emp}[VQ]$, with a set of random vectors extracted from f_X, in the following way:

$$I_{emp}[VQ] = \frac{1}{2N}\sum_{l=0}^{N-1}\sum_{i=1}^{K} 1(\mathbf{x}_l \in R_i)\|\mathbf{x}_l - \mathbf{w}_i\|^2 \tag{6}$$

Batch K-means make use of the Newton optimization method over $I_{emp}[VQ]$ (Bottou & Bengio, 1995), that is defined as:

$$\mathbf{W}[n+1]=\begin{bmatrix}\mathbf{w}_1[n+1]\\ \vdots \\ \mathbf{w}_K[n+1]\end{bmatrix}=\mathbf{W}[n]-\mathbf{H}[n]^{-1}\frac{\partial I_{emp}[VQ[n]]}{\partial \mathbf{W}[n]} \tag{7}$$

$$\mathbf{H}=\begin{bmatrix}\frac{\partial^2 I_{emp}[VQ]}{\partial^2 \mathbf{w}_1} & \cdots & \frac{\partial^2 I_{emp}[VQ]}{\partial \mathbf{w}_1 \partial \mathbf{w}_K}\\ \vdots & \vdots & \vdots \\ \frac{\partial^2 I_{emp}[VQ]}{\partial \mathbf{w}_K \partial \mathbf{w}_1} & \cdots & \frac{\partial^2 I_{emp}[VQ]}{\partial^2 \mathbf{w}_K}\end{bmatrix}$$

The hessian matrix H and the partial derivative of I_{emp} respect to each codevector yield:

$$\frac{\partial I_{emp}[VQ]}{\partial \mathbf{w}_j}=-\frac{1}{N}\sum_{l=0}^{N-1}1(\mathbf{x}_l\in R_j)(\mathbf{x}_l-\mathbf{w}_j)$$

$$\frac{\partial^2 I_{emp}[VQ]}{\partial \mathbf{w}_i \partial \mathbf{w}_j}=\begin{cases}0 & \text{if } i\neq j\\ N_j/N & \text{if } i=j\end{cases} \tag{8}$$

where N_j is the number of training samples that fall in the Voronoi region R_j.
Hence, the learning equation is

$$\mathbf{w}_j[n+1]=\frac{1}{N_j[n]}\sum_{l=0}^{N-1}1(\mathbf{x}_l\in R_j[n])\,\mathbf{x}_l \qquad j=1,\ldots,K \tag{9}$$

Then the batch K-means algorithms is defined by

Batch K-means(*K, {x_l, l=0...N-1}* **)**

K	Number of protoypes or codevectors
$\{\mathbf{x}_l, l=0...N-1\}$	Training data

1. n=0
2. Initialize $C[0]=\{\mathbf{w}_j[0], j=1,...,K\}$

3. Compute $T_{R_j[n]} = \{\mathbf{x}_i \mid R_j[n]\}$ where $R_j[n] = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{w}_j[n]\| = \min_{i=1..K} \|\mathbf{x} - \mathbf{w}_i[n]\|\}$

and $\mathbf{x}_i \backslash R_j[n]$ are the training samples that fall in region $R_j[n]$

4. Apply the following update equation:

$$\mathbf{w}_j[n+1] = \frac{1}{N_j[n]} \sum_{l=0}^{N-1} 1(\mathbf{x}_l \in R_j[n])\, \mathbf{x}_l \qquad j = 1,\dots,K$$

5. n=n+1

6. If *stop criteria is not met* goto 3

7. End

2.3. Cyclic and Random online K-means

As the previous algorithm, cyclic on-line K-means make use of fixed training data $\{\mathbf{x}_l, l=0,\dots,N-1\}$. But instead of using the Newton optimization method, it applies the pattern-based version of gradient descent. The empirical error function gradient is re-evaluated at each algorithm's step using *only one pattern at a time*. Then, we update the codevectors with

$$\mathbf{w}_j[n+1] = \mathbf{w}_j[n] + \alpha[n] \frac{\partial I_{emp}[n+1]}{\mathbf{w}_j[n]} \qquad j = 1,\dots,K \tag{10}$$

$$I_{emp}[n+1] = \frac{1}{2} \sum_{i=1}^{K} \|\mathbf{x}[n+1] - \mathbf{w}_i[n]\|^2$$

Note that in the cyclic version we go through the strip of training data in the following fashion: $\mathbf{x}_0$, $\mathbf{x}_1$, ...$\mathbf{x}_{N-1}$, $\mathbf{x}_0$, $\mathbf{x}_1$, ... The complete cyclic and random on-line version of K-means is shown below.

On-line K-means(*K, {x_l, l=0...N-1}, α[n]* **)**

K	The number of protoypes or codevectors
{x_l, l=0...N-1}	The training data
α[n]	The step size function

```
1. n=0
2. Initialize C[0]={$\mathbf{w}_j[0]$, j=1,...,K}
3. If cyclic version then
        $\mathbf{x}[n+1]=\mathbf{x}_{n \bmod N}$
   otherwise
        $\mathbf{x}[n+1]=\mathbf{x}_{random(1,N)}$
```

4. Compute the current nearest codevector to **x**[n+1] named $\mathbf{w}_W$[n] using the Euclidean distance:

$$\mathbf{w}_W[n] = \underset{\mathbf{w}_j,\ j=1,..,K}{\arg} \min \left\| \mathbf{x}[n+1] - \mathbf{w}_j[n] \right\|$$

5. Update only the winning prototype in the following way:

$$\mathbf{w}_W[n+1] = \mathbf{w}_W[n] + \alpha[n](\mathbf{x}[n+1] - \mathbf{w}_W[n])$$

6. n=n+1

7. If *stop criteria is not met* go to 3

8. End

3. Study of convergence

3.1. Optimal K-means

If we equal the partial derivative $\partial I[VQ]/\partial \mathbf{w}_i$ to zero, we obtain the optimal values of the codevectors that minimize the functional I[VQ]:

$$\mathbf{w}_j^*[n] = E_{X|R_j[n]} \qquad j = 1,...,K \tag{11}$$

These optimal values are the conditional expectations of X to belong to each of the Voronoi regions R_j j=1,...,K.

