Available online at www.sciencedirect.com
ScienceDirect
NEUROCOMPUTING
Neurocomputing 70 (2006) 475–488
www.elsevier.com/locate/neucom

The regularized LVQ1 algorithm

Sergio Bermejo

Department of Electronic Engineering, Universitat Politècnica de Catalunya (UPC), Jordi Girona 1-3, edifici C4, 08034 Barcelona, Spain

Received 21 October 2004; received in revised form 12 December 2005; accepted 12 December 2005
Communicated by M. Virleysen
Available online 15 May 2006

Abstract

This paper introduces a straightforward generalization of the well-known LVQ1 algorithm for nearest neighbour classifiers that includes the standard LVQ1 and the *k*-means algorithms as special cases. It is based on a regularizing parameter that monotonically decreases the upper bound of the training classification error towards a minimum. Experiments using 10 real data sets show the utility of this simple extension of LVQ1.

Keywords: LVQ1 algorithm; Nearest neighbour classifiers; Online gradient descent; Newton optimization; Pattern recognition

1. Introduction

Nearest neighbour (NN) methods [16] are still among the simplest and most successful ways of solving pattern recognition problems. As several comparative studies on real-world problems [39,29] suggest, NN methods are often very competitive in comparison with more sophisticated and modern algorithms. Their success can be explained from a theoretical point of view, since they converge to Bayes classifier as the number of neighbours K and the number of prototypes M tend to infinite at an appropriate rate for all distributions. Also, with NN methods, there is a slightly higher probability of misclassifications occurring than with Bayes error $\mathrm{Perr}_{\mathrm{B}}$ for K finite and $M\to\infty$ (i.e.: $\mathrm{Perr}_{K\text{-NN}}\leqslant(1+\sqrt{(2/K)})\mathrm{Perr}_{\mathrm{B}}$ and $\mathrm{Perr}_{1\text{-NN}}\leqslant 2\mathrm{Perr}_{\mathrm{B}}$ [17]). These facts, combined with recent advances on memory-based systems [46], lazy methods [1] and local regression [20], have revived the interest for these techniques in the last decade. A plethora of new learning algorithms has been recently studied (e.g. [18,19,36] and other commented in Section 2.2).

NN classifiers are local learning systems [15]: they fit the training data only in a region around the location of an input pattern. Given a pattern $\mathbf{x}$ to classify, the K-nearest-neighbour classification rule is based on the following algorithm:

(i) Find the K nearest patterns to $\mathbf{x}$ in the set of prototypes $\mathbf{P}=\{(\mathbf{m}_j, \mathrm{cl}(\mathbf{m}_j), j=0,\ldots,M-1\}$ where $\mathbf{m}_j$ is a prototype that belongs to one of the classes and $\mathrm{cl}(\mathbf{m}_j)$ is the class indicator variable.
(ii) Establish the classification by a majority vote amongst these K patterns.

NN classifiers allow for a variety of design choices that can be adjusted automatically from data such as the metric d to measure closeness between patterns, the number of neighbours K, the set of prototypes $\mathbf{P}$ and the size M. The most common similarity measurement $d(\mathbf{x},\mathbf{y})$ is the Euclidean distance $d(\mathbf{x},\mathbf{y})=||\mathbf{x}-\mathbf{y}||^2$, while K can be automatically selected using a validation set, although $K=1$ is a common choice due to the following:

(1) Euclidean 1-NN classifiers form class boundaries with piecewise linear hyperplanes. They can therefore be used to solve a large class of classifiers since any border can be approximated by a series of locally defined hyperplanes.

E-mail address: sbermejo@eel.upc.edu.

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2005.12.123

(2) Most of the learning algorithms that compute **P** from training data work with 1-NN classifiers.

Finally, the design of the set of prototypes is the most difficult and challenging task. The simplest method would be to select the whole training set $\mathbf{D}_N = \{(\mathbf{x}_i, \text{cl}(\mathbf{x}_i)),\ i = 0,\ldots,N-1\}$ (where $\mathbf{x}_i$ is a random sample of X and $\text{cl}(\mathbf{x}_i)$) is the class label associated to $\mathbf{x}_i$) as **P**. Nevertheless, this option would result in large memory and execution requirements in large databases. Therefore, in practice, a small set of prototypes of size M (with $M \ll N$) is mandatory. There are three main classes of learning algorithms that are designed to reduce the number of prototypes stored:

(1) *Condensing algorithms*: Since only near class border training data are useful in the classification process, condensing procedures aim to keep those points from training data which form class boundaries [17, Section 19].
(2) *Editing algorithms*: These retain any training patterns that fall inside class borders estimated to be within the same training set. Such patterns tend to form homogeneous clusters because only points at the centre of the natural groups in the data are retained [17, Section 26].
(3) *Clustering algorithms*: It is also feasible to use any NN vector quantization algorithm [23] (e.g. K-means [38]) to form a set of labelled prototypes). Firstly, we obtain a set of unlabelled prototypes from training data using the clustering algorithm. These prototypes can then be used to divide the input space in K-nearest-neighbour cells. Finally, we can assign labels to prototypes according to a majority vote by the training data in each cell [17, Section 21.5]. However, it is also possible to compute labelled centroids using a one-step learning strategy such as learning vector quantization (LVQ) algorithms [30].

As pointed out in [17, Section 19.3], clustering algorithms seem preferable to condensing and editing algorithms for the following reason: if the values of **P** are allowed to be arbitrary, prototypes are not constrained to training points. A more flexible class of classifiers can therefore be designed. However, the most preferable strategy for designing prototypes is to minimize the empirical classification error produced in the training set $\mathbf{D}_N$ [17, p. 311], since generalization error bounds for the 1-NN classifier based on the VC theory [17, Section 19] can be applied. Our work here shows that LVQ1 does not minimize the classification error, but a simple and straightforward generalization that introduces a regularizing parameter monotonically decreases the upper bound of the misclassification rate of the 1-NN classifier and thus improves the classification results obtained by LVQ1.

The paper is organized as follows. In Section 2, a review of LVQ1 and its basic limitations is presented. Section 3 introduces and analyses RegLVQ1, a regularized form of the LVQ1 algorithm, which controls the upper bound of the training error. Section 4 includes a comparative empirical study of RegLVQ1 and other learning algorithms for 1-NN classifiers in ten real-world problems. Finally, some conclusions are given in Section 5.

2. Limitations and extensions of the LVQ1 algorithm

2.1. Limitations of LVQ1

Suppose we have N observation pairs $\mathbf{D}_N = \{(\mathbf{x}_i, cl(\mathbf{x}_i)),\quad i = 0,\ldots,N-1\}$, where $\mathbf{x}_i \in \Re^p$ is a random pattern that belongs to one of the c classes and $\text{cl}(\mathbf{x}_i)$ is the class label associated to $\mathbf{x}_i$. The aim of a learning algorithm for a Euclidean nearest neighbour (NN) classifier is to design a set of labelled prototypes $\mathbf{P} = \left[\mathbf{m}_\mathbf{0}^\mathbf{T} \quad \mathbf{m}_\mathbf{1}^\mathbf{T} \quad \ldots \mathbf{m}_{\mathbf{M\text{-}1}}^\mathbf{T}\right]^\mathbf{T}$ using $\mathbf{D}_N$. There are M Voronoi regions $R_j = \{\mathbf{x} | \|\mathbf{x} - \mathbf{m}_j\| = \min_{i=1,\ldots,K}\|\mathbf{x} - \mathbf{m}_i\|\}$, $j = 0,\ldots,M-1$, where the classifier maps any input pattern that falls within it to the class to which its codevector $\mathbf{m}_j \in \Re^p$ belongs. The LVQ1 algorithm [30] applies to compute **P**, the following on-line update equation (i.e. prototypes are adapted each time an input pattern is presented):

$$\mathbf{m}_j[n+1] = \begin{cases} \mathbf{m}_j[n] + \alpha[n]\big(\mathbf{x}[n+1] - \mathbf{m}_j[n]\big) & \text{if} \quad \mathbf{x}[n+1] \in R_j[n] \quad \text{and} \quad \mathbf{x}[n+1], \mathbf{m}_j \in \text{same class}, \\ \mathbf{m}_j[n] - \alpha[n]\big(\mathbf{x}[n+1] - \mathbf{m}_j[n]\big) & \text{if} \quad \mathbf{x}[n+1] \in R_j[n] \quad \text{and} \quad \mathbf{x}[n+1], \mathbf{m}_j \notin \text{same class}, \end{cases} \quad j = 0,\ldots,M-1, \tag{1}$$

where $\alpha[n]$ is the step size function that belongs to the interval (0,1). As the learning progresses through the repeated application of (1), the number of misclassified samples tends to decrease, since prototypes are moved away from samples of other classes and towards samples belonging to the same class. LVQ1 can be considered a modified version of on-line K-means in which class labels affect the way that the clustering process is performed and on-line (or pattern-based) gradient

descent is used over the following cost function:

