Available online at www.sciencedirect.com
SCIENCE DIRECT®
Neurocomputing 69 (2006) 922–927
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Letters

An LVQ-based adaptive algorithm for learning from very small codebooks

J.S. Sánchez[a,*], A.I. Marqués[b]

[a]Dept. de Llenguatges i Sistemes Informàtics, Universitat Jaume I, Av. Sos Baynat s/n, E-12071 Castelló de la Plana, Spain
[b]Dept. d'Administració d'Empreses i Màrqueting, Universitat Jaume I, Av. Sos Baynat s/n, E-12071 Castelló de la Plana, Spain

Received 4 September 2005; received in revised form 18 October 2005; accepted 20 October 2005
Available online 1 December 2005
Communicated by R.W. Neucomb

Abstract

The present paper introduces an adaptive algorithm for competitive training of a nearest neighbor (NN) classifier when using a very small codebook. The new learning rule is based on the well-known LVQ method, and uses an alternative neighborhood concept to estimate optimal locations of the codebook vectors. Experiments over synthetic and real databases suggest the advantages of the learning technique here introduced.

Keywords: Learning; LVQ; Nearest centroid neighbor; Nearest neighbor

1. Introduction

The NN rule [3] is one of the most widely used learning algorithms. Each class is given as a set of n previously labeled prototypes, and then an input sample is classified according to the class of its nearest neighbor among all prototypes in the training set. The NN rule and its extension to k neighbors (or k-NN rule, in which the k closest neighbors "vote" for the label of the unknown input sample) combine their conceptual and implementational simplicity with the fact that their asymptotic (that is, $n \to \infty$) error rate is conveniently bounded in terms of the optimal Bayes error when $k \to \infty$.

In practice, it is not always possible to reach the expected asymptotic performance due to the relatively small number of prototypes available. On the other hand, if we have a sufficiently large training set, then the major problem refers to the large computational burden because of the considerable number of distance computations required by the k-NN rule. Among the different proposals to overcome the computational difficulties, a first possibility consists of selecting a sufficiently small subset of prototypes that leads to approximately the same performance as the NN rule applied to the whole training set [4]. Other techniques try to generate a reduced set of codebook vectors which are moved into optimal positions to guarantee low error rates. Belonging to this group, the most representative approaches correspond to the supervised learning vector quantization (LVQ) algorithms [6].

2. The LVQ algorithms

The ultimate aim of LVQ is to define optimal class regions in the input data space. In general, from a given training set X of n prototypes, n_M codebook vectors m_i are initially picked up. Then these codebook vectors are adaptively modified in such a way that the NN rule minimizes the average expected misclassification probability. The learning process consists of incrementally moving some of the codebook vectors m_i in the neighborhood of a prototype x towards it and some away from it according to the result of the plain NN rule [6]. Taking into account that differences among the various LVQ algorithms refer to the

*Corresponding author. Tel.: +34 964 728350; fax: +34 964 728435.
E-mail address: sanchez@uji.es (J.S. Sánchez).

0925-2312/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2005.10.001

modifications to the codebook vectors, a general LVQ procedure can be written as follows:

Algorithm 1 *General_LVQ*(X, n_M, r)

$M(0) \longleftarrow Init(X, n_M)$
$t \longleftarrow 0$
repeat
  $x \longleftarrow Extract(X)$
  $M(t+1) \longleftarrow Modify(x, M(t))$
  $t \longleftarrow t+1$
until $(t = L)$ /* *L is the total number of learning steps* */

From the above algorithm, it has to be remarked that $Init(X, n_M)$ chooses n_M prototypes from the original training set X, producing the initial codebook $M(0)$, that is, the codebook at step $t = 0$. Then, during the learning stage, $Extract(X)$ picks up a prototype x from the training set X, and $Modify(x, M(t))$ updates the current codebook $M(t)$ by using the prototype x selected at step t.

