An analysis of the GLVQ algorithm

A.I. Gonzalez, M. Graña*, A. D'Anjou
Dpto. CCIA[1], Universidad Pais Vasco/EHU
Apartado 649, 20080 San Sebastián
e-mail: ccpgrrom@si.ehu.es
*IEEE member

Abstract: Generalised Learning Vector Quantization (GLVQ) has been proposed in [5] as a generalisation of the Simple Competitive Learning (SCL) algorithm. The main argument of GLVQ proposal is its superior insensitivity to the initial values of the weights (codevectors). In this letter we show that the distinctive characteristics of the definition of GLVQ disappear outside a small domain of applications. GLVQ becomes identical to SCL when either the number of codevectors grows or the size of the input space is large. Besides that, the behaviour of GLVQ is inconsistent for problems defined on very small scale input spaces. The adaptation rules fluctuate between performing descent and ascent searches on the gradient of the distortion function.

1 Introduction

The GLVQ algorithm has been proposed in [5] as a generalisation of the LVQ algorithm. The definition of LVQ given in [5] really corresponds to what is usually known as SCL in the neural network literature. To clarify terms, the formal definitions of SCL, LVQ, and GLVQ are reviewed in section 2.

According to the authors of [5], GLVQ is related to the Self-Organizing Map (SOM), (due to Kohonen [3]) in the sense that the updating of a neighbourhood of the winning codevector is the mechanism used to escape from the local minima of the distortion. The authors claim that all the codevectors will be updated using the GLVQ rules, unless there is a perfect matching of the input and winning codevector ($\|\mathbf{x}_t - \mathbf{w}_{i,t}\|^2 = 0$). In that case, the authors note, GLVQ reduces to SCL. We show, in section 3.1, that GLVQ is identical to SCL, even in the case of very imperfect matching, for clustering problems defined in non-small spaces or that imply the searching for a non-small number of clusters. On the other hand, in section 3.2 it is shown that perfect matching is not a sufficient condition for GLVQ to become identical to SCL. Even worse situations may happen, as, for some very small input spaces, GLVQ may become inconsistent. Section 4 shows how inside very small input spaces GLVQ rules may fluctuate between going down and up the distortion function.

The derivation of GLVQ follows from the proposition of a loss function, closely related to the distortion. The codevector updating rules are derived as a local (stochastic) gradient descent along this loss function. Some parallelism can be found in this method of definition, and ensuing rules, with Soft Competition [6,7]. In the case of Soft Competition, the purpose is to minimise the cross entropy (Kullback distance) of the input distribution and the distribution modelled by the codevectors, taken as a mixture of Gaussian distributions whose means are the codevectors. As with GLVQ, the Soft Competition rules

---

[1]Work supported by the Departamento de Economia de la Excma. Diputación de Guipuzcoa, and project PGV9220 of the Gobierno Vasco, Departamento de Educación, Universidades e Investigación.

update all the codevectors after the presentation of each input vector. Soft-Competition uses as neighbourhood coefficients the estimations of the conditional probabilities of the input for classes represented by the codevectors. Soft Competition has the additional problem of the estimation of the variances of the Gaussian distributions, but it does not share the limitations of GLVQ that will be discussed in this letter. It will become clear that much of the trouble is due to the definition of the loss function, as we refer to it looking for explanations of undesired limit forms of the GLVQ rules.

Section 2 gives the definitions of SCL, LVQ and GLVQ. Section 3 gives the conditions for GLVQ to behave identically to SCL. Section 4 discusses the conditions for inconsistent behaviour of GLVQ. Section 5 summarises our conclusions.