$$E_{\mathrm{LVQ1}}(\mathbf{P}) = \frac{1}{2}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} 1\left(\mathbf{x}_i \in R_j\right) 1\left(\mathrm{cl}(\mathbf{x}_i) = \mathrm{cl}\left(\mathbf{m}_j\right)\right)\left\|\mathbf{x}_i - \mathbf{m}_j\right\|^2 - \frac{1}{2}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} 1\left(\mathbf{x}_i \in R_j\right) 1\left(\mathrm{cl}(\mathbf{x}_i) \neq \mathrm{cl}\left(\mathbf{m}_j\right)\right)\left\|\mathbf{x}_i - \mathbf{m}_j\right\|^2, \tag{2}$$

where the $\mathrm{cl}(\mathbf{x})$ function returns the class label of $\mathbf{x}$, and $\mathbf{1}$ (condition) is the indicator function (which is 1 if condition is true and 0 otherwise). As shown in [6], the minimization of Eq. (2) only ensures that the classification error in the training set is less than 50%. This is possible since the minimum points of E_{LVQ1} fulfil that $2N_{m_j} > N_j$ for all $j = 0,\ldots,M-1$ where N_j is the number of training data that fall in Voronoi region R_j, and N_{m_j} are those training samples that fall in R_j belonging to the same class as $\mathbf{m}_j$, so $\sum_{j=0}^{M-1} 2N_{m_j} > \sum_{j=0}^{M-1} N_j = N$ and the classification rate $= \sum_{j=0}^{M-1} N_{m_j} > N/2$.

In spite of the minimum points of E_{LVQ1} do not guarantee that training error is minimized, the average training error with LVQ1 are far smaller from the upper bound of 0.5 which can be understood through the study of the asymptotic convergence of LVQ1 for the infinite sample case. Applying tools of the stochastic approximation theory [3] that consider the convergence of an ordinary differential equation (ODE) with the same asymptotic convergence than (1), we apply the condition of stochastic equilibrium over the ODE, which gives:

$$E_X[(\mathbf{x} - \mathbf{m}_i)1(\mathbf{x} \in R_i)(1(\mathrm{cl}(\mathbf{x}) = \mathrm{cl}(\mathbf{m}_i)) - 1(\mathrm{cl}(\mathbf{x}) \neq \mathrm{cl}(\mathbf{m}_i)))] = \int_{R_i} (\mathbf{x} - \mathbf{m}_i)\left(p\left(\mathbf{x}\middle| C_{\mathrm{cl}(\mathbf{m}_i)}\right) P\left(C_{\mathrm{cl}(\mathbf{m}_i)}\right) - \sum_{\substack{j=1 \\ j \neq \mathrm{cl}(\mathbf{m}_i)}}^{C} p\left(\mathbf{x}\middle| C_j\right) P\left(C_j\right)\right) = 0, \quad i = 0, ..., M-1, \tag{3}$$

where C is the number of classes, $\{P(C_j)\}$ are the (prior) class probabilities, $\{p(\mathbf{x}|C_j)\}$ are the class density functions. (We have omitted the technical details that the learning system must fulfil in order to derive (3). The reader can find them in [35].) Eq. (3) can be considered as the stochastic equilibrium condition of a vector quantizer (see e.g. [32, pp. 228–232]) that places prototypes according to the following density function:

$$p_{\mathrm{LVQ1}}(\mathbf{x}) = p\left(\mathbf{x}\middle| C_{\mathrm{cl}(\mathbf{m}_i)}\right) P\left(C_{\mathrm{cl}(\mathbf{m}_i)}\right) - \sum_{\substack{j=1 \\ j \neq \mathrm{cl}(\mathbf{m}_i)}}^{C} p\left(\mathbf{x}\middle| C_j\right) P\left(C_j\right) \quad \forall \mathbf{x} \in R_i, \quad i = 0, \ldots, M-1. \tag{4}$$

For a two-class problem and $M \to \infty$, Eq. (4) is a probability density function that is zero at Bayes borders. Hence, LVQ1 places prototypes around Bayes borders and consequently the resulting nearest-neighbour classifier estimates Bayes classifier. Fig. 1b shows LVQ1's input density and the prototypes computed with LVQ1 for a two-class pattern recognition problem (Fig. 1a). Observe that the density function is zero at Bayes point (4.5) and prototypes are placed according the LVQ1's density and hence an estimation of the Bayes classifier is finally performed by the nearest-neighbour classifier.

Although LVQ1 in two-class problems performs a clustering process over a probability density function that is zero at Bayes borders, this behaviour is not maintained for problems with a greater number of classes since regions in which the density of points of the majority class is smaller than the sum of the other densities ($2N_{m_i} < N_i$ for any i) cause a maximum of the cost function E_{LVQ1}, as shown in [6]. Hence, LVQ1 gets away from these (repelling) points. If Bayes borders are in these regions of "negative" density of points, a correct placement of prototypes for estimating Bayes borders will be not possible. Fig. 2 illustrates this behaviour in a three-class classification problem.

2.2. Extensions of LVQ1: from clustering to large margin classification

LVQ1 belongs in the class of so-called data-dependent partitioning methods [17, Section 21], in which a set of prototypes for 1-NN classifiers is based on the use of a supervised cluster algorithm in which the partition $\mathscr{P}_N = \bigcup_{i=1}^{M} R_i$ of the input region is obtained (i.e. the input space is divided into M cells) using the estimated cluster centroids to induce $\mathscr{P}_N$ in the following way:

$$R_i = \left\{ \mathbf{x} \middle| D(\mathbf{x}, \mathbf{m}_i) = \min_{j=1,\ldots,M} D\left(\mathbf{x}, \mathbf{m}_j\right) \right\}, \quad i = 1, \ldots, M, \tag{5}$$

where $D\left(\mathbf{x}, \mathbf{m}_j\right)$ is here the Euclidean distance metric, i.e. $D_2\left(\mathbf{x}, \mathbf{m}_j\right) = \left(\sum_{i=1}^{p} \left|x_i - m_j\right|^2\right)^{1/2}$ and the cluster centroids are computed minimizing the cost function E_{LVQ1}. In order to improve the classification accuracy of the resulting nearest neighbour classifier, different cost functions and distance metrics have been proposed in the last decade. As noted in [42], a family of LVQ algorithms (called generalized LVQ or simply GLVQ) can be derived from minimizing the following

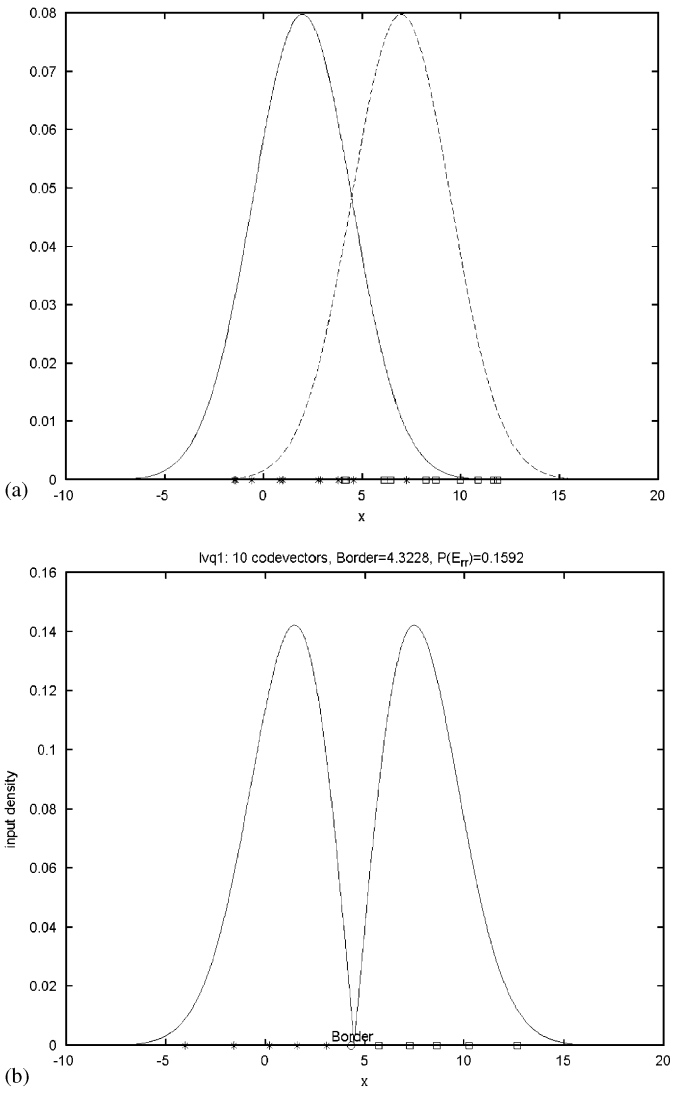

Fig. 1. A two-class pattern recognition problem: (a) Two normal class densities (centred at 2 and 7 with variance of 2.5^2) and 10 random samples of each class (squares and x's) taken from their density functions. Note that the optimal decision point is near 5 (4.5) where both class densities achieve the same value. (b) LVQ1's input density function for the problem of Fig. 2 and the LVQ1's prototypes computed with a training set. These prototypes are denoted with squares and x's. The border point of the resulting nearest-neighbour classifier (4.32) is marked with a circle. Note that prototypes are placed according to the displayed density function.