If the step size is small enough, we can use K ordinary differential equations (ODEs) to approximately describe the behavior of the K difference equations of this learning system. Thus, the equations

$$\mathbf{w}_j[n+1] = \mathbf{w}_j[n] - \alpha[n]\frac{\partial I[VQ[n]]}{\partial \mathbf{w}_j[n]} \qquad j = 1,...,K \tag{12}$$

can be approximated by

$$\frac{d\mathbf{w}_j(t)}{dt} = \frac{\partial I[VQ[n]]}{\partial \mathbf{w}_j[n]} \qquad j = 1,...,K \tag{13}$$

where the relation between the discrete and continuous time is given by α n=t.

Let be $E_X\left[\|\vec{x} - VQ(\mathbf{x}, t)\|^2\right]$ the Liapunov function associated to the learning system. This function is the instantaneous mean quantization error using the Euclidean distance as a measure of distortion. Since

$$\frac{dE_X\left[\|\mathbf{x} - VQ(\mathbf{x}, t)\|^2\right]}{dt} < 0 \tag{14}$$

near a ball defined around each fixed point for all possible sets of K prototypes, these points are stable. Finally due to the fact that

$$\frac{dE_X\left[\|\mathbf{x} - VQ(\mathbf{x}, t)\|^2\right]}{dt} = E_X\left[\|\mathbf{x} - VQ(\mathbf{x}, t)\|^2\right] \tag{15}$$

we can see that the instantaneous error function is exponentially decreasing

$$E_X\left[\|\mathbf{x} - VQ(\mathbf{x}, t)\|^2\right] = ce^{-\frac{t}{2}} \tag{16}$$

See for a complete derivation of these results (Kosko, 1992).

3.2. Batch K-means

If we again solve $\partial I_{emp}[VQ]/\partial \mathbf{w}_i = 0$, we will obtain the sub-optimal values of the codevectors that minimize the functional $I_{emp}[VQ]$:

$$\mathbf{w}_j^{\text{Batch}} = \frac{1}{N_j}\sum_{l=0}^{N-1} 1(\mathbf{x}_l \in R_j)\,\mathbf{x}_l \quad j = 1,\ldots,K \tag{17}$$

This sub-optimal values are empirical estimators of the conditional expectations of X to belong to each of the Voronoi regions R_j j=1,...,K.

Since the batch K-means uses the Newton optimization method, this algorithm converges very fast. In fact, near attraction basins, its convergence is superlinear (Hestenes, 1980).

3.3. Cyclic online K-means with constant step size

3.3.1. Case K=1

When there is only one codevector, the resulting learning equation is linear so its dynamics can be completely characterized using discrete-time linear analysis tools like the z-Transform (Oppenheim & Schafer, 1989) §4. The discrete-time dynamic system then has the following equation:

$$\mathbf{w}[\mathrm{n}+1]=\mathbf{w}[\mathrm{n}]+\alpha(\mathbf{x}[\mathrm{n}+1]-\mathbf{w}[\mathrm{n}]) \quad \mathrm{n}\geq 0, \alpha>0 \tag{18}$$

The sequence of input patterns **x**[n] is a periodic signal of period N, since we cyclically sample a training set of size N. The relation between the sequences **w**[n] and **x**[n] is defined as

$$\mathbf{w}[\mathrm{n}]=\mathrm{h}[\mathrm{n}]*\mathbf{x}[\mathrm{n}]+(1-\alpha)^{\mathrm{n}}\,\mathrm{u}[\mathrm{n}]\,\mathbf{w}[0] \tag{19}$$

where h[n] is the impulse response of the filter that relates **x**[n] with **w**[n], **w**[0] is the initial value of the codevector, u[n] is the step function and * is the convolution operator. To obtain h[n], we transform the equation (19) into the z-domain, compute H(Z) and anti-transform achieving

$$h[n]=\alpha(1-\alpha)^{n}u[n] \tag{20}$$

This linear system is stable in the Bounded-input bounded-output (BIBO) sense (i.e. if **x**[n] is bounded, **w**[n] is bounded too) if and only if h[n] is absolutely summable (Oppenheim & Schafer, 1989) p. 29,

$$\sum_{\mathrm{n}=-\infty}^{\infty}|\mathrm{h}[\mathrm{n}]|<\infty \tag{21}$$

If |1-alpha|<1, the sum S of the terms of an infinite geometric series is finite so the system is stable. Since alpha>0, alpha must belong to the open interval (0,2) to ensure BIBO stability.

Since **x**[n] is a bounded sequence ($\|\mathbf{x}[n]\|\leq B_X=\|\mathbf{x}_{\text{máx}}\|=\arg\max\|xi\|$), **w**[n] has the following bound

$$\left\|\mathbf{w}[\mathrm{n}]\right\|_{\mathrm{n}>0} = \left\|\mathbf{x}_{\max}\right\| + \left|1-\alpha\right|^{\mathrm{n}} \left\|\mathbf{w}[0]\right\| \tag{22}$$

As the discrete-time system iterates, the initial condition disappears and **w**[n] is confined into a ball of radius ||**x**||.

Now we can solve the convolution operation between **x**[n] and h[n] in equation (19),

$$\begin{gathered} w[n] = \sum_{i=-\infty}^{\infty} \alpha(1-\alpha)^i x[n-i] + (1-\alpha)^n u[n] w[0] = \{x[n-i]=0,\, n \le i < 0\} = \\ \sum_{i=0}^{n-1} \alpha(1-\alpha)^i x[n-i] + (1-\alpha)^n u[n] w[0] \end{gathered} \tag{23}$$

The values of the signal **w**[n] that are of special interest are those in which the algorithm have passed through the training set a multiple number of N times. So we explicitly compute **w**[lN] that yields

$$\mathbf{w}[\mathrm{lN}] = \frac{\alpha\left(1-(1-\alpha)^{\mathrm{lN}}\right)}{1-(1-\alpha)^{\mathrm{N}}} \sum_{\mathrm{i}=0}^{\mathrm{N}-1} \mathbf{x}[\mathrm{N}-\mathrm{i}](1-\alpha)^{\mathrm{i}} + (1-\alpha)^{\mathrm{lN}} \mathbf{w}[0] \tag{24}$$

This equation can be described as the sum of a transient and permanent terms

$$\mathbf{w}[\mathrm{lN}] = \mathrm{A} + \mathrm{B}(1-\alpha)^{\mathrm{lN}} \tag{25}$$

If the system is stable, the transient part vanishes with an exponential rate of exponent 1-alpha as l tends to infinite. Then **w**[lN] gives

$$\lim_{\mathrm{l}\to\infty} \mathbf{w}[\mathrm{lN}] = \frac{\alpha}{1-(1-\alpha)^{\mathrm{N}}} \sum_{\mathrm{i}=0}^{\mathrm{N}-1} \mathbf{x}[\mathrm{N}-\mathrm{i}](1-\alpha)^{\mathrm{i}} \tag{26}$$

One can see that the fixed point of this dynamic system is a weighted mean of the training data that gives more relevance to those samples which are at the end of the memory array. If alpha is not small enough, this behavior can notably affect the performance of the algorithm in the presence of outliers.