Let $x \in X$ be an input sample, let m_c be the nearest codebook vector m_i to x, and let $m_c(t)$ represent the codebook vector m_c at step t. The learning process in the basic LVQ1 algorithm consists of updating the position of m_c. If the class label of the codebook vector m_c matches the class label of the training prototype x, then the codebook vector is moved towards x. Otherwise, it is moved away from the given input prototype. The modifications to the codebook vector m_c are performed according to the following general rule:

$$m_c(t+1) = \begin{cases} m_c(t) + \alpha(t)[x - m_c(t)] & \text{if } class(x) = class(m_c), \\ m_c(t) - \alpha(t)[x - m_c(t)] & \text{if } class(x) \neq class(m_c), \end{cases} \quad (1)$$

where $0 < \alpha(t) < 1$ denotes the corresponding learning rate, and it may be constant or decrease monotonically with time.

In the case of LVQ2, two codebook vectors, m_i and m_j, which are the nearest neighbors to an input sample x are simultaneously updated. While m_i must belong to the correct class, m_j to some wrong class. Moreover, x must fall into a "window" defined around the midplane of m_i and m_j. The adjustments in the LVQ2 algorithm can be expressed as follows:

$$m_i(t+1) = m_i(t) + \alpha(t)[x - m_i(t)],$$

$$m_j(t+1) = m_j(t) - \alpha(t)[x - m_j(t)]. \quad (2)$$

The LVQ2 algorithm can be improved in order to include corrections which ensure that the m_i continue approximating the class distributions although it becomes a longer learning process. Thus, the general modifications in the LVQ3 scheme are performed according to the LVQ2 conditions. Moreover, if x, m_i and m_j belong to the same class, then the learning rule is defined as

$$m_k(t+1) = m_k(t) + \varepsilon\alpha(t)[x - m_k(t)] \quad (3)$$

for $k \in \{i, j\}$. In [7], the authors recommend applicable values of ε between 0.1 and 0.5, and state that the optimal value of ε depends on the size of the "window".

Finally, the LVQ1 algorithm can be extended by assigning a different learning rate $\alpha_i(t)$ to each m_i. Thus the learning process for OLVQ1 (i.e., the optimized version of LVQ1) can be defined as follows:

$$m_c(t+1) = \begin{cases} m_c(t) + \alpha_c(t)[x - m_c(t)] & \text{if } class(x) = class(m_c), \\ m_c(t) - \alpha_c(t)[x - m_c(t)] & \text{if } class(x) \neq class(m_c). \end{cases} \quad (4)$$

The optimal values of $\alpha_i(t)$ are determined by the recursion [7]

$$\alpha_i(t) = \frac{\alpha_i(t-1)}{1 + s(t)\alpha_i(t-1)}, \quad (5)$$

where $s(t) = +1$ if x and m_c belong to the same class, and $s(t) = -1$ if x and m_c belong to different classes.

3. The learning *k*-NN rules

Three adaptive learning rules (namely, *Learning k-NN rules*) are suggested in [8] as a further refinement of the basic LVQ1 algorithm. Their general idea consists of using a number k of nearest codebook vectors to a given training sample for "presumably" obtaining an increase in performance. By the first scheme, all the k nearest codebook vectors to an input sample are moved. The second approach consists of moving only the kth and the $(k+1)$th nearest codebook vectors if the interchange of their order would vary the classification of the training sample from incorrect to correct. Finally, in the third learning rule, all the $(k+1)$ nearest codebook vectors are moved. The modifications to the codebook vectors are made according to the learning rule of Eq. (1).

From the empirical analysis carried out in [8], it seems that the standard LVQ1 performs better than any of the proposed Learning k-NN rules if the codebook size is rather small. However, those extensions to k neighbors achieve lower error rates than the basic LVQ1 when the codebook size is substantially increased, especially in the case of the first rule proposed. These results agree with the asymptotical theoretical analysis for the k-NN rule [4], in the sense that this can reach the optimal classification accuracy only when a sufficiently large number of codebook vectors is available.

4. An alternative to standard LVQ methods

The Learning k-NN rules involve the idea that considering a larger number of codebook vectors close to an input sample may lead to lower error rates than using the nearest prototype only, in a similar way as the standard k-NN rule usually outperforms the plain NN classifier, that is, the 1-NN rule. Despite these initial expectations, the experimental analysis reported in [8] has demonstrated that

improvement in performance can just be achieved if the codebook size is large enough.

Applying a strategy similar to that used in the Learning k-NN rules, a new extension to the LVQ methods is introduced in this section. The novel learning approach is based on the application of an alternative neighborhood definition, called *Nearest Centroid Neighborhood* [2], which has already been proven to behave significantly better than the conventional nearest neighborhood in most small sample size classification problems [9].