generalized cost function:

$$E_{\mathrm{GLVQ}}(\mathbf{P}) = \sum_{i=0}^{N-1} f(\mu(\mathbf{x}_i)), \tag{6}$$

where $f(\mu)$ and $\mu(\mathbf{x}_i)$ are functions to be defined. If we only consider the two prototypes nearest to $\mathbf{x}_i$ from different classes in order to define f, i.e. $\mathbf{m}_{\mathrm{cc}}$ is the nearest prototype to $\mathbf{x}_i$ that belongs to the same class and $\mathbf{m}_{\mathrm{wc}}$ is the nearest prototype to $\mathbf{x}_i$ that belongs to a different class, then the GLVQ gradient-descent update equations are:

$$\mathbf{m}_{\mathrm{cc}}[n+1] = \mathbf{m}_{\mathrm{cc}}[n] + \alpha[n]\frac{\partial f}{\partial \mu}\frac{D_{\mathrm{wc}}}{(D_{\mathrm{cc}} + D_{\mathrm{wc}})^2}\left(\mathbf{x}[n+1] - \mathbf{m}_j[n]\right), \tag{7}$$

$$\mathbf{m}_{\mathrm{wc}}[n+1] = \mathbf{m}_{\mathrm{wc}}[n] - \alpha[n]\frac{\partial f}{\partial \mu}\frac{D_{\mathrm{cc}}}{(D_{\mathrm{cc}} + D_{\mathrm{wc}})^2}\left(\mathbf{x}[n+1] - \mathbf{m}_j[n]\right), \tag{8}$$

where $D_{\mathrm{wc}} = D_2^2(\mathbf{x}_i, \mathbf{m}_{\mathrm{wc}})$ and $D_{\mathrm{cc}} = D_2^2(\mathbf{x}_i, \mathbf{m}_{\mathrm{cc}})$. Observe that (7) and (8) have a very intuitive geometrical interpretation: prototypes belonging in the same class as $\mathbf{x}_i$ and in a different class from $\mathbf{x}_i$ come closer and move away, respectively, according to a weighting factor that is now modulated by a function of the distances between these two nearest prototypes

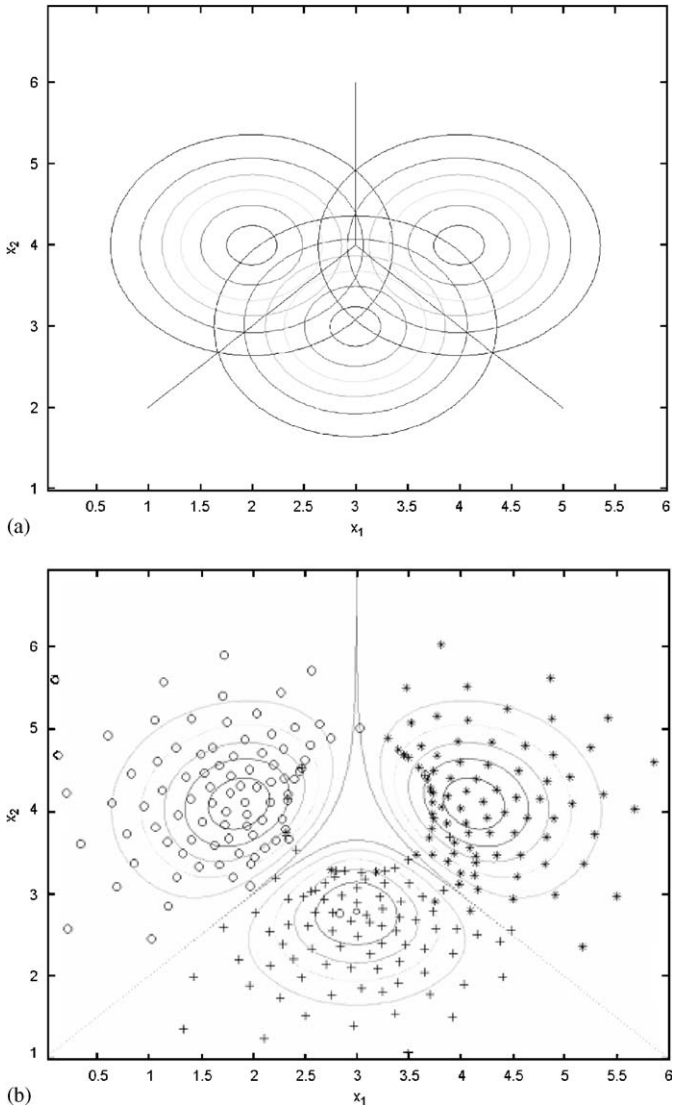

Fig. 2. A 2-D three-class pattern recognition problem: (a) three radial normal class densities (centred at (2,4), (3,3) and (4,4) with variance 0.5) and all classes has the same priors (1/3) and Bayes borders are the lines displayed; (b) prototypes computed with LVQ1 and the contour levels of LVQ1's density function for the three-class problem. Observe that prototypes are placed according to the density displayed and the prototypes are repelled from the valley centred around (3,4) near where there density is zero.

and $\mathbf{x}_i$. In addition, it is worth noting that (7) and (8) resemble a soft version of LVQ. Observe that the LVQ1 algorithm is obtained when $f(\mu) = \mu$ and

$$\mu = \begin{cases} D_{\mathrm{cc}} & \text{if} \quad D_{\mathrm{cc}} < D_{\mathrm{wc}}, \\ -D_{\mathrm{wc}} & \text{if} \quad D_{\mathrm{cc}} > D_{\mathrm{wc}}. \end{cases} \tag{9}$$

A heuristic improvement of LVQ1, the LVQ2.1, which was also introduced by Kohonen, can be obtained if $\mu = D_{\mathrm{cc}} - D_{\mathrm{wc}}$ and $f(\mu)$ is bounded as follows:

$$f(\mu) = \begin{cases} \mu & \text{if} \left|\mu\right| < s, \\ c & \text{otherwise}, \end{cases} \tag{10}$$

where c is a constant and s is a width factor that adjusts the locality of the input regions in which LVQ2.1 updates its prototypes. As shown in [14], LVQ2.1 is a simplified version of an algorithm based on minimizing a cost function that is a continuous approximation to a binary variable, which indicates whether the $\mathbf{x}_i$ pattern has been misclassified. Hence, the good results of LVQ2.1 could presumably be explained to some extent by its relationship to the minimization of the misclassification rate. Other heuristic improvements of LVQ2.1, i.e. LVQ3 (also by Kohonen) and many others, have also been proposed (e.g. [37]), although it remains difficult to assess the ultimate impact of these modifications on the classifiers'

accuracy. It is possible to obtain additional benefits when using LVQ algorithms if we combine them according to some well-known ensemble learning techniques [44], such as: (i) voting [2], (ii) local averaging [9], and (iii) resampling methods like bootstrapping [25].

The soft update of prototypes anticipated by Sato and Yanada [42] when a cost function is minimized as in (6) has been a recurrent feature in many recent approaches to the design of NN prototypes, such as those based on: (i) probabilistic modelling of class densities [4,29,33,43], (ii) the use of self-organizing maps [34], (iii) fuzzified versions of LVQ algorithms [10], and (iv) large margin classification [18,19]. Large margin classifiers [45], which have recently been proposed in the context of statistical learning theory, advocate the use of alternative minimization cost functions in which training patterns are correctly classified with the highest confidence (or maximum margin) instead of minimizing the misclassification rate, or others based on clustering like those reviewed above. Interest in them is theoretically motivated since the large-margin feature ensures tighter bounds on the generalization error of the classifier [45]. In the context of NN classifiers, large margin solutions [18,19] can be computed by minimizing (6) with

$$f(\mu) = -\frac{1}{N}\sum_{j=1}^{C} y_{ji} d_j(\mathbf{x}_i), \tag{11}$$

where $\mathbf{y}_i$ is an indicator variable, i.e. if $\mathbf{x}_i$ belongs to the nth class, the nth coefficient of $\mathbf{y}_i$ is equal to 1, the other coefficients are equal to 0 and a set of discriminant functions $\{d_j(\mathbf{x}),\ j = 1,\ldots,C\}$, which are computed with the classifier's prototypes, fulfil $\sum_{j=1}^{C} d_j(\mathbf{x}_i) = 1$. The cost function using (11) in (6) is a modification of the cross-entropy error function for multiple classes [11, Section 6.9] and can also be related to the Kullback information-theoretic divergence measure that has previously been employed in the context of nearest-neighbour classifiers [48]. The absolute minimum with respect to the $\{d_j(\mathbf{x}_i)\}$ occurs when all the training points are correctly classified with the maximum confidence, that is, when $d_j(\mathbf{x}_i) = y_{ji}$ for all j and i. Moreover, it has been shown in [18,19] that the absolute value of this cost function will converge to the maximum probability of correct classifications when the number of prototypes is large enough to hold $d_j = 1$ for all the points in each R_j for $N \to \infty$. Note that, for different choices of $d_j(\mathbf{x})$, many different algorithms can be derived. In [19], the $\{d_j(\mathbf{x})\}$ have been chosen as estimators of posterior class probabilities based on a mixture modelling yielding

