ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 19-26

Using the Kohonen Algorithm for Quick Initialization of Simple Competitive Learning Algorithm

Eric de Bodt[1], Marie Cottrell[2], Michel Verleysen[3]

[1] Université Catholique de Louvain, IAG-FIN, 1 pl. des Doyens,
B-1348 Louvain-la-Neuve, Belgium
and
Université Lille 2, ESA, Place Deliot, BP 381,
F-59020 Lille, France
[2] Université Paris I, SAMOS-MATISSE, 90 rue de Tolbiac,
F-75634 Paris Cedex 13, France
[3] Université Catholique de Louvain, DICE, 3 pl. du Levant,
B-1348 Louvain-la-Neuve, Belgium

Abstract. In a previous paper ([1], ESANN'97), we compared the Kohonen algorithm (SOM) to Simple Competitive Learning Algorithm (SCL) when the goal is to reconstruct an unknown density. We showed that for that purpose, the SOM algorithm quickly provides an excellent approximation of the initial density, when the frequencies of each class are taken into account to weight the quantifiers of the classes. Another important property of the SOM is the well known topology conservation, which implies that neighbor data are classified into the same class (as usual) or into neighbor classes. In this paper, we study another interesting property of the SOM algorithm, that holds for any fixed number of quantifiers. We show that even we use those approaches only for quantization, the SOM algorithm can be successfully used to accelerate in a very large proportion the speed of convergence of the classical Simple Competitive Learning Algorithm (SCL).

1. Simple Competitive Learning and Vector Quantization

The SOM algorithm (as defined by T.Kohonen in [4]), can be seen as an extension of the Simple Competitive Learning Algorithm (SCL). Let us give the definition of the SCL algorithm that we use here. It is defined in most textbooks [3].

Let Ω be the data space (with dimension d), endowed with a density probability function $f(x)$. The data are randomly drawn according to the density $f(x)$ and are denoted by $x_1, x_2,\ldots,x_N,$. The number of desired classes is a priori fixed to be n. The quantifiers $q_1, q_2, \ldots, q_n$ are randomly initialized. At each step t,

- a data x_{t+1} is randomly drawn according to the density $f(x)$;
- the winning quantifier $q_{i*(t)}$ is determined by minimizing the classical Euclidean norm :

$$\| x_{t+1} - q_{i*(t)} \| = \min_j \| x_{t+1} - q_j \| ;$$

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 19-26

- the quantifier $q_{i^*(t)}$ is updated by $q_{i^*(t+1)} = q_{i^*(t)} + \varepsilon(t)\,(x_{t+1} - q_{i^*(t)})$.

where $\varepsilon(t)$ is an adaptation parameter which satisfies the classical Robbins-Monro conditions ($\Sigma\, \varepsilon(t) = \infty$ and $\Sigma\, \varepsilon^2(t) < \infty$).

We observe that this definition is a particular case of the SOM algorithm, when the neighborhood is reduced to zero. Sometimes it is called *0-neighbor Kohonen algorithm*. In the general case, for the SOM algorithm, the updating concerns not only the winning quantifier, but also its neighbors.

The SCL algorithm is in fact the *stochastic or on-line* version of the Forgy algorithm (also called *moving centers algorithm*, Lloyd's algorithm, LBG). See for example [7], [8], [9]. In this version of the algorithm, the quantifiers are randomly initialized. At each step t, the classes C_1, C_2, …, C_n are determined by putting in class C_i, the data which are closer to q_i than to any other quantifier q_j. Then the mean values of each class is computed and taken as new quantifiers, and so on. The Forgy algorithm works off-line as a batch algorithm and at each step all the quantifiers are updated. It also exists an intermediate version of the algorithm, frequently named the K-means method (Mac Queen, [9]). In that case, at each step, only one data is randomly chosen, and the winning quantifier is updated as the mean value of its class.

In the following, we will denote by BVQ (for batch) the Forgy algorithm.

It can be proven and it is well-known that BVQ (as well as any Vector Quantization algorithm) minimizes the so-called distortion, which is exactly the mean quadratic error:

$$\xi_0(f, q_1, q_2, \ldots, q_n) = \sum_{i=1}^{n} \int_{C_i} \|x - q_i\|^2 f(x)\,dx \qquad (1)$$

estimated by

$$\hat{\xi}_0(f, q_1, q_2, \ldots, q_n) = \frac{1}{N} \sum_{i=1}^{n} \sum_{x_j \in C_i} \|x_j - q_i\|^2 \qquad (2)$$

from the data $x_1, x_2, \ldots, x_N$.

Note that the stochastic SCL algorithm also minimizes this distortion, but only in mean value.