2 Definition of SCL, LVQ and GLVQ

We will denote $\mathbf{x}_t$ the input vector at time t, $\mathbf{w}_{i,t}$ are the codevectors (weights) at time t, α_t is the learning rate (gain), and $\Delta\mathbf{w}_{i,t}$ is the adaptation of $\mathbf{w}_{i,t}$ after presentation of input $\mathbf{x}_t$. The definition of LVQ given in [5] coincides with what is known in the neural networks' literature as the Simple Competitive Learning (SCL) rule [1, 2, 3, 4]:

$$\begin{aligned} &\Delta\mathbf{w}_{i,t} = \alpha_t\left(\mathbf{x}_t - \mathbf{w}_{i,t}\right) \quad \text{if } \left\|\mathbf{x}_t - \mathbf{w}_{i,t}\right\|^2 = \min_k \left\|\mathbf{x}_t - \mathbf{w}_{k,t}\right\|^2 \\ &\Delta\mathbf{w}_{k,t} = 0 \qquad \text{for } k \neq i \end{aligned} \tag{1}$$

SCL is the basic form of unsupervised competitive learning, and can be derived as a stochastic gradient descent of the distortion:

$$\begin{aligned} J_e &= \sum_{t=1}^{m}\sum_{i=1}^{N} \left\|\mathbf{x}_t - \mathbf{w}_i\right\|^2 \delta_i(\mathbf{x}_t, \mathbf{w}_i) \\ \delta_i(\mathbf{x}, \mathbf{w}) &= \begin{cases} 1 & \left\|\mathbf{x} - \mathbf{w}_i\right\|^2 = \min_k \left\|\mathbf{x} - \mathbf{w}_k\right\|^2 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

where *m* is the sample size and *N* is the number of codevectors. The LVQ algorithm was introduced by Kohonen [3, 4] as a *supervised* learning algorithm, which assumes a priori knowledge about the classes to which the input vector and the winning codevector belong. Allowing *class(**x**)* to denote the class to which **x** belongs, then the formal definition of LVQ reads as follows:

$$\begin{aligned} &\Delta\mathbf{w}_{i,t} = \alpha_t\left(\mathbf{x}_t - \mathbf{w}_{i,t}\right) \quad \text{if } \left\|\mathbf{x}_t - \mathbf{w}_{i,t}\right\|^2 = \min_k \left\|\mathbf{x}_t - \mathbf{w}_{k,t}\right\|^2 \quad \text{and } \text{class}(\mathbf{x}_t) = \text{class}(\mathbf{w}_{i,t}) \\ &\Delta\mathbf{w}_{i,t} = -\alpha_t\left(\mathbf{x}_t - \mathbf{w}_{i,t}\right) \quad \text{if } \left\|\mathbf{x}_t - \mathbf{w}_{i,t}\right\|^2 = \min_k \left\|\mathbf{x}_t - \mathbf{w}_{k,t}\right\|^2 \quad \text{and } \text{class}(\mathbf{x}_t) \neq \text{class}(\mathbf{w}_{i,t}) \\ &\Delta\mathbf{w}_{k,t} = 0 \qquad \text{for } k \neq i \end{aligned} \tag{3}$$

In what follows, we will refer to SCL instead of LVQ. We will assume that all the references to LVQ made in [5] were intended to SCL. This means that the main purpose of the definition of GLVQ was to improve Simple Competitive Learning trying to make it insensitive to initial conditions.

The GLVQ codevector adaptation rules are formally written as follows (with a slight change in notation from the formulas in [5]):

$$\begin{aligned}
&\Delta \mathbf{w}_{i,t} = \alpha_t \left(\mathbf{x}_t - \mathbf{w}_{i,t} \right) C_1 \qquad \text{if } \left\| \mathbf{x}_t - \mathbf{w}_{i,t} \right\|^2 = \min_k \left\| \mathbf{x}_t - \mathbf{w}_{k,t} \right\|^2 \\
&\Delta \mathbf{w}_{j,t} = \alpha_t \left(\mathbf{x}_t - \mathbf{w}_{j,t} \right) C_2 \qquad \text{for } j \neq i \\
&C_1 = 1 - \frac{1}{D_t} + C_2 \qquad C_2 = \frac{\left\| \mathbf{x}_t - \mathbf{w}_{i,t} \right\|^2}{D_t^2} \qquad D_t = \sum_{j=1}^{N} \left\| \mathbf{x}_t - \mathbf{w}_{j,t} \right\|^2
\end{aligned} \tag{4}$$