$$d_i(\mathbf{x}) = \hat{P}(C_i|\mathbf{x}) \approx \frac{G\left(\mathbf{x} - \mathbf{w}_W^i; \gamma\right)}{\sum_{i'=1}^{C} G\left(\mathbf{x} - \mathbf{w}_W^{i'}; \gamma\right)}, \quad i = 1,\ldots,C, \tag{12}$$

where $\mathbf{w}_W^j$ is the nearest centre of the mixture model to $\mathbf{x}$ that belongs to class j using the distance metric D, and kernel G is a bounded and even function of X that peaks about $\mathbf{0}$ of the form $G\left(\mathbf{x} - \mathbf{w}_W^j; \gamma\right) = G\left(D\left(\mathbf{x}, \mathbf{w}_W^j\right); \gamma\right)$ where γ denotes the locality control parameters associated with G. For radian Gaussian kernels, i.e. $G(D(\mathbf{x},\mathbf{w});\gamma) = \exp\left(-\|\mathbf{x}-\mathbf{w}\|^2/2\sigma\right)$, the resulting gradient-descent learning algorithm is called *Gaussian learn1NN*. If σ is finite, this algorithm is shown to resemble a fuzzified version of the update equation of LVQ algorithms since the step size α is modulated by a factor that depends on the current value of the discriminant functions. When $\sigma \to \infty$, learn1NN coincides with LVQ2 in regions near class borders. However, an adaptive soft K-NN classifier is presented in [18] where the $\{d_j(\mathbf{x})\}$ are defined as local Parzen window estimates of posterior class probabilities computed using the K nearest prototypes, i.e.

$$d_i(\mathbf{x}) = \hat{P}(C_i|\mathbf{x}) \approx \frac{\sum_{i=1}^{K_l} G\left(\mathbf{m}_i^l - \mathbf{x}; \gamma\right)}{\sum_{j=1}^{C}\sum_{u=1}^{K_j} G\left(\mathbf{m}_u^j - \mathbf{x}; \gamma\right)}, \tag{13}$$

where $\left\{\mathbf{m}_u^j, u = 1,\ldots,K_j\right\}$ are those prototypes of class j among the K nearest prototypes to $\mathbf{x}$, and $K = \sum_{j=1}^{C} K_j$.We note that the (crisp) K-NN algorithm in (13) is recovered in the limit $\sigma \to \infty$. In the particular case $K = 2$, the resulting classifier is equivalent to the 1-nearest-neighbour classification rule, since only two prototypes are involved in the decision. Again, the learning rules for Gaussian kernels resemble a fuzzified version of Kohonen's LVQ algorithm. When hard decisions are employed ($\sigma \to \infty$) and $K = 2$, the gradient-descent learning equations give:

$$\begin{aligned}
&\text{if } \mathbf{m}_i[n], \mathbf{x}[n+1] \in \text{same class and } \mathbf{m}_j[n], \mathbf{x}[n+1] \notin \text{same class},\\
&\quad \mathbf{m}_i[n+1] = \mathbf{m}_i[n] + \frac{\eta}{4}(\mathbf{x}[n+1] - \mathbf{m}_i[n]),\\
&\quad \mathbf{m}_j[n+1] = \mathbf{m}_j[n] - \frac{\eta}{4}\left(\mathbf{x}[n+1] - \mathbf{m}_j[n]\right),
\end{aligned} \tag{14}$$

where $\mathbf{m}_i[n]$ and $\mathbf{m}_j[n]$ are the two nearest prototypes to the pattern $\mathbf{x}[n+1]$. Eq. (14) is just the LVQ2.1 algorithm when $\mathbf{x}[n+1]$ falls in the window defined in the LVQ2 and LVQ3 algorithms. A last and important remark concerning these two classifiers is that as they minimize the number of misclassifications while maximizing the average margin of training samples (e.g. many training samples are classified with high confidence), prototypes in their discriminant functions are based on the extreme data points (or support vectors) of each class. This fact relates them to support vector machines [45].

Finally, other recent efforts in improving LVQ algorithms have focused on learning weighted Euclidean metrics $D_{\mathbf{W}}(\mathbf{x},\mathbf{m}_j)=\|\mathbf{W}(\mathbf{x}-\mathbf{m}_j)\|^2$ where $\mathbf{W}=[w_{ij}]$ is a weighted matrix computed from training data. Two major approaches exist: (a) local learning metrics (e.g. [21,41]), in which the weighted matrix $\mathbf{W}$ depends on $\mathbf{x}$ and $\mathbf{m}$, i.e. $\mathbf{W}(\mathbf{x},\mathbf{m})$, and (b) global learning metrics (e.g. [13,40]), in which $\mathbf{W}$ is constant over the whole input space. In global approaches like the distinction-sensitive approach [40] and relevance determination [13], $\mathbf{W}$ is restricted to positive-defined square diagonal matrices, i.e.

$$w_{ij}=\begin{cases}0, & i\neq j\\ w_{ii}, & i=j\end{cases}\quad i,j=1,\ldots,p, \tag{15}$$

with $w_{ii}>0$. The incorporation of such a matrix ($\mathbf{W}$) as a parameter in the E_{GLVQ} (6) yields a global learning algorithm like the generalized relevance LVQ algorithm [26], in which prototypes $\{\mathbf{m}_i\}$ and $\mathbf{W}$ are both computed using a gradient-descent learning algorithm. Another global learning algorithm is proposed in [8]. It computes a non-square $\mathbf{W}$ (i.e. a $q\times p$ matrix where q is the number of features selected) based on principal components oriented to those projections that improve the classification accuracy of large margin classifiers. In [8], one can see the combination of such an oriented principal component analysis with the adaptive soft K-NN classifier.

3. The regularized LVQ1 algorithm

3.1. The RegLVQ1 cost function

A simple solution to improve the classification rate of LVQ1 leads to the so-called regularized LVQ1 (RegLVQ1) algorithm, which is based on minimizing the following cost function:

$$\begin{aligned}&E_{\mathrm{RegLVQ1}}(\mathbf{P},\lambda)=\frac{1}{2N}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}E_{\mathrm{RegLVQ1}}(\mathbf{m}_j,\lambda)\quad\text{with}\\ &E_{\mathrm{RegLVQ1}}(\mathbf{m}_j,\lambda)=1(\mathbf{x}_i\in R_j)\big(1(\mathrm{cl}(\mathbf{x}_i)=\mathrm{cl}(\mathbf{m}_j))-\lambda 1(\mathrm{cl}(\mathbf{x}_i)\neq\mathrm{cl}(\mathbf{m}_j))\big)\|\mathbf{x}_i-\mathbf{m}_j\|^2,\end{aligned} \tag{16}$$

where a regularizing parameter λ is introduced in the cost function. Note that (16) gives the quantification error performed separately for each class when $\lambda=0$ and it is equivalent than E_{LVQ1} when $\lambda=1$ since, e.g., the resulting gradient-based learning equations are the same in both cases except for an additional scaling factor of $1/N$ on the step size that does not affect its convergence properties. The inclusion of the regularizing parameter λ provides control over the upper bound of the classification error through the minimization of E_{LVQ1}, as shown below. Since the minimum points of E_{RegLVQ1} are achieved when $\nabla E_{\mathrm{RegLVQ1}}(\mathbf{P}^*)=0$ and the eigenvalues of $\mathbf{H}(\mathbf{P}^*)=\nabla^2 E_{\mathrm{RegLVQ1}}(\mathbf{P}^*)$ are positive [28], solving the equation $\nabla E_{\mathrm{RegLVQ1}}(\mathbf{P}^*)=0$ yields:

$$\mathbf{m}_j^*=\frac{\sum_{i=0}^{N-1}1(\mathbf{x}_i\in R_j)\left(1\left(\mathrm{cl}(\mathbf{x}_i)=\mathrm{cl}\left(\mathbf{m}_j^*\right)\right)-\lambda 1\left(\mathrm{cl}(\mathbf{x}_i)\neq\mathrm{cl}\left(\mathbf{m}_j^*\right)\right)\right)\mathbf{x}_i}{\sum_{i=0}^{N-1}1(\mathbf{x}_i\in R_j)\left(1\left(\mathrm{cl}(\mathbf{x}_i)=\mathrm{cl}\left(\mathbf{m}_j^*\right)\right)-\lambda 1\left(\mathrm{cl}(\mathbf{x}_i)\neq\mathrm{cl}\left(\mathbf{m}_j^*\right)\right)\right)}\quad j=0,\ldots,M-1. \tag{17}$$