We can rearrange the equation (26) using the Taylor expansion of (1-alpha)^i in a series of polynomial terms,

$$\lim_{l\to\infty} \mathbf{w}[lN] = \frac{\alpha}{\sum_{j=1}^{N-1} \frac{N!(-1)^{j+1}}{(N-j)!} \frac{\alpha^j}{j!}} \sum_{i=0}^{N-1} \mathbf{x}[N-i] \left\{ 1 + \sum_{j=1}^{i-1} \frac{i!(-1)^j}{(i-j)!} \frac{\alpha^j}{j!} \right\} \tag{27}$$

If (N-1)α/2<1 (condition 1) we can take only the two first terms of the polynomial expansion,

$$\lim_{l\to\infty} \mathbf{w}[lN] = \frac{1}{N} \sum_{i=0}^{N-1} \mathbf{x}[N-i] - \frac{\alpha}{N} \sum_{i=0}^{N-1} i\,\mathbf{x}[N-i] \tag{28}$$

Then the fixed point **fp** of the learning system tends to the empirical mean of the training data. If each training sample **x**j can be approximated by the empirical mean $\mathbf{x_m}$ (e.g. X is a mixture of radial gaussians) and condition 1 is met then the last equation can be approximated by

$$\lim_{l\to\infty} \mathbf{w}[lN] \approx \left(1 - \frac{\alpha(N-1)}{2}\right) \frac{1}{N} \sum_{i=0}^{N-1} \mathbf{x}[N-i] \approx \frac{1}{N} \sum_{i=0}^{N-1} \mathbf{x}[N-i] \tag{29}$$

An important point in the study of dynamical systems is the persistence of the system under small changes or perturbations in their parameters: the notion of structural stability (Devaney, 1989) §1.9. In our case this means to study how a small change in alpha (α+ε with ε<<α) affects the value of **fp**. If the system suffers a soft change in its dynamics, we then will say that this system is structurally stable and $\mathbf{fp}_\alpha$ will be near to $\mathbf{fp}_{\alpha+\varepsilon}$. We define as a measure of the closeness between two fixed points with similar step sizes the following relative distance d_r

$$d_r(\mathbf{w}_\alpha, \mathbf{w}_{\alpha+\varepsilon}) = \frac{\|\mathbf{w}_\alpha - \mathbf{w}_{\alpha+\varepsilon}\|}{\|\mathbf{x}_{\max}\|} \tag{30}$$

where $\mathbf{x_{max}}$ is the training sample that has the biggest module. It is easy to show that if condition 1 is met, dr<<1 and this implies that this system is structurally stable.(As we will see later, this behavior will can help us to design a decreasing step size function that achieves good convergence rate and precision of computation.)

3.3.2. Case K>1

When the number of codevectors K is greater than 1, the resulting discrete-time dynamic system is highly non-linear due to the winner-takes-all process that is performed for each iteration of the algorithm,

$$\mathbf{w}_j[n+1] = \mathbf{w}_j[n] + \alpha_j\, 1(\mathbf{x}[n+1] \in R_j[n])(\mathbf{x}[n+1] - \mathbf{w}_j[n]) \quad n \geq 0, \alpha_j > 0, j = 1,\ldots,K \qquad (31)$$

Anyway the overall non-linear system can be described by K linear systems

$$\mathbf{w}_j[n_j+1] = \mathbf{w}_j[n_j] + \alpha_j\,(\mathbf{x}_j[n_j+1] - \mathbf{w}_j[n]) \quad n_j \geq 0, \alpha_j > 0, j = 1,\ldots,K \qquad (32)$$

where each linear system has its own discrete time n_j and input sequence $\mathbf{x_j}[n_j]$ that is filled with those training samples of the original training sequence $\mathbf{x}[n]$ that fall in the Voronoi region $R_j[n]$. In this way the non-linearity of the original system has been transferred to each sequence $\mathbf{x_j}$.

The cost function of K-means (equation 6) is divided in regions where training samples are always assigned to the same prototypes. Near attraction basins, the algorithm is inside one of these regions so each prototype always ‘sees’ the same training samples. Hence the K coupled linear systems of equation 6 can be decomposed in K uncoupled linear equations as we have just seen in the above section.

Convergence of the algorithm from any starting point can be also studied when we consider the conditions that ensure that on-line K-means was a line search method (Dennis & Schnabel, 1989)§4.2 that minimises a (globally defined) cost function using a gradient descent approach.

The idea is then to compare the behaviour of on-line K-means at the end of each epoch and a (batch) gradient version. If $(N_j - 1)\alpha < 2$ j=1,...,K (where N_j is the number of training samples assigned to $\mathbf{w}_j$), it is easy to show that the algorithm effectively performs (some kind of) batch gradient descent inside regions of constant assignation of samples to prototypes. Then, we only might ensure that the algorithm makes a correct transition between these regions. This implies the fulfilment of several technical conditions (Dennis & Schnabel, 1989)§4.2. However, if the algorithm performs gradient descent on the regions, good transitions are not observed in practice. (See for more details (Bermejo, 2000)).

If the above considerations are fulfilled the on-line K-means will converge to a similar local minimum of the mean quantization error as its counterpart batch version. Then, the algorithm minimizes a global error function, we will observe two very different phases in the dynamics:

- *Phase 1.* Formation of the Voronoi Regions: the system goes from the initial point of departure to an attraction basis of a local minimum of the error function.
- *Phase 2.* Final convergence: the system finally converges from the origin of the attraction basis to the local minimum.