The values of the coefficients C_1 and C_2 perform the switch between the GLVQ and SCL: C_1=1 and C_2=0 imply that GLVQ is identical to SCL. According to [5], the formal derivation of GLVQ follows from the stochastic gradient descent of a loss function defined as follows.

$$L = \sum_{t=1}^{m} \sum_{i=1}^{N} g_i(\mathbf{x}_t, \mathbf{w}) \left\| \mathbf{x}_t - \mathbf{w}_i \right\|^2 \quad g_i(\mathbf{x}, \mathbf{w}) = \begin{cases} 1 & \text{if } \delta_i(\mathbf{x}, \mathbf{w}) = 1 \\ \left[\sum_{j=1}^{N} \left\| \mathbf{x} - \mathbf{w}_j \right\|^2 \right]^{-1} & \text{otherwise} \end{cases} \tag{5}$$

3 Convergence of GLVQ to SCL

We will use the term "convergence" in a very broad sense, meaning the identity of behaviour of both algorithms under certain conditions. The main statement of [5] is that GLVQ is insensitive to initial conditions (initial codevectors) and that, so, it behaves better than Simple Competitive learning for clustering applications. The authors did not realise that GLVQ becomes identical to SCL algorithm when some parameters of the clustering problem grow, and that this happens rather unexpectedly. On top of that, the conditions assumed in [5] for GLVQ convergence to SCL appear to be insufficient. Two extreme situations condition the convergence landscape: imperfect and perfect matching. In the first case, the desired behaviour of GLVQ is the significant updating of all the codevectors, whereas SCL only updates the winner. In the second case, the desired behaviour is the same as SCL. In both cases, GLVQ can behave against the desires of their designers. In this section we discuss when and why this happens.

3.1 Imperfect matching: undesired convergence

Let us assume the occurrence of imperfect matching in the sense that all the codevectors are almost at the same distance of the input. This situation roughly corresponds to the initialisation of the codebook with values right outside of the convex hull of the sample, or when the input is a wildshot (far away from any other input). In this case, the distance of the input to the winner codevector is close to the distance to any other codevector:

$$\left\| \mathbf{x}_t - \mathbf{w}_{i,t} \right\|^2 \cong \left\| \mathbf{x}_t - \mathbf{w}_{k,t} \right\|^2 \tag{6}$$

Then, the normalisation factor D_t can be assumed to be:

$$D_t \cong N \left\| \mathbf{x}_t - \mathbf{w}_{i,t} \right\|^2 \tag{7}$$

Within these approximations, the expression of the C_1 and C_2 coefficients can be rewritten as follows:

$$C_1 \cong 1 - \frac{1}{N\left\|\mathbf{x}_t - \mathbf{w}_{i,t}\right\|^2} + C_2 \qquad C_2 \cong \frac{1}{N^2\left\|\mathbf{x}_t - \mathbf{w}_{i,t}\right\|^2} \qquad (8)$$

In the case of imperfect matching, GLVQ is meant to behave quite differently from SCL, that is: C_1 must be significantly smaller than 1 and C2 must be significantly larger than 0. However, as the number of clusters or the expected magnitude of the distance grows, it is obvious from the above expressions that C_1 converges to 1. Simultaneously, C_2 converges even faster to 0.

$$C_1 \xrightarrow[\|.\|^2 \to \infty]{N \to \infty} 1 \qquad C_2 \xrightarrow[\|.\|^2 \to \infty]{N^2 \to \infty} 0 \qquad (9)$$