Due to this, $\mathbf{H}$ can be divided into blocks according to each pair of codevectors:

$$\mathbf{H}=\begin{bmatrix}\dfrac{\partial^2 E_{\mathrm{RegLVQ1}}}{\partial^2\mathbf{m}_0} & \cdots & \dfrac{\partial^2 E_{\mathrm{RegLVQ1}}}{\partial^2\mathbf{m}_0\partial\mathbf{m}_{M-1}}\\ \vdots & \vdots & \vdots\\ \dfrac{\partial^2 E_{\mathrm{RegLVQ1}}}{\partial^2\mathbf{m}_{M-1}\partial\mathbf{m}_0} & \cdots & \dfrac{\partial^2 E_{\mathrm{RegLVQ1}}}{\partial^2\mathbf{m}_{M-1}}\end{bmatrix}. \tag{18}$$

The first and second partial derivatives of E_{RegLVQ1} with respect to codevectors give:

$$\frac{\partial E_{\mathrm{RegLVQ1}}[VQ]}{\partial \mathbf{m}_j} = -\sum_{i=0}^{N-1} 1(\mathbf{x}_i \in R_j)\big(1(\mathrm{cl}(\mathbf{x}_i) = \mathrm{cl}(\mathbf{m}_j)) - \lambda 1(\mathrm{cl}(\mathbf{x}_i) \neq \mathrm{cl}(\mathbf{m}_j))\big)(\mathbf{x}_i - \mathbf{m}_j),$$

$$\frac{\partial E_{\mathrm{RegLVQ1}}[VQ]}{\partial \mathbf{m}_j \partial \mathbf{m}_j} = \begin{cases} \mathbf{0} & \text{if } i \neq j, \\ \Big((1+\lambda)N_{m_j} - \lambda N_j\Big)\mathbf{I} & \text{if } i = j, \end{cases} \tag{19}$$

where $\mathbf{I}$ is the $p \times p$ identity matrix, N_j is the number of training samples that fall in the Voronoi region R_j, and N_{m_j} is the number of training points that fall within R_j and belong to the same class as $\mathbf{m}_j$. Since $\mathbf{H}$ is a constant diagonal matrix, its pK eigenvalues γ_i are $\left\{\gamma_{pj+k} = (1+\lambda)N_{m_j} - \lambda N_j, k = 1,\ldots,p, \quad j = 0,\ldots,M-1\right\}$. Since the eigenvalues of H are all positive, we can write $\left\{(1+\lambda)N_{m_j} - \lambda N_j > 0, k = 1,\ldots,p, \quad j = 0,\ldots,M-1\right\}$ and, accordingly, the minimum points ensure that $N_{m_j} > \lambda N_j/(1+\lambda)$ for all $j = 0,\ldots,M-1$. Then, the number of correct classifications is $\sum_{j=0}^{M-1} N_{m_j} > \lambda/N(1+\lambda)$, and so the minimization of $E_{\mathrm{RegLVQ1}}(\lambda)$ ensures a training error $\mathrm{err}_{\mathrm{T}}(\lambda) \leqslant \max \mathrm{err}_{\mathrm{T}}(\lambda)$ where $\max \mathrm{err}_{\mathrm{T}}(\lambda) = 1/(1+\lambda)$. Minimization of E_{RegLVQ1} will tend to minimize the number of training classification errors for very large values of λ ($\lambda \gg 1$) since $\max \mathrm{err}_{\mathrm{T}}(\lambda_2) < \max \mathrm{err}_{\mathrm{T}}(\lambda_1)$, if and only if $\lambda_2 > \lambda_1$. However, the cost function E_{RegLVQ1} has more chances of being ill conditioned as λ augments, since the number of feasible solutions (i.e. the minimum points of E_{RegLVQ1}) is reduced progressively. Fig. 3 shows this decrease in the number of minimum points for the following two-class 1-D synthetic problem $\mathbf{D}_N = \{(-0.6, \mathrm{b}), (-0.6, \mathrm{b}), (-0.6, \mathrm{b}), (-0.5, \mathrm{a}), (1.0, \mathrm{a}), (1.5, \mathrm{a}), (2.0, \mathrm{a}), (2.5, \mathrm{a}), (3.0, \mathrm{b}), (4.0, \mathrm{a}), (6.0, \mathrm{b})$, which has been solved with two prototypes $\mathbf{P} = (m_1, m_2)$, one for each class. The global minimum point for E_{RegLVQ1} with $\lambda = 1$ (i.e. the solution computed with the LVQ1 algorithm) is $\mathbf{P}^* = (m_1^*, m_2^*) = (0.6, -0.65)$ with a frontier point $F = (m_1^* + m_2^*)/2 = -0.025$ which achieves a classification error of $\frac{3}{11}$ while the minimum error for this classification problem is $\frac{2}{11}$. This minimum error would result from any F between -0.6 and -0.5. As one can observe in Fig. 3, the E_{RegLVQ1} moves $\mathbf{P}^*$ to the left as λ increases. For $\lambda = 1.25$, there is only one single global minimum point corresponding to a $\mathbf{P}^*$ that induces a sub-optimal F with an error of $\frac{2}{11}$. For ensuring an upper bound around $\frac{2}{11}$, λ must be around 4.5. However, this last

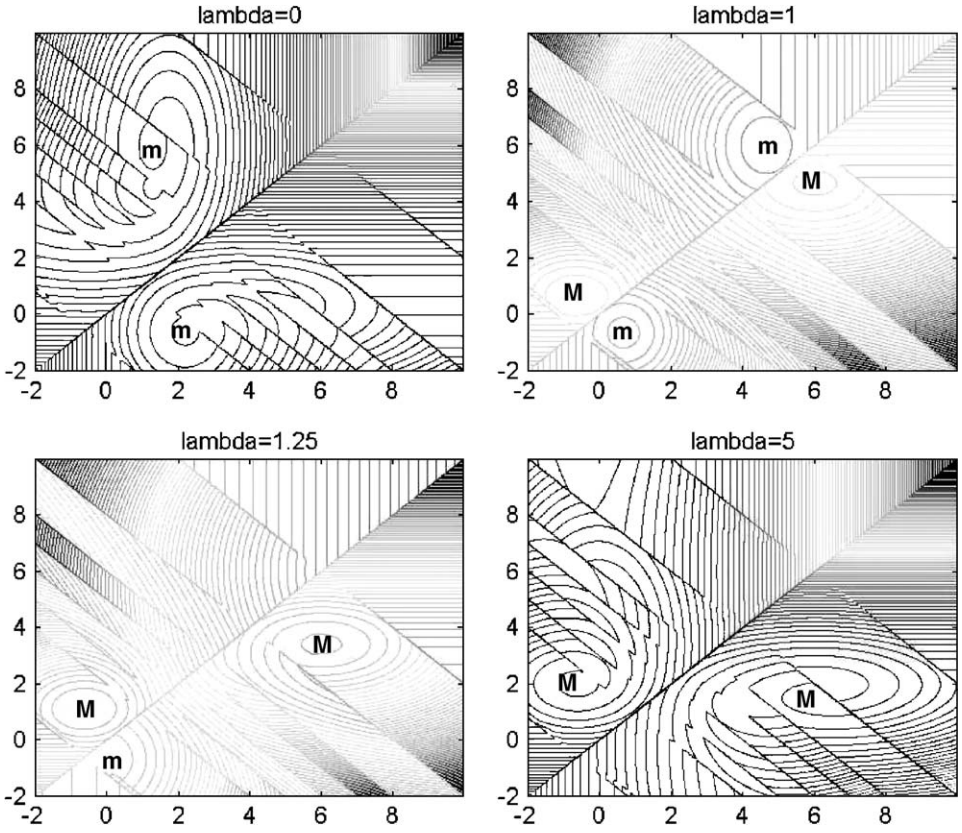

Fig. 3. The effect of the regularizing parameter λ on a cost function $E_{\mathrm{RegLVQ1}}(m_1, m_2)$ for a 1-D synthetic problem. *Note*: m and M denote minimum and maximum points of E_{RegLVQ1}, respectively.

global minimum disappears for $\lambda > 2$, so no feasible solutions exist that achieve the minimum classification error (see bottom right of Fig. 3 when $\lambda = 5$). Therefore, the best classifier obtained with E_{RegLVQ1} does not achieve the minimum error. The optimal set of prototypes using the data set is $\mathbf{P}' = (m_1', m_2') = (-0.5, -0.6)$, which induces a linear classifier and achieves a maximal separation (or margin) between the support vectors of each class. However, we must bear in mind that prototypes computed with E_{RegLVQ1} are placed in proportion to some particular density of points (see (20)) and not on such extreme points as support vectors. This example helps to understand intuitively the effect of the regularizing term in the solutions and also the inherent limitations of any clustering approach to the design of $\mathbf{P}$. More precisely, several considerations can be made in light of the above example.

Firstly, the constraint imposed by the regularizing parameter λ serves as a mechanism for reducing the number of undesired local minimum points of the learning system. It is fairly similar to the weight decay employed in neural networks [11]. Also, note that all the minimum points of E_{RegLVQ1} disappear as λ increases. Consequently, this ill conditioning causes a numerical instability in the learning algorithm for large values of λ.