Once the Voronoi Regions are formed each sequence $\mathbf{x_j}[n_j]$ is N_j periodic, where N_j is the number of training samples that fall in the Voronoi region R_j, so we can compute the resulting fixed point using the results for K=1,

$$\lim_{l_j \to \infty} \mathbf{w}_j[l_j N_j] = \frac{\alpha_j}{1-(1-\alpha_j)^{N_j}} \sum_{i=0}^{N_j-1} \mathbf{x}_j[N_j - i](1-\alpha_j)^i \tag{33}$$

3.4. Online K-means with variable step size

3.4.1. Case K=1

General expressions for K=1, when the step size is variable and we sample the training set without imposing any restrictions, can be also derived. In this general case, the discrete-time dynamic system has the following equation:

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \alpha[n](\mathbf{x}[n+1] - \mathbf{w}[n]) \qquad n \geq 0, \alpha > 0 \tag{34}$$

Computing $\mathbf{w}[n]$ when n takes the value of different multiples of N, we can derive a recurring relation between $\{\mathbf{w}[lN], l>1\}$, $\mathbf{w}[0]$ and $\{\mathbf{x}[i], i=1,...,N\}$:

$$\mathbf{w}[N] = \mathbf{w}[0]A_0 + S_0 \tag{35a}$$
$$\mathbf{w}[2N] = \mathbf{w}[N]A_N + S_N = \mathbf{w}[0]A_0 A_N + S_0 A_N + S_N \tag{35b}$$
$$\mathbf{w}[3N] = \mathbf{w}[2N]A_{2N} + S_{2N} = \mathbf{w}[0]A_0 A_N A_{2N} + S_0 A_N A_{2N} + S_N A_{2N} + S_{2N} \tag{35c}$$
$$\vdots$$
$$\mathbf{w}[lN] = \mathbf{w}[0]\prod_{j=0}^{l-1} A_{jN} + \sum_{j=0}^{l-1} S_{jN} \prod_{m=j+1}^{l-1} A_{mN} \tag{35d}$$
$$A_{jN} = \prod_{i=1}^{N}(1-\alpha[i+jN]), \quad S_{jN} = \sum_{i=1}^{N} \mathbf{w}[i+jN]\alpha[i+jN]\prod_{m=i+1}^{N}(1-\alpha[m+jN]) \tag{35e}$$

In the cyclic case, S_{jN} can be simplified using the fact that $\mathbf{x}[n]$ is a periodic signal of period N.

The transient part of equation (35d) asymptotically vanishes if e.g. $\alpha[n]$ is a positive non-increasing function and $0<\alpha[0]<1$, since $\prod_{j=0}^{l-1} A_{jN}$ is bounded by the following expression

$$\prod_{j=0}^{l-1} A_{jN} = \prod_{j=0}^{l-1}\prod_{i=1}^{N}(1-\alpha[i+jN]) \le \prod_{j=0}^{l-1}(1-\alpha[jN])^{N} \le (1-\alpha[0])^{lN} \tag{36}$$

that converges to zero when l → ∞,

$$\lim_{l\to\infty}\prod_{j=0}^{l-1} A_{jN} \le \lim_{l\to\infty}(1-\alpha[0])^{lN} = 0 \tag{37}$$

Here the asymptotic value of **w**[n] is much less clear than in the case of the constant step size. Nevertheless, the important thing to stress is that **w**[∞] is a weighted averaging of the sequence {**x**[i], i=1...N} that depends on the shape of the step size $\alpha[n]$ and the way of sampling the training data.

3.4.2. Case K>1

Following the same arguments of the section 3.3.2 and using equations (35d) and (35e), we can derive a similar expression than equation (33) where each codevector has its own sequence of training samples.

4. Generalization properties

The K-means learning model approximates locally RV **x** by

$$\mathbf{x} \approx VQ(\mathbf{x}) = \mathbf{w}_j = \sum_{i=1}^{K} 1(\mathbf{x} \in R_i)\mathbf{w}_i \tag{38}$$

where $\mathbf{w}_j$ is the nearest codevector to **x**

The expected quantization error will measure how well we approximate for all possible cases,

$$L = E_X\left[\|\mathbf{x} - VQ(\mathbf{x})\|^2\right] \tag{39}$$

It is easy to show that this error function can be decomposed as the sum of an approximation error and an estimation error,

$$L = E_X\left[\left\|\mathbf{x} - VQ(\mathbf{x})\right\|^2\right] = E_X\left[\left\|\mathbf{x} - VQ_{opt}(\mathbf{x})\right\|^2\right] + E_X\left[\left\|VQ_{opt}(\mathbf{x}) - VQ(\mathbf{x})\right\|^2\right] \tag{40}$$

$$VQ_{opt}(\mathbf{x}) = \sum_{i=1}^{K} 1(\mathbf{x} \in R_i) E_{X|R_i}(\mathbf{x}|R_i)$$

The approximation error is the error induced by the kind of model that we use to approximate RV **x** and can be described by

$$E_X\left[\left\|\mathbf{x} - VQ_{opt}(\mathbf{x})\right\|^2\right] = \sum_{j=1}^{K} P(X \in R_j) E_{X|R_j}\left[\left\|\mathbf{x} - E_{X|R_j}(\mathbf{x}|R_j)\right\|^2\right] = \sum_{j=1}^{K} P(X \in R_j)\operatorname{trace}\left(K_{X|R_j}\right) \tag{41}$$

The estimation error is due to the use of training sets of finite size and is defined by

$$E_X\left[\left\|VQ_{opt}(\mathbf{x}) - VQ(\mathbf{x})\right\|^2\right] = \sum_{j=1}^{K} P(X \in R_j)\left\|E_{X|R_j}(\mathbf{x}|R_j) - \mathbf{w}_j\right\|^2 \tag{42}$$

It is minimized when the codevectors $\{\mathbf{w}_j, j=1,...,K\}$ are the conditional expectations. This can only happen if $\mathbf{w}_j^*$, the fixed points of the dynamical system, are the empirical conditional means $\hat{E}_{X|R_{ji}}(\mathbf{x}|R_j)$ and $Nj \rightarrow \infty$ due to the convergence of the means to the expectations by the law of large numbers.

Finally a third kind of error can be considered due to the optimization process if the learning systems do not converge to a global minimum of the quantization error and/or their fixed points are not the empirical means (as in the case of the on-line K-means).

5. Experimental Results

In the experimental part of our work, we will study the real convergence properties of the on-line K-means algorithm using artificial data and how good is the linearized model of K-means near attraction basis. Since the most simple and intuitive expressions have been derived for constant step size and cyclic sampling of the training set, we will only perform simulations in this particular case.

Training data was sampled from a 2-dimensional normal distribution with **0** mean and the identity matrix as a covariance matrix. Two training sets with size N=50 and 500 have been generated. Ten runs with different initial values of the set of prototypes have been made given a training set, constant step size α and number of centroids K. These initial values are randomly chosen from the training set. Each run consists of executing the algorithm until the K fixed points have been reached or 30000/K epochs have passed without convergence. The simulations make use of alpha values from 0.00005 to 1.9 using multiples of 1.2.