4.1. The nearest centroid neighborhood

This concept focuses on the idea that neighborhood of a point is subject to two complementary constraints. Thus, by the *distance criterion*, the k neighbors of a sample p are as near to it as possible. Second, by the *symmetry criterion*, their centroid is also as close to p as possible. The centroid (also called *center of mass* or *center of gravity*) $\mathscr{C}$ of a set of points $\mathscr{X} = \{x_1, x_2, \ldots, x_N\}$ can be computed as

$$\mathscr{C} = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{6}$$

The k nearest centroid neighbors (NCN) of a sample p can be searched for through an iterative procedure [2] as follows:

(1) The first NCN to p corresponds to its NN, say q_1.
(2) The ith neighbor, q_i ($2 \leqslant i \leqslant k$) is such that the centroid of this and all previously selected neighbors, say $q_1, \ldots, q_i$ is the closest to p.

This algorithm produces a kind of neighborhood in which spatial distribution of neighbors is taken into account because of the centroid criterion. Besides, proximity of neighbors to the sample is guaranteed because of the incremental nature of the way in which they are obtained from the first NN. From this neighborhood, it has been defined the k-NCN classification rule [9], in which the k-NCN decide, by majority voting, the class label of the input sample.

4.2. Learning with k-NCN rules

Like LVQ algorithms, the approaches proposed here also use a fixed number of codebook vectors, whose initial positions in the input space will be further modified in order to minimize the misclassification rate. When a training prototype is selected, some of its k nearest codebook vectors are moved towards it and some away from it according to their class labels. The alternative suggested in this paper (hereafter, called *Learning k-NCN*) makes use of the nearest centroid neighborhood previously described, while the Learning k-NN schemes are based on the traditional concept of nearest neighborhood. The basic aim of all the NCN-based rules is to inspect a sufficiently small region *around* the codebook vector.

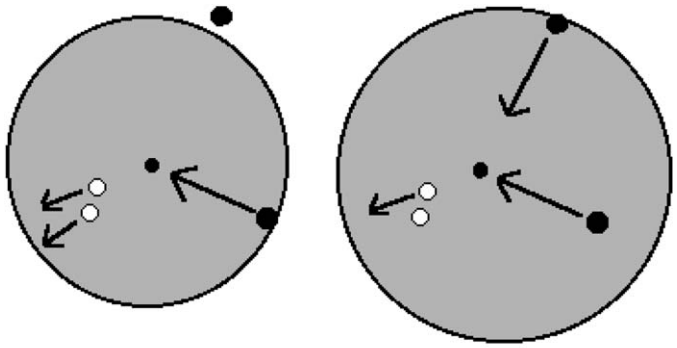

Fig. 1. Comparison between the learning k-NN and k-NCN rules.

In a few words, the adaptive learning algorithm presented here consists of moving all neighbors which are *surrounding* an input sample. If the class of a codebook vector matches with the class of the training prototype, the codebook vector is moved towards it. Otherwise, it is moved in the opposite direction. The modifications to the codebook vectors are performed according to the general rule of Eq. (1). For $k = 1$, this rule is equivalent to LVQ1. Nevertheless, when $k > 1$, the NCN tend to *surround* the input prototype.

Fig. 1 illustrates the application of the basic Learning k-NN and Learning k-NCN rules in a 2-class (white and black) problem. For this example, k is set to 3. The region containing both the training sample and its 3 NN corresponds to the shaded area in Fig. 1(a). Analogously, Fig. 1(b) represents the region in which the training sample and its 3 NCN lie. One can observe that the region given by the NCN results greater than that produced by the NN. The input sample has been depicted by a smaller black ball. Note that this would be misclassified by means of the 3-NN rule, but not by using the 3-NCN classifier.

5. Comparative analysis

In this section, the k-NN rule, the LVQ method and the first Learning k-NN approach are empirically compared to the Learning k-NCN scheme proposed here. Typical settings for the parameter k (3, 5, 7, 9, and 11) have been tried and those leading to the best performance have finally been included in the figures.

The *holdout* method averaged over five different random partitions of the original database has been used to obtain error rate estimates. This method consists of splitting the original set into two mutually exclusive subsets called a training set and a test set [1,5]. In our experiments, we have used half for initialization and training the classifier and half for testing. The relative number of prototypes per class within each partition keeps the original class distribution. All algorithms have been repeated ten times for each random partition.