Secondly, the disappearance of a minimum point of E_{RegLVQ1} can be explained in terms of the Voronoi regions in which the E_{RegLVQ1} is divided. In each region, a permanent assignation of the training samples to the set of prototypes $\mathbf{P}$ is done. However, minimum points only exist in regions in which $\nabla E_{\mathrm{RegLVQ1}}(\mathbf{P}^*, \lambda)$ and the prototypes $\mathbf{P}^*$ computed with (17) implement the assignation of training samples that induce it. If a minimum point exists in a particular Voronoi region of E_{RegLVQ1}, the increase of the regularizing parameter λ by a small amount $\Delta\lambda$ causes a smooth displacement of the minimum, as one can infer from (17). As λ is increased, the minimum point is pushed away from its Voronoi region so it finally disappears. Furthermore, as it can be deduced from the relationship between the minimization of E_{RegLVQ1} and the classification error stated above, an increase of λ causes a monotonic decrease of the misclassification rate in the training set ($\mathrm{err_T}$) toward a limit case in which only one single (and thus global) minimum point exist in the cost function. However, the exact nature of this last minimum (and the $\mathrm{err_T}$ achieved in this case) remains unclear. In other words, for large values of λ, all minima of the cost function may disappear before achieving a minimum point of the classification error in the training set if this point does not fall in the Voronoi region in which this last minimum belongs, as happens in the above example.

Finally, the effect of λ can be understood considering the clustering process performed by the algorithm. Fig. 4 shows RegLVQ1's prototypes for the three-class synthetic problem in Fig. 2. As the minimum points of E_{RegLVQ1} are controlled by λ, ensuring that $N_{m_j} > \lambda N_j/(1+\lambda)$, RegLVQ1's prototypes can be placed in regions near Bayes border in which a density of points of the majority class is smaller than the sum of the other densities, unlike LVQ1's prototypes. In fact, the density function of RegLVQ1 can be obtained, using a similar rationale employed for deriving (4), as:

$$p_{\mathrm{RegLVQ1}}(\mathbf{x}) = \max\left\{ p\left(\mathbf{x}\middle|C_{\mathrm{cl}(\mathbf{m}_i)}\right)P\left(C_{\mathrm{cl}(\mathbf{m}_i)}\right) - \lambda \sum_{\substack{j=1 \\ j\neq \mathrm{cl}(\mathbf{m}_i)}}^{C} p\left(\mathbf{x}\middle|C_j\right)P\left(C_j\right), 0 \right\} \quad \forall \mathbf{x} \in R_i, \quad i = 0, \ldots, M-1. \tag{20}$$

In Fig. 4, we can observe the correspondence between the density of RegLVQ1's prototypes and the contour levels of the density function computed with (20) for the three-class synthetic problem (Fig. 2). It is worth noting that, particularly for this classification problem, distributions with $\lambda < 1$ (as shown in Fig. 2), are helpful. This could be paradoxical bearing in mind our aim of ensuring $N_{mj} \gg N_j$. However, as we previously remarked, the regularizing term affects the upper bound of $\mathrm{err_T}$, i.e. $\max \mathrm{err_T}(\lambda_2) < \max \mathrm{err_T}(\lambda_1)$ for $\lambda_2 > \lambda_1$. This behaviour will reduce the classification error, i.e. $\mathrm{err_T}(\lambda_2) < \mathrm{err_T}(\lambda_1)$ for $\lambda_2 > \lambda_1$, only when it can be improved. In the previous two-class example, $\mathrm{err_T}(1) < \mathrm{err_T}(0) = \frac{5}{11}$ but $\mathrm{err_T}(1) = \mathrm{err_T}(1.25) = \frac{3}{11}$ since the Voronoi region in which the last global minimum belongs cannot improve $\mathrm{err_T}$. For the three-class synthetic problem, three radial Gaussian classes are defined and then Bayes borders are induced with a nearest-neighbour classifier that employs each class centroid as a prototype. Consequently, the optimal density function for RegLVQ1 is $\lambda = 0$, so larger values of λ will not improve this optimal solution. In practice, we have not observed such extreme cases in real databases, since classifiers with $\lambda > 1$ always outperform those computed with $\lambda < 1$.

3.2. Online and batch versions

In the on-line (or pattern-based) version, a scaling factor is introduced in the amount of change of the winning prototype for cases in which its label does not agree with the input pattern's label:

$$\begin{aligned} \mathbf{m}_j[n+1] &= \mathbf{m}_j[n] - \alpha[n]\frac{\partial E_{\mathrm{RegLVQ1}}\left(\mathbf{m}_j[n], \lambda\right)}{\partial \mathbf{m}_j[n]} \\ &= \begin{cases} \mathbf{m}_j[n] + \alpha[n]\left(\mathbf{x}[n+1] - \mathbf{m}_j[n]\right) & \text{if } \mathbf{x}[n+1] \in R_j[n] \text{ and } \mathbf{x}[n+1], \mathbf{m}_j \in \text{sameclass}, \\ \mathbf{m}_j[n] - \lambda\alpha[n]\left(\mathbf{x}[n+1] - \mathbf{m}_j[n]\right) & \text{if } \mathbf{x}[n+1] \in R_j[n] \text{ and } \mathbf{x}[n+1], \mathbf{m}_j \notin \text{sameclass}, \end{cases} \quad j = 0, \ldots, M-1, \end{aligned} \tag{21}$$

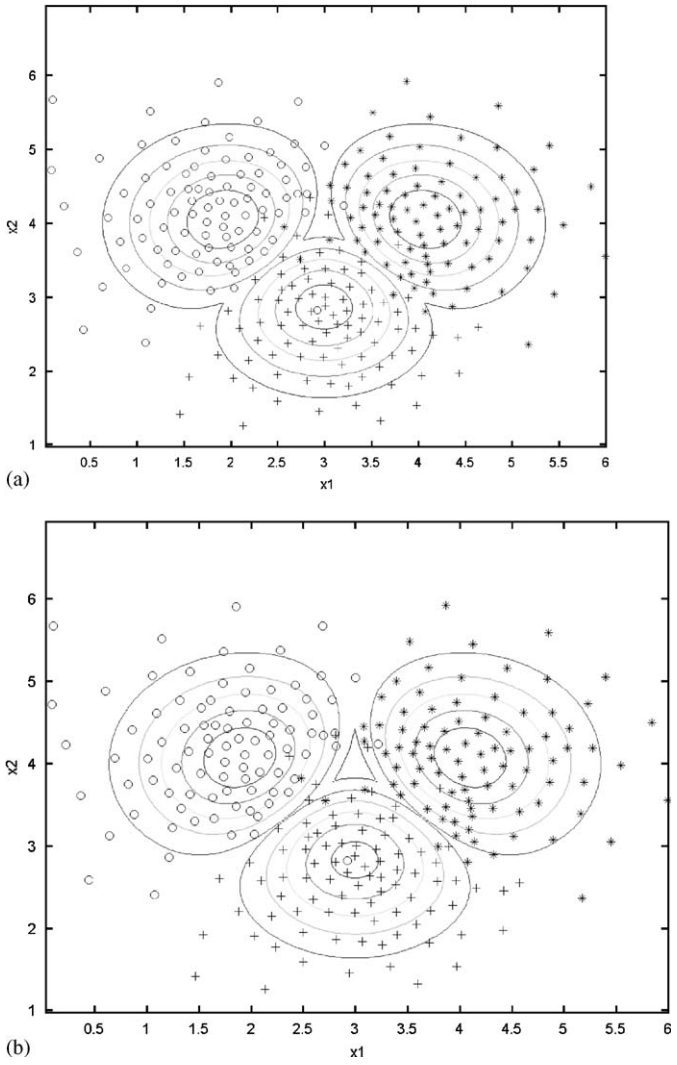

Fig. 4. Prototypes computed with RegLVQ1 and the estimated contour levels of RegLVQ1's density function for the synthetic three-class problem for (a) $\lambda = 0.5$ and $\lambda = 0.7$.

where $\lambda > 0$. Note that (21) yields the on-line K-means algorithm performed separately for each class when $\lambda = 0$, and gives LVQ1 when $\lambda = 1$. A simple geometrical interpretation of (21) in comparison with (1) (LVQ1) is such that λ scales the amount by which the prototype with a different class is moved away from the input pattern, thus distinguishing it from the other case in which the prototype of the same class is moved closer. It can be considered a simple version of differentiating updates according to the prototype's class, as (7) and (8) do.