We have focussed in collecting four measures from experiments. The first, called E1, quantifies the error between the linearized K-means and the real K-means. It is the average squared error between the fixed points achieved in simulations and the theoretic fixed points using equation (33). E1 is given by

$$E_1 = \frac{1}{10K} \sum_{j=1}^{10} \sum_{i=1}^{K} \left\| \mathbf{p}_{\text{empirical}}{}_i^j - \mathbf{p}_{\text{theoretic}}{}_i^j \right\|^2 \tag{43}$$

The second one (E2) measures the difference between on-line and batch K-means since it averages the Euclidean distance between on-line and batch fixed points using the Voronoi regions computed with the on-line K-means:

$$E_2 = \frac{1}{10K} \sum_{j=1}^{10} \sum_{i=1}^{K} \left\| \mathbf{p}_{\text{empirical}}{}_i^j - \mathbf{p}_{\text{batch}}{}_i^j \right\|^2 \tag{44}$$

We also display the number of epochs in order to the algorithm converges. In all these first three measures, we have assigned to them the value –1 if the algorithm did not converge before 30000/K epochs. Finally we indicate if the algorithm converges or does not before these number of epochs.

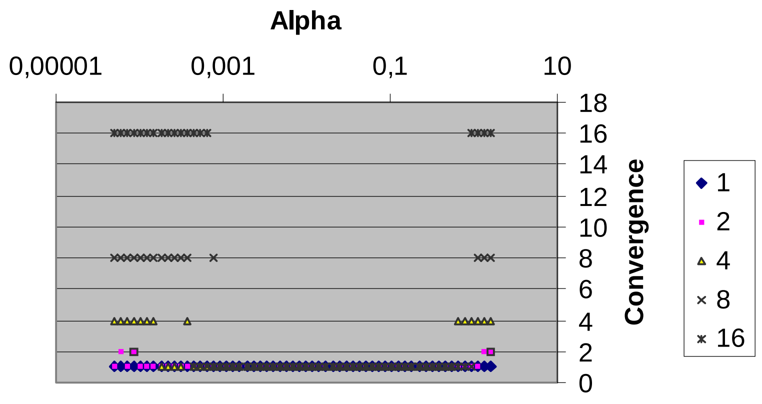
Before we start introducing the simulation results, we present a factor of merit FM_i associated to the codevector $\mathbf{w_i}$ that we will use in order to explain the empirically-observed behavior of the algorithm. This factor of merit has the following expression:

$$FM_i = N_i \, \alpha_i \quad \text{with } N_i >> 1 \tag{45}$$

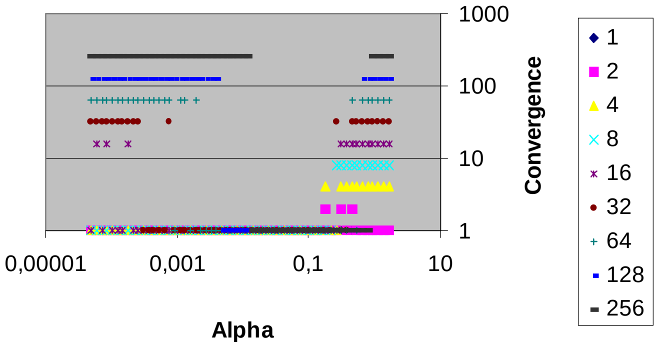
where N_i is the number of training samples that fall in the Voronoi region associated to $\mathbf{w_i}$ and α_i is the step size applied to the update equation of $\mathbf{w_i}[n]$. In fact, the necessary condition to ensure a good estimation of the empirical mean is fulfilled when $FM_i<2$ and $N_i>>1$.

Figure 1 displays the cases in which the algorithm converges in all ten repetitions. There are three regions that have a different behavior. For a $FM_i<<2$, the algorithm converges very slowly. (We can observe, for instance, the left-side of figures 2a and 2b where E1 departures from –1 to 0 indicating that the algorithm converges before 30000/K epochs depending on the initial points are near to attraction basis.) By contrast when $FM_i>2$, the algorithm does not converge to fixed points. In these cases, periodic points appear, as we have checked. If $FM_i<2$ convergence is guaranteed and the convergence rate logarithmically increases as α does (figure 2). As K increases, we need a greater value of α to maintain the same convergence speed, since FM_i decreases because of less training samples fall in a growing number of Voronoi regions. The epochs-to-converge curve in a logarithmic scale is a 'noisy' decreasing line. Its envelope is a line with a negative slope since the number of steps to eliminate a given percentage of the transient part is inversely proportional to FM, as one can check from equation (25). The noise is because 'finite-sample' K-means does not form equally probable Voronoi regions. As a consequence, N_i cannot be approximated by N/K and some codevectors converge before than others. In this way, we do not measure an average number of epochs but the epochs of the codevector with the lowest N_i that randomly varies from the mean N/K.

E1 is always equal to zero when K=1, since the proposed model is exact. In the region of $FM_i<2$ for K>1, as one can see in figure 3, E1 moderately increases as FM_i does, indicating that the linearized model of K-means can serve as a good description of the real dynamic system. Although, as FM_i increases the fixed points are perturbed versions of those predicted by our model. Hence, K-means near attraction basis could be described as a perturbed version of a linearized K-means. Also in this region, E2 increases with FM_i (see figure 4), as one can expect from equation (33).

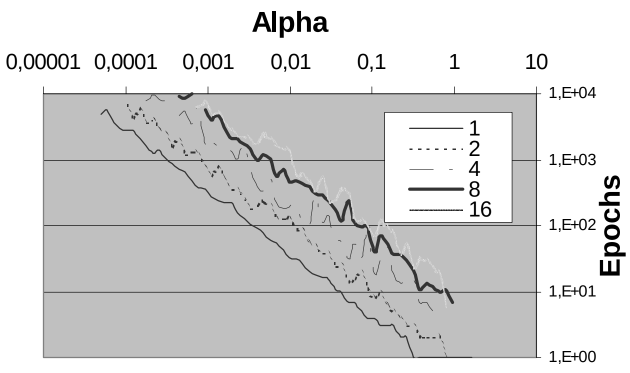


a)

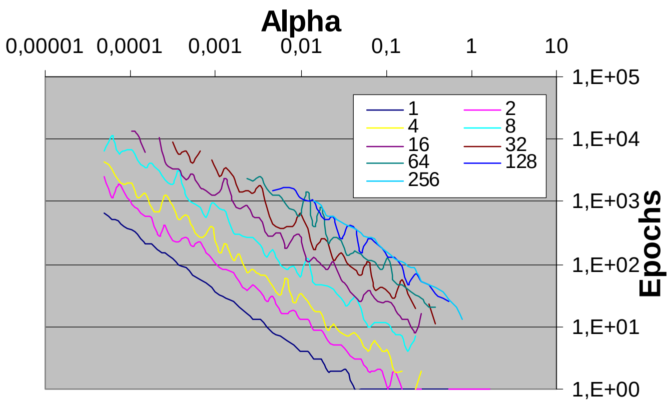


b)

Fig.1. Existence of convergence of the algorithm toward fixed points as function of the constant step size α: a) N=50 and b)N=500. If the algorithm converges, we mark with a value of 1. Otherwise we mark with the value of K.