Particularly interesting for this kind of classifiers is the codebook size, that is, the number of prototypes initially placed into the input data space. Hence, the present comparative analysis primarily focuses on how the codebook size affects to the error rate resulting from the different learning algorithms. The number of learning steps will be 50 times the total number of codebook vectors, while the learning rate will be constant ($\alpha(t) = \pm 0.2$) for all the experiments. The LVQ approach is here applied according to the usual method [7], that is, learning starts with the OLVQ1, which has very fast convergence, and then the basic algorithms are used.

5.1. The artificial databases

Experimentation has been carried out over two artificial databases and a real data set. The first synthetic problem is to distinguish between an unimodal normal distribution and a multimodal Gaussian distribution (a sum of three different normal distributions), with high overlapping between both classes. The original data set contains 5000 patterns (50% from each class).

As can be seen in Fig. 2, the proposed Learning k-NCN algorithm ($k = 3$) clearly behaves better than any other scheme. When 150 codebook vectors are placed into the input data space, the LVQ method and the novel learning approach indeed achieve very similar classification accuracies. However, from 300 prototypes, differences between the performance of the Learning k-NCN rule and those of the conventional algorithms are statistically significant.

The aim of the experiments carried out over the second artificial database is to analyze the linear separability of the classifiers when some class is nested in another without overlapping. The original data set contains 2500 prototypes corresponding to two distinct classes, which are two-dimensional uniform concentric circular distributions. Class 0 contains 921 prototypes uniformly distributed into a circle of radius 0.3 and centered at point (0.5, 0.5). There are 1579 vectors belonging to class 1, corresponding to points uniformly distributed into a ring centered at (0.5, 0.5) with internal and external radii equal to 0.3 and 0.5, respectively. Fig. 3 provides the graphical representation for this database.

Fig. 4(a) shows the classification accuracy achieved by each algorithm as a function of the codebook size over the second synthetic database. Despite the general good behavior, note that the novel Learning k-NCN rule also outperforms all the other approaches. In fact, the highest classification accuracy is achieved with the algorithm proposed here when applied to the codebook with 600 prototypes (98.63%). With regard to differences between the conventional LVQ method and the Learning k-NN rule, it seems that the former exhibits a better performance than the latter when using a small number of prototypes. Nevertheless, from using a moderate codebook size (300 codebook vectors), the Learning k-NN approach outperforms the standard LVQ.

Fig. 4(b) illustrates performances obtained over the set of 300 codebook vectors when varying the neighborhood size, k. As can be seen, the approach proposed here systematically achieves better results than the other two algorithms. In this case, the highest classification accuracy corresponds to the Learning 5-NCN rule (98.14%), while the best performance of the standard k-NN classifier reaches a recognition rate of 96.56%.

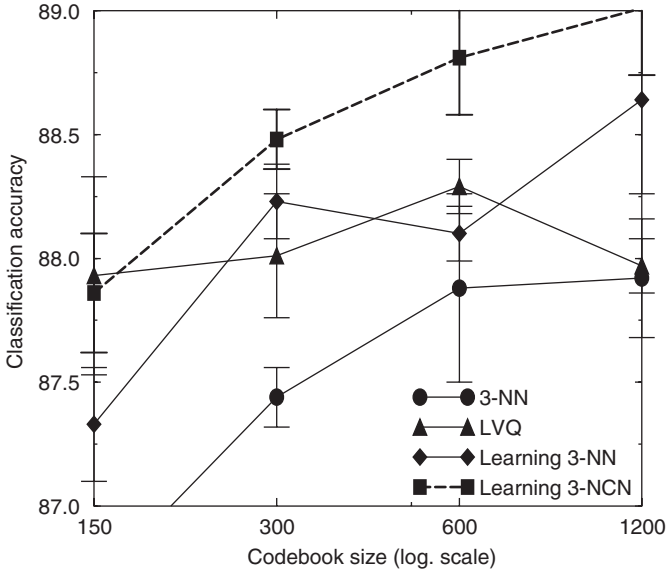

Fig. 2. Classification accuracy with varying codebook sizes.