Additionally, the batch version makes use of Newton optimization over E_{RegLVQ1} and can be defined as

$$\mathbf{P}[n+1] = \left[\mathbf{m}_0^{\mathrm{T}}[n+1] \cdots \mathbf{m}_{M-1}^{\mathrm{T}}[n+1]\right]^{\mathrm{T}} = \mathbf{P}[n] - \mathbf{H}[n]^{-1} \frac{\partial E_{\mathrm{RegLVQ1}}[n]}{\partial \mathbf{P}[n]}, \tag{22}$$

where $\mathbf{H}$ is Hessian matrix defined as (18) and (19) denote. Hence, the update equation in the batch version is

$$\mathbf{m}_j[n+1] = \frac{1}{(1+\lambda)N_{m_j}[n] - \lambda N_j[n]} \sum_{i=0}^{N-1} 1\left(\mathbf{x}_i \in R_j[n]\right)\left(1\left(\mathrm{cl}(\mathbf{x}_i) = \mathrm{cl}\left(\mathbf{m}_j\right)\right) - \lambda 1\left(\mathrm{cl}(\mathbf{x}_i) \neq \mathrm{cl}\left(\mathbf{m}_j\right)\right)\right)\mathbf{x}_i, \quad j = 0, \ldots M-1, \tag{23}$$

where N_{m_j} and N_j are defined as above. While (21) performs the *pattern-based version* of gradient descent over (16) and is globally convergent for λ belonging to certain interval, batch RegLVQ1 is locally convergent and must be avoided when the algorithm is not near a minimum point, since $\mathbf{H}$ can be negative when $(1+\lambda)N_{m_j}/\lambda < N_j$. If this were the case, the algorithm would not converge to a minimum at all, and it must be restarted. Another problem arises when $(1+\lambda)N_{m_j}[n] = \lambda N_j[n]$. This problem can be solved by retaining the last value: $\mathbf{m}_j[n+1] = \mathbf{m}_j[n]$. If this strategy is

insufficient and it sticks at these points, the algorithm may need restarting. However, the convergence of batch RegLVQ1 is very fast due to the Newton effect [27].

4. Simulations

4.1. Data sets

Our experiments used three small data sets (Glass, Sonar and Soybean), four medium-sized data sets (DNA, Satimage, Segment and Speech) and three large sets (Lower, Upper and Shuttle). All but the Speech, Lower and Upper data sets are in the UCI repository of machine learning databases [12]. The Speech data set was taken from the LVQ_PAK [31] and the Upper and Lower data sets were generated from the handwritten NIST database [22]. All these databases belong to real-world problems. Their main features are summarized in Table 1.

4.2. Results

The databases were divided into training, validation and test sets according to the distributions shown in Table 1. Samples were randomly selected from the databases, but the original distribution of samples was preserved for each class in each set. For the three small data sets, 1000 runs were generated. 100 runs were computed for the rest. In the simulations, the initial value of the prototypes for all the learning algorithms was computed through a random selection of training samples of the same class (LVQ_PAK's eveninit program [31]). The training was over when the misclassification rate in the validation set stopped decreasing.

All the parameters, including the regularizing parameter λ, were estimated using the validation set. The optimal values of λ estimated with the validation set are much smaller than those large λs that may cause non-convergence of the learning algorithm. In this way, we have practically ensured the convergence of the RegLVQ1 algorithm for the enclosed data set.

The reported experiments have only included comparisons between on-line versions of LVQ1 and RegLVQ1 algorithms for two reasons: (1) the batch versions have additional convergence problems that must be taking into account for their convenient use and (2) on-line and batch versions exhibit different generalization capabilities since their convergence points are not the same [6].

In a preliminary set of experiments, we compared the performance of on-line LVQ1 and RegLVQ1 with the 10 real-world classification data sets. Table 2 shows the test classification errors (mean/variance) for the two resulting NN classifiers, the estimated optimum number of prototypes, the estimated optimum value of λ with the validation set and the relative improvement (RI) of RegLVQ1 over LVQ1. As can be observed, RegLVQ1 outperforms LVQ1 in all the problems. Particularly, RegLVQ1 worked very well with the DNA, Shuttle and Segment data sets with a relative improvement of 50.44%, 35.22% and 14.76%, respectively. For the Soybean, Sonar & Glass data sets, the improvement was moderate (around 10%), and it was around 6% for the Satimage and Upper data sets. Finally, the improvement in the Lower and Speech problems was hardly noticeable (around 1%). These differences in performance could be due to the fact that a finite value of λ, which has been estimated with the validation set, is employed, so the upper bound of the training classification error is decreased only to some extent. Finally, it is worth noting that the accuracy of RegLVQ1 with the ten classification problems is competitive with other learning algorithms previously tested: e.g. the results on the sonar problem, which has been proved linearly separable [47], are far similar than those reported by some learning perceptron algorithms [47].

A second set of experiments, summarized in Table 3, were performed for the three data sets in which RegLVQ1 performs best (DNA, Shuttle and Segment). The aim was the comparison of RegLVQ1 with large-margin NN classifiers such as soft 2-NN [5] and learn 1-NN [7], which has been proved superior to LVQ 2.1 and LVQ3. Finally, we also include results with

Table 1
Databases

Database	Classes	Inputs	Samples	Training set size	Validation set size	Test set size
DNA	3	180	3186	1501	499	1186
Glass	4	9	214	109	32	73
Lower	26	64	35 904	24 205	1031	10 453
Satimage	5	36	6425	4293	712	1430
Segment	7	19	2300	698	205	1397
Shuttle	7	9	58 000	34 803	8697	14 500
Sonar	2	60	209	104	34	71
Soybean	19	35	684	307	121	256
Speech	20	20	3925	1962	647	1315
Upper	26	64	41 574	24 420	5162	11 992

Table 2
LVQ1 vs. RegLVQ1

Database	LVQ1		RegLVQ1			
	Test error[a]	No. of prototypes	Test error	No. of prototypes	λ	RI%
DNA	23.39 (8.21)	64	11.59 (2.89)	64	2.5	50.44
Glass	28.76 (0.1)	64	26.54 (0.8)	64	5	7.72
Lower	14.48 (0.23)	1600	14.28 (0.3)	1600	2.5	1.36
Satimage	11.08 (0.38)	256	10.40 (0.45)	256	2	6.1
Segment	17.34 (0.04)	256	14.78 (0.27)	256	8	14.76
Shuttle	1.19 (0.19)	256	0.77 (0.20)	256	4	35.22
Sonar	22.53 (0.5)	64	20.22 (1.90)	64	3	10.25
Soybean	13.26 (1.5)	256	12.09 (1.28)	256	8	8.8
Speech	10.086 (0.30)	1600	9.895 (0.37)	1600	3.5	1.89
Upper	7.65 (0.20)	1600	7.24 (0.2)	1600	5	5.3

[a]Mean (variance).

Table 3
Results of the RegLVQ1-based 1-NN, Learn 1-NN, soft 2-NN and full 1-NN algorithms in three classification problems

Database	RegLVQ1-based 1-NN		Learn 1-NN [7]		Soft 2-NN [5]		Full 1-NN	
	Test error[a]/ training time[a]	No. of prototypes	Test error/ training time	No. of prototypes	Test error/ training time	No. of prototypes	Test error/ training time	No. of prototypes
DNA	11.59 (2.89)/ 2.011 (0.53)	64	39.50 (0.56)/ 60.55 (2.94)	512	14.86 (1.45)/ 18.81 (4.87)	128	25.38/0	1501
Segment	14.78 (0.27)/ 0.17 (0.37)	256	12.95 (1.10)/ 73.21 (24.58)	128	16.37 (0.03)/ 9.99 (0.87)	512	13.24/0	698
Shuttle	0.77 (0.20)/6.6 (1.08)	256	4.33 (0.890)/ 11.86 (0.38)	512	0.57 (0.30)/ 5.52 (1.53)	128	0.15/0	34803

[a]Mean (variance).

1-NN classifiers that use all the training samples as the set of prototypes, denoted as full 1-NN, which typically exhibit a high accuracy since they include the entire training database. According to these results and the first experiments, Table 3 shows that the RegLVQ1-based 1-NN outperforms the other three classifiers in the DNA data set while it is ranked 3 out of 4 in the Segment and Shuttle data sets. Also, it is shown that RegLVQ1 employs a much shorter time for training than large-margin NN classifiers, due to the simplicity of its learning algorithm.

These results demonstrate the utility of RegLVQ1 1-NN classifiers compared to NN classifiers such as soft K-NN and learn 1-NN, which have a greater computation burden not only in the learning phase but also when the classifier is employed after the prototypes has been computed. As denoted in (12) and (13), their discriminant functions involve the computation of kernel functions that employ the nearest-neighbour prototypes of each class and the two nearest-neighbours for the learn 1-NN and soft K-NN classifiers, respectively, each time an input pattern is presented. Also, the number of prototypes employed in RegLVQ1 is much lower than in a lazy method such as the full 1-NN, in which the whole training set is employed as a prototype. Therefore, it reduces the execution time of the NN classifier, since the search for the nearest prototype to an input pattern monotonically increases as the number of prototypes increases.