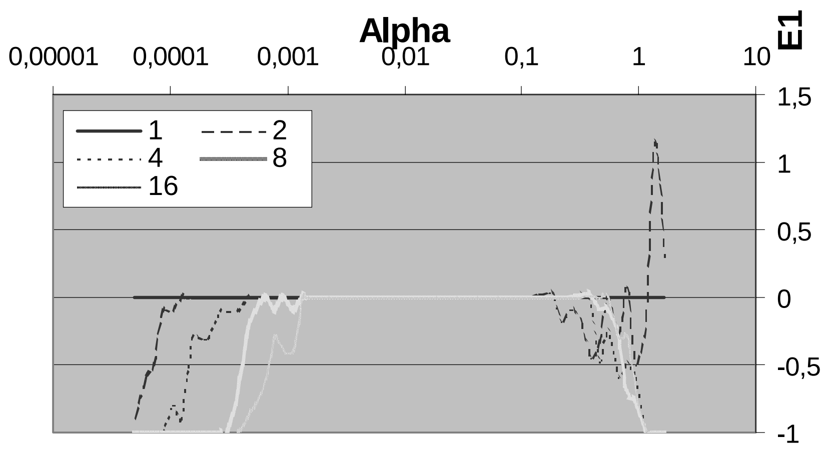


a)

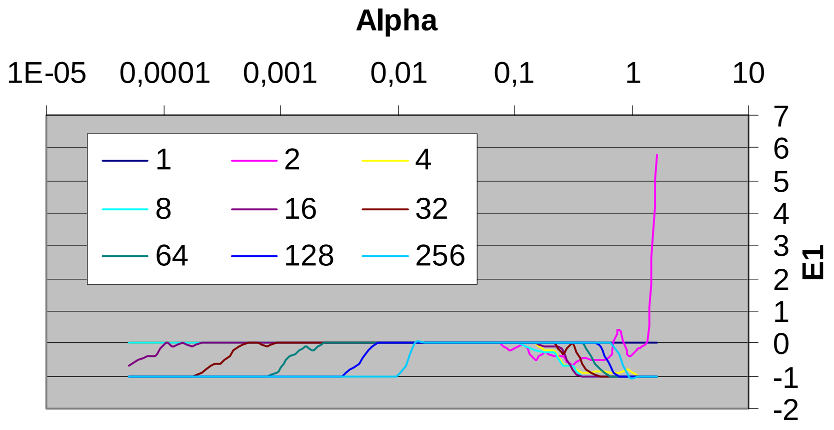


b)

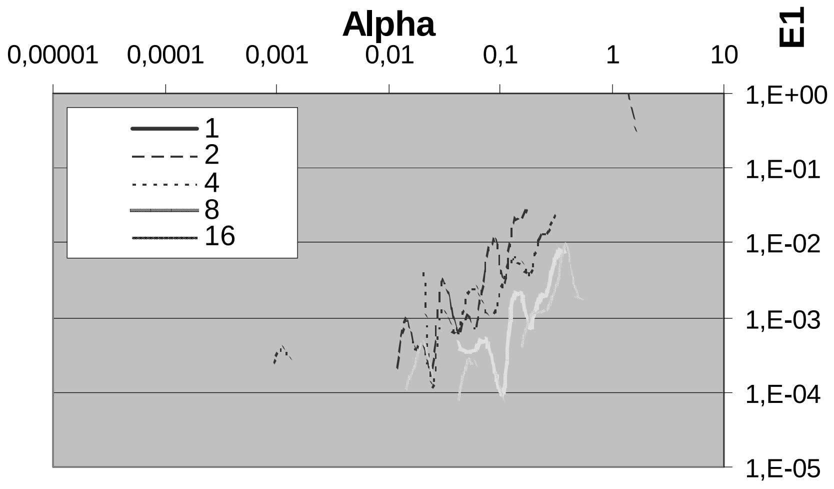
Fig.2. Number of epochs in order to the algorithm converges. We only show those cases where the algorithm converges in the ten repetitions. a) N=50 and K=1,2,4,8 and 16, b) N=500 and K=1,2,4,8,16,32,64,128,256.


Alpha
0,00001
0,0001
0,001
0,01
0,1
1
10
E1
1,5
1
0,5
0
-0,5
-1
1
2
4
8
16

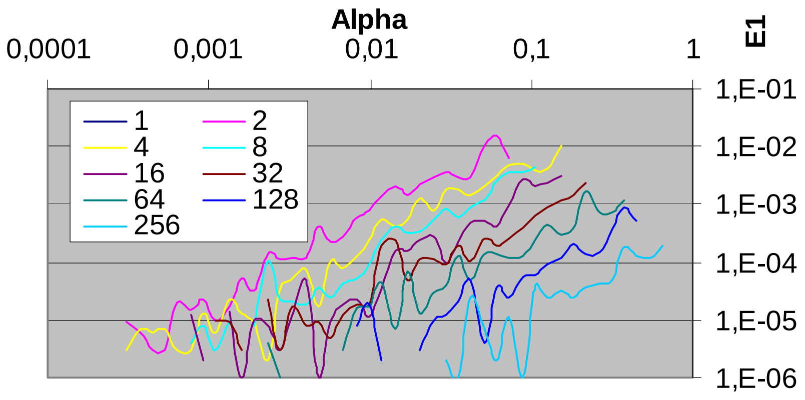
a)


Alpha
1E-05
0,0001
0,001
0,01
0,1
1
10
7
6
5
4
3
2
1
0
-1
-2
E1
1
2
4
8
16
32
64
128
256

b)