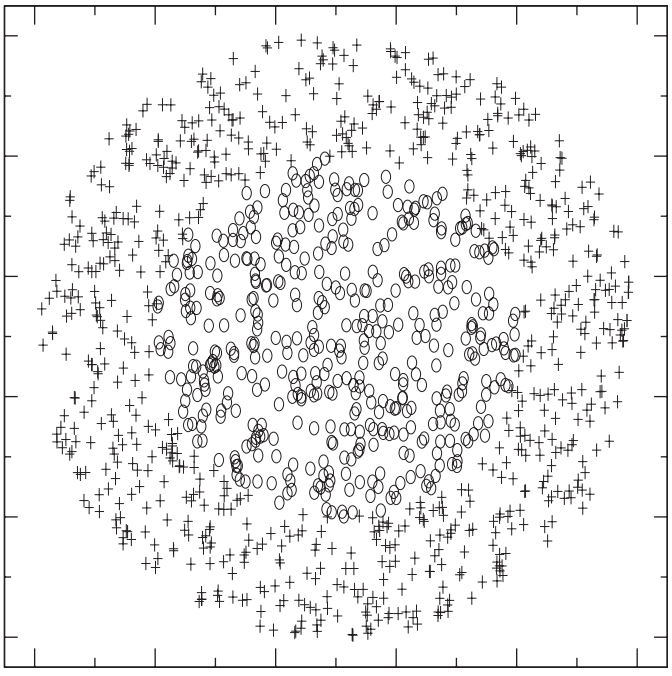

Fig. 3. The second artificial database. Points from class 0 are circles, while class 1 is represented by the symbol "+".

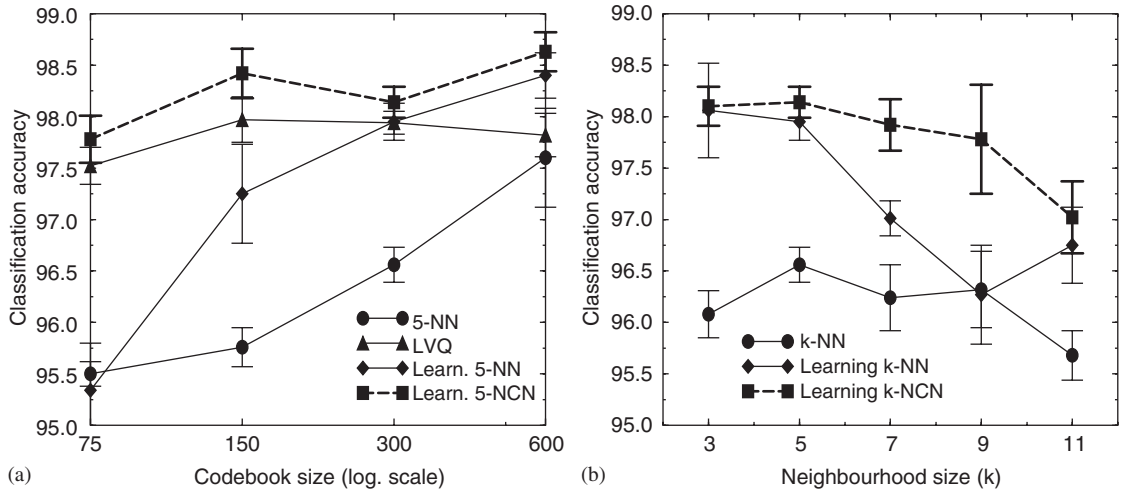

Fig. 4. Comparison of different classifiers with respect to (a) the codebook size, and (b) the neighborhood size, for the second artificial database.