5. Conclusions

A straightforward generalization of LVQ1 (RegLVQ1) has been presented which includes K-means and LVQ1 as special cases. The minimization of the RegLVQ1 cost function ensures a training classification error $\mathrm{err}_T(\lambda) \leqslant 1/(1+\lambda)$ where λ is a regularizing parameter determined by the user. This would also reduce the number of feasible solutions and can be considered a simple mechanism for reducing undesired local minimum points. However, in practice, a finite value of λ is employed and the amount of improvement depends on the particular problem. Experimental results show that this simple extension increases the classification accuracy of the nearest-neighbour classifier compared to LVQ1, and it performs as well as other learning algorithms and lazy methods that are computationally more complex.

Acknowledgements

The author wishes to thank the anonymous reviewers for their comments, which have helped to improve the final version of this paper. This work was supported in part by the *Ministerio de Educación y Ciencia* and by the EU's European Regional Development Fund through Grant TEC2004-05127-C02-01.

References

[1] D.W. Aha, Special issue on lazy learning, Artif. Intell. Rev. (1997) 1–5.
[2] E. Alpaydin, Voting over multiple condensed nearest neighbor classifiers, Artif. Intell. Rev. 1–5 (1997) 115–132.
[3] A. Benveniste, M. Métivier, P. Priouret, Adaptive Algorithms and Stochastic Approximations, Springer, Berlin, 1990.
[4] S. Bermejo, J. Cabestany, A batch learning vector quantization for nearest neighbour classifiers, Neural Process. Lett. 11 (2000) 173–184.
[5] S. Bermejo, J. Cabestany, Adaptive soft K-nearest neighbor classifiers, Pattern Recognition 33 (2000) 1999–2005.
[6] S. Bermejo, J. Cabestany, Finite-sample convergence of the LVQ1 algorithm and the BLVQ1 algorithm, Neural Process. Lett. 13 (2001) 135–157.
[7] S. Bermejo, J. Cabestany, Learning with 1-nearest-neighbour classifiers, Neural Process. Lett. 13 (2001) 159–181.
[8] S. Bermejo, J. Cabestany, Oriented principal component analysis for large margin classifiers, Neural Networks 14 (2001) 1447–1461.
[9] S. Bermejo, J. Cabestany, Local averaging of ensembles of LVQ-based nearest neighbour classifiers, Appl. Intell. 20 (2004) 45–55.
[10] J.C. Bezdek, T.R. Reichherzer, G-S. Lik, Y. Attikiouzel, Multiple-prototype classifier design, IEEE Trans. Systems, Man, Cybern-Part C: Appl. Rev. 28 (1998) 67–79.
[11] C. Bishop, Neural Networks and Pattern Recognition, Oxford University Press, Oxford, 1995.
[12] C.L. Blake, C.J. Merz, UCI Repository of machine learning databases, University of California, Department of Information and Computer Science, Irvine, CA, 1998. [Online] Available: http://www.ics.uci.edu/~mlearn/MLRepository.html.
[13] T. Bojer, B. Hammer, D. Schunk, K.T. von Toschanowitz, Relevance determination in learning vector quantization, in: M. Verleysen (Ed.), Proceedings of the European Symposium on Artificial Neural Networks (ESANN 2001), D-facto Publications, Bruges, Belgium, 2001, pp. 271–276.
[14] L. Bottou, Online learning and stochastic approximation, in: D. Saal (Ed.), Online Learning and Neural Networks, Cambridge University Press, Cambridge, UK, 1998.
[15] L. Bottou, V. Vapnik, Local learning algorithms, Neural Comput. 4 (1992) 888–890.
[16] B.V. Darasay (Ed.), Nearest Neighbor Pattern Classification Techniques, IEEE Computer Society Press, Los Alamitos, 1991.
[17] L. Devroye, L. Györfi, G. Lugosi, A Probabilistic Theory of Pattern Recognition, Springer, Berlin, 1996.
[18] A. Djouadi, On the reduction of the nearest-neighbor variation for more accurate classification and error estimates, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1998) 567–571.
[19] A. Djouadi, E. Bouktache, A fast algorithm for the nearest-neighbor classifier, IEEE Trans. Pattern Anal. Mach. Intell. 19 (1997) 277–282.
[20] J. Fan, I. Gijbels, Local Polynomial Modelling and its Applications, Chapman & Hall, London, 1996.
[21] J. Friedman, Flexible metric nearest neighbor classification, Technical Report, Department of Statistics and Stanford Linear Accelerator Center, Stanford University, Stanford CA, 1994.
[22] M. Garris, et al., NIST form-based handprint recognition system (Release 2.0), National Institute of Standards and Technology, 1997.
[23] A. Gersho, R.M. Gray, Vector Quantization and Signal Compression, Kluwer Academic Publishers, Boston, 1992.
[25] Y. Hamamoto, S. Uchimura, S. Tomita, A bootstrap technique for nearest neighbor classifier design, IEEE Trans. Pattern Anal. Mach. Intell. 19 (1997) 73–79.
[26] B. Hammer, T. Villmann, Generalized relevance learning vector quantization, Neural Networks 15 (2002) 1059–1068.
[27] M. Hestenes, Conjugate Direction Methods in Optimization, Springer, Berlin, 1980.
[28] J.B. Hiriart-Urruty, C. Lemaréchal, Convex Analysis and Minimization Algorithms, Springer, Berlin, 1993.
[29] L. Holmström, P. Koistinen, J. Laaksonen, E. Oja, Neural and statistical classifiers-taxonomy and two case studies, IEEE Trans. Neural Networks 8 (1997) 5–17.
[30] T. Kohonen, Self-organizing Maps, second ed., Springer, Berlin, 1996.
[31] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, K. Torkkola, LVQ_PAK, the learning vector quantization program package, Version 3.1, Helsinki University of Technology, Laboratory of Computer and Information Science, Helsinki, 1995.
[32] B. Kosko, Neural Networks and Fuzzy Systems, Prentice-Hall, Englewood Cliffs, 1992.
[33] L.I. Kuncheva, J.C. Bezdek, Presupervised and postsupervised prototype classifier design, IEEE Trans. Neural Networks 10 (1999) 1142–1152.
[34] A. Laha, R. Pal, Some novel classifiers designed using prototypes extracted by a new scheme based on self-organizing feature map, IEEE Trans. Systems, Man, Cybern.-Part B: Cybern. 31 (2001) 881–890.
[35] A. LaVigna, Nonparametric classification using learning vector quantization, Ph. D. Dissertation, University of Maryland, 1990.
[36] E.-W. Lee, S.-I. Chae, Fast design of reduced-complexity nearest-neighbor classifiers using triangular inequality, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1998) 562–566.
[37] S-W. Lee, H-H. Song, Optimal design of reference models for large-set handwritten character recognition, Pattern Recognition 27 (1994) 1267–12774.
[38] J. McQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics and Probability, Berkley, 1967, pp. 281–296.
[39] D. Michie, D.J. Spiegelhalter, C.C. Taylor (Eds.), Machine Learning, Neural and Statistical Classification, Ellis Horwood, London, 1994.
[40] M. Pregenzer, G. Pfurtscheller, D. Flotzinger, Automated feature selection with a distinction sensitive learning vector quantizer, Neurocomputing 11 (1996) 19–29.
[41] F. Ricci, P. Avesani, Data compression and local metrics for nearest neighbor classification, IEEE Trans. Pattern Anal. Mach. Intell. 21 (1999) 380–384.
[42] A. Sato, K. Yamada, Generalized learning vector quantization, in: G. Tesauro, D. Touretzk, T. Lee (Eds.), Advances in Neural Information Processing Systems (NIPS), vol. 7, MIT Press, Boston, MA, 1995, pp. 423–429.
[43] S. Seo, M. Bode, K. Obermayer, Soft nearest prototype classification, IEEE Trans. Neural Networks 14 (2003) 390–398.
[44] A. Sharkey (Ed.), Combining Artificial Neural Nets, Springer, Berlin, 1999.

[45] A. Smola, P. Barlett, B. Schölkopf, B. Shuurmans, Introduction to large margin classifiers, in: A. Smola, P. Barlett, B. Schölkopf, B. Shuurmans (Eds.), Advances in Large Margin Classifiers, MIT Press, Boston, MA, 1999.
[46] C. Stanfill, D. Waltz, Toward memory-based reasoning, Comm. ACM 29 (1986) 1213–1228.
[47] J.M. Torres, M.B. Gordon, Characterization of the sonar signals benchmark, Neural Process. Lett. 7 (1998) 1–4.
[48] K. Urahama, Y. Furukawa, Gradient descent learning of nearest neighbor classifiers with outlier rejection, Pattern Recognition 28 (1995) 761–768.

Sergio Bermejo received his M.Sc. and Ph.D. degrees in Telecommunication Engineering, in 1996 and 2000, respectively, from the Universitat Politècnica de Catalunya (UPC). In 1996, he joined UPC's Department of Electronics Engineering (DEE) as a researcher. Currently, he holds a position as associate professor in the DEE and teaches at the School of Telecommunications Engineering of Barcelona (ETSETB). His research interests are statistical learning, with a special focus on large margin classification, unsupervised learning and their application to smart sensors, signal processing, software agents and autonomous robotics