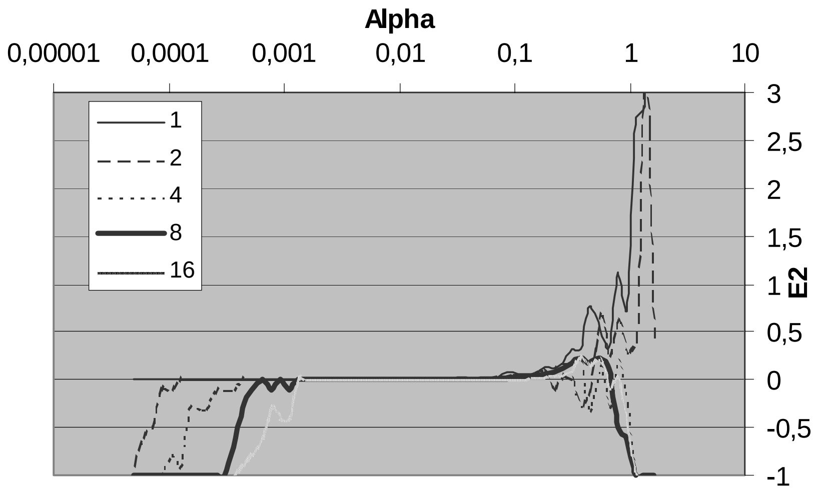

Alpha
0,00001
0,0001
0,001
0,01
0,1
1
10
E1
1,E+00
1,E-01
1,E-02
1,E-03
1,E-04
1,E-05
1
2
4
8
16

c)

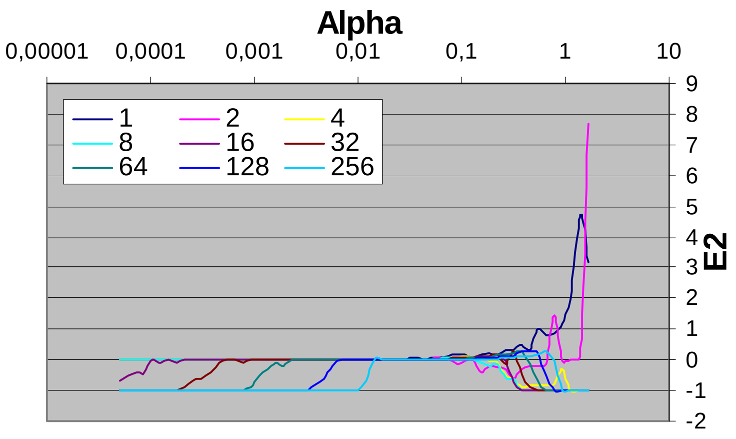


d)

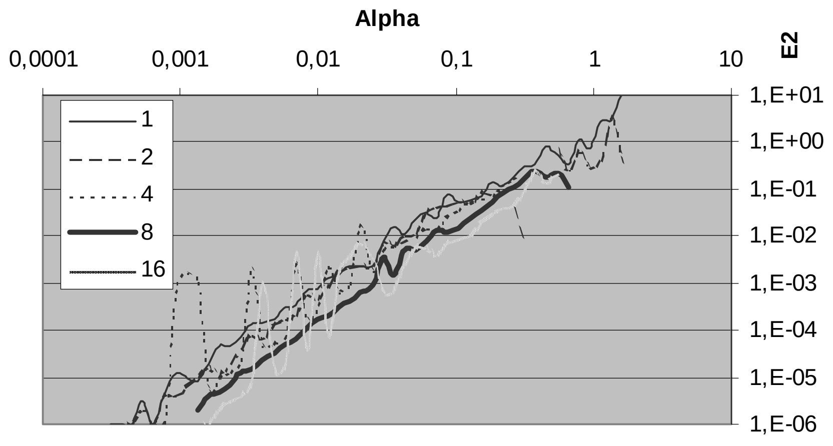
Fig.3. Averaged error between the linearized and real K-means (E1): a) and b) E1 in all the cases for N=50 and N=500 respectively, c) and d) E1 in those cases where the algorithm converges in the ten repetitions for N=50 and N=500 respectively.



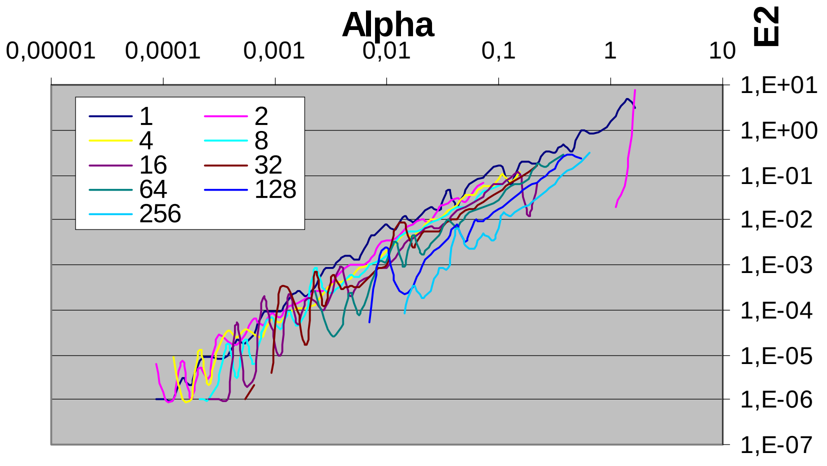
a)



b)



c)



d)

Fig.4. Averaged error between the on-line and batch K-means given the same configuration of the Voronoi regions (E2): a) and b) E2 in all the cases for N=50 and N=500 respectively, c) and d) E2 in those cases where the algorithm converges in the ten repetitions for N=50 and N=500 respectively.

6. Discussion

6.1. On-line K-means for constant step size and cyclic sampling.

The step size α notably affects the fixed points $\mathbf{p_j}$ and its stability. But it is the relation between of α and N_j (the number of training points that are used to compute the fixed point $\mathbf{p_j}$) which determines the behavior and value of $\mathbf{p_j}$. If this ratio is small enough (e.g. <2), the fixed points are stable and their values tend to be the empirical estimators of their counterpart optimal points. Otherwise, the fixed points softly tend to get away from the estimators. Then, each training example contributes in a different way since the fixed point is also a function of the position of each training example in the memory array. In this way, if we use these algorithms in the presence of outliers, the solutions could in some cases be notably worse than batch solutions. Specifically, outliers have a major impact if they are stored at the end of the data array.