Let us denote by q_1^*, q_2^*, …, q_n^* one set of quantifiers which minimizes the distortion. Generally the minimum is not unique and depends on the initial values[1]. At a minimum, each q_i^* is the gravity center of its class C_i, with respect to the density f. In an exact form[2],

[1] To take this into account, we will realize all our comparison between algorithms starting from the same initial points.
[2] These equations are equivalent to the BVQ algorithm.

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 19-26

$$q^*_i = \frac{\int_{C_i} x\, f(x)\, dx}{\int_{C_i} f(x)\, dx} \tag{3}$$

estimated by

$$\hat{q}^*_i = \frac{\sum_{x_j \in C_i} x_j}{\sum_{x_j \in C_i} 1} \tag{4}$$

If we are able to exactly compute these values q^*_i, then it will be possible to precisely evaluate the performances (speed of convergence) of the algorithm. This is the goal of the next section.

2. Optimal values for the one-dimensional case, with known density.

In one-dimensional cases (d = 1), if the set Ω is a real interval, and if the density f is known and well-behaved, it is possible to directly compute the solutions q^*_i, starting from a given set of increasing initial values, by a iterative equation.

As the initial values are ordered, the current values $q_1, q_2, \ldots, q_n$ are still ordered. The classes C_i ($1 \le i \le n$) are therefore intervals defined by $C_i = [a_i, b_i]$, with $a_i = ½ (q_{i-1} + q_i)$ and $b_i = ½ (q_{i+1} + q_i)$, for $1 < i < n$, and $a_1 = \inf(\Omega)$, $b_n = \sup(\Omega)$.

Equations (3) or (4) have no explicit solutions, but it is possible to get the solutions q^*_i, with any desired precision, using numerical iterations. Annex A presents the recurrent equations for the densities $f(x) = 2x$, $3x^2$, e^{-x}. Knowing the optimal location points of the quantifiers, it will now be possible to study the speed at which any Vector Quantization Algorithm converge towards them. For this, we will study the Euclidean distance between the current values of ($q_i(t)$) resulting from some VQ algorithm and the solutions (q^*_i), as a function of the numbers of steps. We define the mean quadratic error

$$D^2(t) = D^2(q(t),q^*) = (1/n) \sum_{1 \le i \le n} (q_i(t) - q^*_i)^2$$

which will be the *error measure* of the Vector Quantization algorithm that we consider. Note that, as stated above, we carefully start from the same initial increasing points for both VQ algorithm and deterministic computation of the (q^*_i).

In practical situations, the error measure $D^2(t)$ decreases to 0 very slowly when using the SCL algorithm.

In Figures 1, 2, 3, we represent the variations of the error measure for both SCL and SOM with 2 neighbors for the three examples of densities. We can see that the 2-neighbors SOM decreases to the optimal values (q^*_i) much quicker than the SCL algorithm, even if it finally converges to its own optimal points (that can be computed

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 19-26

using equations (3) or (4), where $C_i = [a_i, b_i]$, and $a_i = ½ (q_{i-2} + q_i)$ and $b_i = ½ (q_{i+2} + q_i)$), which minimize the generalized distortion (extended to the neighbors) :

$$\xi_2(f, q_1, q_2, \ldots, q_n) = \sum_{i=1}^{n} \int_{\bigcup_{k \in V(i)} C_k} \left\| x - q_i \right\|^2 f(x) dx \qquad (5)$$

where $V(i)$ is the set $\{ i - 1, i, i + 1\}$.

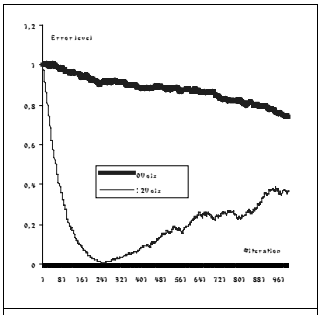

Figure 1 ($2x$)

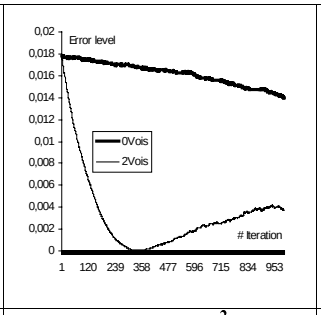

Figure 2 ($3x^2$)

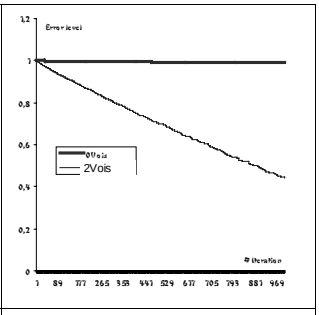

Figure 3 (exp (– x))

See in Annex A some details about the calculations. See also in the Annex A a very simple method to estimate in particular cases *the so-called magnification factor* ([4], [10]). We have also to note that the increase of the total processing time of one iteration when using the 2-neighbors SOM algorithm instead of the SCL algorithm is significantly less than 1%.