These expressions summarise the sensitivity of GLVQ to the number of classes (codevectors) and the average magnitude of the distance on the input space. Increasing the number of classes forces the undesired convergence GLVQ to SCL. Large input spaces imply large values of $\left\|\mathbf{x}_t - \mathbf{w}_{i,t}\right\|^2$ which also force this convergence. The effect of large input spaces can be also deduced from a close look at the loss function that GLVQ tries to minimise. It can be seen that $L_{\mathbf{x}}$ is bounded as follows:

$$J_e(\mathbf{x}) < L_{\mathbf{x}} \le J_e(\mathbf{x}) + 1 \qquad (10)$$

Where $J_e(\mathbf{x}) = \sum_{i=1}^{N} \left\|\mathbf{x} - \mathbf{w}_i\right\|^2 \delta_i(\mathbf{x}, \mathbf{w}_i)$ is the distortion introduced by quantizing input $\mathbf{x}$ with the codevector $\mathbf{w}_i$ (δ_i retains the meaning given in (2)) When $J_e(\mathbf{x}) >> 1$ the loss function becomes the distortion, and then, the stochastic gradient descent on $L_{\mathbf{x}}$ becomes the SCL. In some clustering applications such as Vector Quantization of grey level images, $[0,255]^{4x4}$ or $[0,255]^{8x8}$ are standard input spaces. Obviously, even for extraordinarily good codebooks, in those applications $J_e(\mathbf{x}) >> 1$. In fact, we started trying to apply GLVQ to the problem of image Vector Quantization. We were surprised by the identity of results produced by GLVQ and SCL. The search for explanations led us to the results reported in this letter.

The above discussion means that GLVQ can become identical to SCL in the case of imperfect matching. That means that outside some restricting conditions (small number of classes and small scale input spaces) GLVQ loses its more appealing property (the insensitivity to initial conditions) because it unexpectedly becomes identical to SCL.

3.2 An illustrative experiment

A simple experiment was carried out to illustrate the foregoing analysis, and to explore the sensitivity of the GLVQ coefficients to the number of codevectors and the scale of the input space. Simulated samples from a mixture of four Gaussian distributions were generated as the sample input. Although the authors in [5] give some discussions about learning rate scheduling, we have found that this is a matter of secondary importance, that does not influence the inherent qualitative features of the algorithms.

The first experiment was addressed to assess the convergence of GLVQ to SCL as the number of codevectors grows. The sample points generated fall inside $[-0.6,1.2]^2$ and the initial codebooks (+ in the figures) were generated in a corner of this region. The expected behaviour of GLVQ was to update all the codevectors driving them inside the sample. Fig. 1a shows the trajectories of the codevectors as updated by GLVQ in the case of N=2. Fig. 1b shows the trajectories under SCL. The difference in behaviour of both algorithms, putting aside the quality of the final solution, can be easily verified. This difference disappears as the number of codevectors increases. In the case of N=10, the behaviour of GLVQ (shown in Fig. 1.c), and that of SCL (shown in Fig. 1.d) are almost identical. The stuck codevectors are minimally displaced by GLVQ. When the sample is scaled by a factor of 5, the identity of behaviour of GLVQ and SCL can be appreciated even with only N=2. Fig 2a and 2c show the behaviour of GLVQ with N=2 and 3, resp. Fig 2b and 2d show the behaviour of SCL.

3.3 (Almost) Perfect matching: desired convergence

Let us assume the occurrence of (almost) perfect matching of input and winning codevector. Perfect matching can be naively characterised by the following expression:

$$\left\|\mathbf{x}_t - \mathbf{w}_{i,t}\right\|^2 \cong 0 \qquad (11)$$

The authors in [5] state that this is the only condition that can lead to $C_1=1$ and $C_2=0$ (*desired convergence* of GLVQ to SCL). Let us examine the approximate expressions taken by the coefficients under this assumption:

$$C_1 \cong 1 - \frac{1}{D_t} \qquad (12)$$
$$C_2 \cong 0$$

Another way to characterise the almost perfect matching comes from the consideration of the relative magnitudes of the distances of the codevectors to the input ($\mathbf{w}_{i,t}$ is the winning codevector):

$$\forall k \neq i \left\|\mathbf{x}_t - \mathbf{w}_{k,t}\right\|^2 >> \left\|\mathbf{x}_t - \mathbf{w}_{i,t}\right\|^2 \qquad (13)$$