Another interesting point is that the advantage of cyclic on-line approaches over batch versions in terms of convergence speed when there is a high degree of redundant information in the training set must be questioned. For example, if we simply replicate the training data once, the number of iterations to achieve convergence approximately stands, given a fixed α. Hence, the number of epochs decreases by a half. This confirms empirical evidence (Bottou & Bengio, 1995). In contrast, batch version needs in this case the same number of epochs to reach the fixed points so the number of iterations duplicates. But if α is fixed and N grows, since each FM_i increases, the fixed points tend less to the desired empirical estimators. So, some precision must be lost at the expense of better convergence speed. Therefore, it exists a tradeoff between the convergence speed and the precision in order to achieve a proper value of alpha for each statistic computed by the K-means algorithm. This tradeoff is controlled by a factor of merit that is a function of α and N_j and can be summarized with the following statement: If we want fast convergence speed, we cannot achieve good precision and viceversa.

6.2. On-line K-means with variable step size.

The way of sampling affects the fixed points of K-means. If the sampling is deterministic, one can derive a closed formula for the fixed points as we have done in the case of cyclic sampling. Anyway, fixed points are an average weighted means that depend on the way of sampling and the step size function (equation 35d).

We have analyzed in depth an interesting particular case that can give some insight of the behavior of the K-means algorithm that use more complex samplings and step size functions. For cyclic sampling and constant step size, fixed points depend on the relation between α and the number of training samples that fall in each Voronoi region. It seems reasonable to think that a similar relationship can hold when the step size is a function that *softly* decreases. Therefore, one can think that the typical and uncontrolled way of decreasing the step size is far from an adequate policy to compute a good estimate of conditioned expectations in a fast way. For instance, we could take advantage of the structural stability of the K-means algorithm when alpha is constant, since a soft variation of the step size provokes a soft variation of the fixed points. Hence, we could apply a strategy of repetitive convergence using a constant alpha that is decreased each time we achieved a fixed point.

6.3. On-line vs. Batch.

Similar conclusions can be derived for other on-line learning algorithms like Kohonen's LVQ1 algorithm (Bermejo, 2000) and principal component neural networks. This could indicate a very general way of conduct of on-line learning systems when they deal with finite training sets. On-line versions could be more interesting than batch versions before they get to global attraction basis since their noise injection can avoid to be caught by a local minimum. By contrast, batch versions seem more robust once the algorithm is near to the final solution. So, in practice one could start running the on-line version and then switch to the batch version in order to converge.

7. Conclusions

General expressions of the finite-sample convergence of the on-line K-means algorithm have been presented where the fixed points of the K-means are an average weighted conditioned means that depend on the training data, the step size function and the way of sampling. In particular, we have derived a closed formula for cyclic presentation and constant step size using a linear model valid near the attraction basis of the non-linear discrete-time dynamical system. In fact, we have observed from simulations that the proposed model is a simplified version of a more complex based on a perturbation of it. It is left for future research to study the causes that originate this perturbation.

Our study in detail for cyclic sampling and constant step size, remarks the importance of the factor of merit $FM_i=\alpha_i N_i$ in the behavior of the dynamics and convergence of each codevector $\mathbf{w}_i$. If $FM_i<2$, the algorithm converges and the optimization error is small. Otherwise, the dynamical system cannot converge a stable solution or if it does, the optimization error can be arbitrary large. Besides, FM_i controls the trade-off between the convergence rate and the optimization error since if we want fast convergence we loss precision in the computation of the estimators and viceversa.

Lastly, the analytic and experimental results in this on-line learning algorithm and others (Bermejo, 2000) could serve as a guideline for more complex on-line learning systems like back-propagation. In accordance with our work, on-line versions could be better as start-up procedures, possibly because their dynamics are more 'noisy' and can serve to escape from local minimums. To end the learning process, the use of batch versions seems preferred since they are more robust in presence of outliers and sometimes make use of better optimization methods.

References

Bermejo, S. (2000). Finite-Sample Convergence Properties of the LVQ1 algorithm, the BLVQ1 algorithm and the GLVQ1 algorithm, in Bermejo, S. Learning with Nearest Neighbour Classifiers, Ph.D. Thesis, Barcelona: Universitat Politècnica de Catalunya.

Benveniste, A., Métivier, M., & Priouret, P. (1990). Adaptive Algorithms and Stochastic Approximations, Berlin: Springer-Verlag.

Y. Bengio. (1991). Artificial Neural Networks and Their Application to Sequence Recognition,. Ph.D. thesis. Department of Computer Science, McGill University.

Bishop, C. M. (1995). Neural Networks and Pattern Recognition, Oxford: Oxford University Press.

Bottou, L. (1998). Online Learning and Stochastic Approximations, in David Saal (Ed.) Online Learning and Neural Networks, Cambridge: Cambridge University Press.

Bottou, L. & Bengio, Y. (1995). Convergence Properties of K-means, Advances in neural processing systems 8. Boston, MA: MIT Press.

Bottou, L. (1998). Online Learning and Stochastic Approximations, in David Saal (Ed.) Online Learning and Neural Networks, Cambridge: Cambridge University Press.

Dennis Jr., J.E. & Schnabel, R.B. (1989). A View of Unconstrained Optimization in Nemhauser, G.L., Rinnooy Kan, A.H.G. & Todd, M.J. (Eds.) Optimization, Amsterdam: North-Holland.

Devaney, R. L. (1989). An introduction to Chaotic Dynamical Systems, Reading: Addison-Wesley.

Devroye, L., Györfi, L. & Lugosi, G. (1996). A Probabilistic Theory of Pattern Recognition, Berlin: Springer-Verlag

Gersho, A. & Gray, R.M. (1992). Vector Quantization and Signal Compression, Boston, MA: Kluwer Academic Publishers.

Hestenes, M. (1980). Conjugate Direction Methods in Optimization, Berlin: Springer-Verlag.

Kohonen, T. (1996). Self-organizing Maps , Berlin: Springer-Verlag.

Kosko, B. (1992). Neural Networks and Fuzzy Systems, NJ: Prentice-Hall International.

Ljung, T. & Söderström, T. (1983). Theory and Practice of Recursive Identification, Boston, MA: MIT Press.

Moody, J.E. & Darken, C.J. (1989). Fast learning in Networks of Locally Tuned Processing Units, Neural Computation, 1, 281-294.

McQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations, in: Proc. of the Fifth Berkeley Symposium on Mathematics, Statistics and Probability, 1, 281-296.

Oppenheim, A. V. & Schafer, R.W. (1989). Discrete-Time Signal Processing, NJ: Prentice-Hall.

Wiggins, S. (1991). Introduction to Applied Nonlinear Dynamical Systems and Chaos, New York: Springer-Verlag.