So we propose in the next section to use a mixed algorithm, beginning by a SOM algorithm and ending with a SCL algorithm.

3. Hybrid algorithm SOM/SCL

We propose now to use an hybrid VQ algorithm (denoted by KSCL), which consists in an initial phase (a SOM algorithm with ν neighbors), followed by the classical SCL. We compare the value of the error measure after the same number of iterations for KSCL and SCL.

For example, let us fix a total number of iterations T, the initial ordered points $q_1(0)$, $q_2(0)$, …, $q_n(0)$, a constant ε and various probability functions ($f(x) = 2x$ on [0,1], $3x^2$ on [0,1], e^{-x} on [0, +∞ [. Let us also consider the 2-neighbors SOM algorithm, ν = 2.

In Figures 4, 5, 6, we represent for the three probability densities that we took as examples, the variations of the error measure for different KSCL algorithms. We consider 4 cases where the 2-neighbors SOM algorithm is used during 0%, 30%; 60%, 90% of the total number of iterations T.

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 19-26

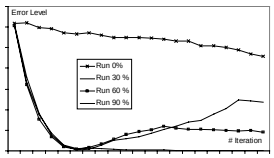

Figure 4 (2*x*)

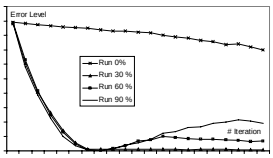

Figure 5 ($3x^2$)

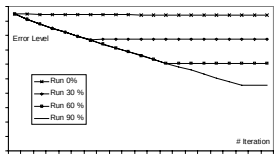

Figure 6 (exp (– *x*))

We can observe that until some step, the 2-neighbors algorithm accelerates a great deal the decrease of the error measure. In all cases, using too early the SCL algorithm slows down the decrease. But even after the optimal point for substituting SCL to SOM, the performances remain better than those of the pure SCL. It is also clear that determining the optimal step for using SCL instead of SOM strongly depends on the probability density. So in practical situations, it will be necessary to experimentally adjust it.

However, in any cases, (we experiment it for other probability densities, real data and several values of the number of neighbors ν), we can conclude that *the SOM algorithm can work as an efficient initialization of the SCL algorithm to accelerate the convergence and improve the performances.*

4. Conclusion

This fact is not surprising and was conjectured by some authors, because the neighbors act as a noise with respect to the algorithm without neighbor. In fact, there are a lot of algorithms in which the noise plays the role of a temperature in the same way as in Simulated Annealing. This temperature is positive at the beginning to help avoiding local minima and decreases to 0 after realizing a first rough optimization.

5. Acknowledgements

The authors are grateful to Jean-Claude Fort and Gil Pagès for fruitful discussions about the topics of this paper.

References

[1] de Bodt E., Verleysen M., Cottrell M., *Kohonen maps versus vector quantization for data analysis*, ESANN'97, M.Verleysen Ed., D Facto, Bruxelles, 211-218, 1997.

[2] Gersho A., *Asymptotically optimal block quantization,* IEEE Trans. Inf. Theory, 25, 373-380, 1979.

[3] Hertz J., Krogh A., Palmer R., Introduction to the Theory of Neural Computation, *Santa Fe Institute,* 1991.
[4] Kohonen T., Self-organizing maps, *Springer*, Berlin, 1995.
[5] Kohonen T., *Computation of VQ and SOM Point Densities Using the Calculus of Variations,* submitted to Neural Computation, 1998.
[6] Kohonen T., *Private Communication,* 1998.
[7] Linde Y., Buzo A., Gray R.M., *An algorithm for vector quantizer design*, IEEE Transactions on Communications, vol. COM-28, no. 1, January 1980.
[8] Lloyd S.P., *Least squares quantization in PCM*, IEEE Transactions on Information Theory, vol. IT-28, no. 2, pp. 129-149, March 1982.
[9] MacQueen J. *Some methods for classification and analysis of multivariate observations*, Proc. Of the Fifth Berkeley Symposium on Math., Stat. and Prob., vol. 1, pp. 281-296, 1967.
[10] Ritter H. and Shulten K., *On the Stationary State of Kohonen's Self-Organizing Sensory Mapping*, Biol. Cybern., 54, 99-106, 1986.
[11] Ritter,H., *Asymptotic level density for a class of vector quantization processes*, IEEE Trans. on Neural Networks, 2, 173-175, 1991.