It can be easily verified that this more general characterisation of almost perfect matching leads to the same expressions (12) for C_1 and C_2. It becomes obvious examining (12), that the magnitude of D_t determines $C_1 \cong 1$, so, (almost) perfect matching is not a sufficient condition for GLVQ to become SCL. Other problem dependent conditions, such as the number of codevectors and the scale of the input space, and the instantaneous positions of the whole codebook determine the value of D_t and, therefore, the desired convergence of GLVQ to SCL, in the case of almost perfect matching.

4 GLVQ inconsistent behaviour

Studying the desired convergence of GLVQ to SCL, we have found the possibility of inconsistent behaviour of GLVQ. It is expected that any codebook design algorithm will perform a gradient descent on the distortion function. Inside very small input spaces GLVQ becomes (temporarily) a gradient ascent search on the distortion function, depending on the instantaneous value of D_t. In general, D_t must be greater than 1 for C_1 to be positive. Positive values of C_1 make (4) a gradient descent of the distortion, whereas

negative values make (4) a gradient ascent of the distortion. Let us examine the two extreme situations of perfect and imperfect matching.

In the approximate expressions (12) obtained in the case of perfect matching, $D_t<1$ clearly imply negative values of C_1. The expression of the GLVQ rules under these circumstances (compare with (1) above) becomes:

$$\begin{aligned} \Delta \mathbf{w}_{i,t} &= -\alpha_t \left(\mathbf{x}_t - \mathbf{w}_{i,t}\right)\left|1 - \frac{1}{D_t}\right| \qquad \text{if } \left\|\mathbf{x}_t - \mathbf{w}_{i,t}\right\|^2 = \min_k \left\|\mathbf{x}_t - \mathbf{w}_{k,t}\right\|^2 \\ \Delta \mathbf{w}_{j,t} &= 0 \qquad\qquad\qquad \text{for } j \neq i \end{aligned} \tag{14}$$

In the case of imperfect matching, the approximate expressions of C_1 and C_2 in terms of D_t, deduced from (7) and (8), are:

$$C_1 \cong 1 - \frac{N-1}{N \cdot D_t} \qquad C_2 \cong \frac{1}{N \cdot D_t} \tag{15}$$

The expression of the GLVQ rules when $D_t < \frac{N-1}{N}$ become

$$\begin{aligned} \Delta \mathbf{w}_{i,t} &= -\alpha_t \left(\mathbf{x}_t - \mathbf{w}_{i,t}\right)\left|1 - \frac{N-1}{N \cdot D_t}\right| \quad \text{if } \left\|\mathbf{x}_t - \mathbf{w}_{i,t}\right\|^2 = \min_k \left\|\mathbf{x}_t - \mathbf{w}_{k,t}\right\|^2 \\ \Delta \mathbf{w}_{j,t} &= \alpha_t \left(\mathbf{x}_t - \mathbf{w}_{j,t}\right)\frac{1}{N \cdot D_t} \quad \text{for } j \neq i \end{aligned} \tag{16}$$

Both (14) and (16) maximise the distortion instead of minimising it. The behaviour of GLVQ becomes inconsistent for problems defined in very small hypercubes, where the probability of finding $D_t<1$ is significant (e.g. $\mathbf{x} \in [0, 1/N]^n$).

Let us return to the loss function that originates GLVQ in search for some explanation of this inconsistent behaviour. Note that it can be rewritten as follows:

$$L_{\mathbf{x}_t} = \left\|\mathbf{x}_t - \mathbf{w}_{i,t}\right\|^2 + \frac{1}{D_t} \sum_{\substack{k=1 \\ k \neq i}}^{N} \left\|\mathbf{x}_t - \mathbf{w}_{k,t}\right\|^2 = J_e\left(\mathbf{x}_t\right) + \frac{1}{D_t} \bar{J}_e\left(\mathbf{x}_t\right) \tag{17}$$