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 19-26

Annex A

We give below in Table 1, for each of the three examples of probability densities, the distribution function and the recurrent equation (3) which provides the exact calculation of the optimal values (q^*_i).
In fact, this exact computation can also be applied to evaluate *the so-called magnification factor*. Many authors (Gersho [2], Ritter [10], Kohonen [5]) give arguments that show that the vector quantization which leads to a minimization of the distortion ξ_0 corresponds to a discrete distribution which converges asymptotically (when n goes to infinity) to a distribution with density :

$$g_\alpha(x) = A f(x)^\alpha \qquad (6)$$

where A is a constant and $\alpha = 1/3$, in the one-dimensional case.
So for the referred densities, (and for all the densities $f(x) = (p+1)\, x^p$ on [0,1], with $p > -1$), it is possible to write down the theoretical probability function g_α, its distribution function G_α, and the relation which provides an estimation method for the exponent α.

This relation is based on the following remark : the theoretical distribution function G_α can be estimated by the empirical distribution function $\hat{G}_\alpha$ defined by

$$\hat{G}_\alpha(q_i) = \frac{i}{n}, \text{ for } 1 \le i \le n$$

So the optimal values (q^*_i) do verify this relation for each i, and this leads to a very accurate estimation of α, for the three studied densities, using a simple linear regression as written down in the last column. All these regression models are satisfied with a correlation coefficient equal to 1. This method to estimate the exponent α is very accurate, because it uses the exact computation of the optimal quantifiers (q^*_i) and there is no noise as in the stochastic computation of these points.

See in Table 2, the estimations that we get for different numbers of quantifiers (n = 12, 25, 50, 100, 200, 500).

This method can also be used to estimate the exponent α for the SOM with 2 or more neighbors, as computed by Ritter in [11]. See for example Kohonen [6] who uses some similar method. The generalization is very easy, it is sufficient to use the corrected values for a_i and b_i. For example for 2 neighbors, $a_i = ½\ (q_{i-2} + q_i)$, $b_i = ½\ (q_{i+2} + q_i)$ and we get approximately $\alpha = 0.6$, as derived by Ritter [11].

However, it is important to take into account that all these computations (as well as the theoretical arguments in Gersho [2], Ritter [10, 11], or Kohonen [5, 6]) rely on the assumption that the limit distribution of the quantifiers (q^*_i) is unique. But in fact, this result is not evident, and very difficult to prove. What classes of probability distributions satisfy the uniqueness is an open question so far.

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 19-26

Density f	Distribution Function	a_0	b_0	q_i	Density g_α	G_α	Relation
$(p+1)x^p$ on [0,1] $(p > -1)$	x^{p+1}	0	1	$q_i = \frac{p+1}{p+2}\frac{b_i^{p+2} - a_i^{p+2}}{b_i^{p+1} - a_i^{p+1}}$	$(p\alpha+1)x^{p\alpha}$	$x^{p\alpha+1}$	$ln\,(i/n) = (p\alpha+1)\,ln\,q_i$
$2x$ on [0,1]	x^2	0	1	$q_i = \frac{2}{3}\frac{b_i^3 - a_i^3}{b_i^2 - a_i^2}$	$(\alpha+1)x^\alpha$	$x^{\alpha+1}$	$ln\,(i/n) = (\alpha+1)\,ln\,q_i$
$3x^2$ on [0,1]	x^3	0	1	$q_i = \frac{3}{4}\frac{b_i^4 - a_i^4}{b_i^3 - a_i^3}$	$(2\alpha+1)x^{2\alpha}$	$x^{2\alpha+1}$	$ln\,(i/n) = (2\alpha+1)\,ln\,q_i$
e^{-x} on [0,+∞[	$1-x^{-1-\varepsilon}$	0	+∞	$q_i = \frac{a_i e^{-a_i} + e^{-a_i} - b_i e^{-b_i} - e^{-b_i}}{e^{-a_i} - e^{-b_i}}$	$\alpha\, e^{-\alpha x}$	$1 - e^{-\alpha x}$	$-ln\,(1-i/n) = (1-2\alpha)\,ln\,q_i$

Table 1 ($a_i = ½\,(q_{i-1} + q_i)$ and $b_i = ½\,(q_{i+1} + q_i)$).

Density	n=12	n=25	n=50	n=100	n=200	n=500
$2x$ on [0,1]	0.20	0.25	0.29	0.31	0.32	0.33
$3x^2$ on [0,1]	0.26	0.30	0.31	0.32	0.33	0.33
e^{-x} on [0,+∞[	0.43	0.39	0.36	0.34	0.34	0.33

Table 2