The loss function is composed of the usual distortion $J_e(\mathbf{x})$, plus some kind of global distortion $\bar{J}_e\left(\mathbf{x}_t\right)$ over the non-winning codevectors, weighted by D_t^{-1}. As long as $D_t>1$, the main term in $L_{\mathbf{x}}$ is the distortion $J_e(\mathbf{x})$. However, when $D_t<1$, the main term in $L_{\mathbf{x}}$ is $\bar{J}_e\left(\mathbf{x}_t\right)$. Then the minimisation of the loss function goes after the minimisation of the "global codebook distortion" instead of (and even against to) the minimisation of the true distortion. In fact, (14) tries to decrease $L_{\mathbf{x}}$ through the increase of D_t, something unexpected for an algorithm which is meant to search optimal codebooks.

Sometimes the algorithm will enter an oscillatory behaviour alternating gradient descent steps, which the distortion (and D_t) decrease, with gradient ascent steps, which increase the distortion and D_t. A detailed analytical study of this phenomenon is outside the scope of this letter. To illustrate the inconsistent behaviour of GLVQ we scaled by a 0.1 factor the sample used in section 3.2. We have applied both GLVQ and SCL to this shrunk sample. This time the initial codebook was a set of sample vectors. This choice of initial codebook highlights the unfitness of GLVQ. Fig 3a shows the trajectories of the codevectors under GLVQ. It can be appreciated the repulsive and attractive phases of GLVQ rules. At first, the codevectors are expelled from the sample. The expulsion is followed by a slow return, until they take positions on the surface of a kind of "repulsion ball" that the GLVQ

dynamics seems to generate around the sample. Fig 3b shows the trajectories of the codevectors under SCL. These follow natural (distortion gradient descent) patterns. Finally, going back to fig. 1a, we have verified that its strangeness is due to the fluctuation of D_t around 1.

5 Conclusions

In spite of the elegance of its definition and its ease of computation, GLVQ must be taken with caution from the practical point of view. A previous analysis of the problem that is intended to solve can show if GLVQ preserves in fact its good qualities (insensitivity to initial codebooks), or if it will behave identically to Simple Competitive Learning, or even if the application is impossible.

We have discussed the GLVQ sensitivity to the number of clusters and the input space size. For a small space, we have shown empirically that a not very large number of clusters is enough to lose the GLVQ characteristics. Also, when the input space is enlarged, the number of clusters that make GLVQ identical to SCL decreases drastically. We have shown how very small input spaces can force the inconsistent behaviour (maximise the distortion) of the GLVQ rules. Empirically, the alternance of behaviours of GLVQ appears under mild conditions. Also, we have seen estrange phenomena such the existence of "repulsion balls" at whose core is the sample. All these undesired results are related to the definition of the loss function, from which the GLVQ rules are derived as a stochastic gradient descent.

References

[1] J.Hertz, A.Krogh, R.G.Palmer "Introduction to the theory of Neural Computation". Addison Wesley, 1991.

[2] B.Kosko "Neural Networks and Fuzzy Systems". Prentice Hall, 1992.

[3] T.Kohonen "Self-Organization and Associative Memory". Springer-Verlag, 1989.

[4] T.Kohonen "Learning Vector Quantisation and the Self-Organising Map" in Taylor, Mannionleds "Theory and Applications of Neural Network". Springer-Verlag, 1992 (3d.edition).

[5] N.R. Pal, J.C. Bezdek, E.C.K. Tsao "Generalized clustering networks and Kohonen's self-organizing scheme" IEEE Trans. Neural Networks 1993 vol 4 pp.549-557

[6] Yair E., Zeger K., Gersho A. (1992) "Competitive Learning and Soft Competition for Vector Quantization" IEEE Trans. Sign. Proc. 40(2) pp.294-308

[7] Tomasini, L. (1993) "Apprentissage d´une representation statistique et topologique d´un environment" PhD Thesis, Ecole Nationale Superieure de l´Aeronautique et de l´Espace.