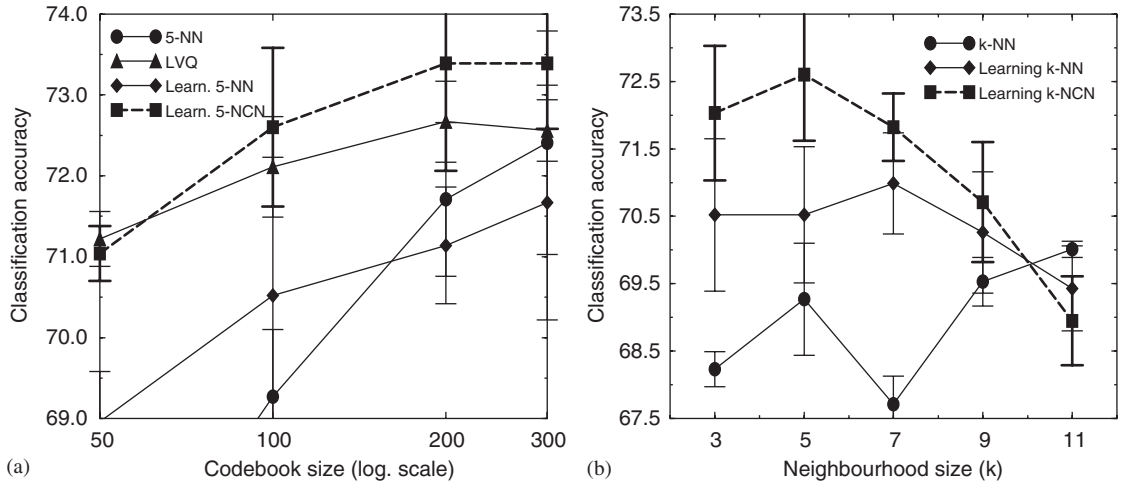

Fig. 5. Performance of the classifiers as a function of (a) the codebook size, and (b) the neighborhood size, for the real database.

5.2. A real database

The real data set, taken from the UCI Machine Learning Database Repository (http://www.ics.uci.edu/~mlearn), tries to determine the presence or absence of diabetes among Pima Indian females at least 21 years old, according to eight different numeric-valued attributes. The original database contains 768 prototypes: 500 from class 0 and 268 belonging to class 1 (tested positive for diabetes).

The results plotted on Fig. 5(a) are quite similar to those achieved over both synthetic databases. Firstly, the best option for most codebooks clearly corresponds to the Learning k-NCN algorithm. Furthermore, it is worth noting that the classification accuracy obtained by the Learning k-NCN approach over the codebook with 100 prototypes is even considerably higher than that of any other algorithm applied to the set of 300 codebook vectors. In fact, for this real problem, differences between the performance of the learning algorithm proposed here and those of all the other rules are significantly important.

Fig. 5(b) compares the basic k-NN rule and the Learning k-NN approach to the Learning k-NCN scheme using different values of the parameter k over the set of 100 codebook vectors. The most important fact is that the algorithm proposed here presents a considerably better behavior than the other two alternatives. In particular, the most statistically significant differences among the three approaches are reached with $k = 5$: while the Learning k-NCN rule achieves an accuracy of 72.60%, the results corresponding to the standard k-NN and the Learning k-NN rules are 69.27% and 70.52%, respectively.

It is to be noted the behavior of both the Learning k-NN and Learning k-NCN rules for $k > 9$, where classification accuracies drop down because of the neighborhood size (k). In general, all learning rules based on some kind of neighborhood tend to increase their accuracy with k until a

maximum is reached. After that, the corresponding classification accuracy presents a (progressive) degradation [3].

6. Concluding remarks

An adaptive learning algorithm, namely Learning k-NCN, has been introduced. This method makes use of the nearest centroid neighborhood concept, in which the k neighbors to an input sample are computed not only in terms of proximity but also taking care of their geometrical distribution around the given sample. The main purpose is to obtain a particular kind of information which allows to estimate more accurate locations of the codebook vectors than those of the standard LVQ, especially for small codebook sizes.

Experiments show a better behavior of the proposed Learning k-NCN rules than other standard approaches. Apart from associating a different learning rate $\alpha_i(t)$ to each m_i, like in the OLVQ1 algorithm, other extensions are possible. In particular, like the k-NN [3,4] and Learning k-NN rules [8], the schemes proposed here allow a reject option in order to discard those input samples whose k neighbors do not clearly belong to a certain class.

Acknowledgments

This work has partially been supported by research Grant TIC2003-08496 from the Spanish CICYT (Dept. of Science and Technology). We would like to thank the anonymous Reviewers and the Editor for their valuable and constructive remarks.

References

[1] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, UK, 1995.
[2] B.B. Chaudhuri, A new definition of neighbourhood of a point in multi-dimensional space, Pattern Recognition Lett. 17 (1) (1996) 11–17.
[3] B.V. Dasarathy, Nearest Neighbor Norms: NN Pattern Classification Techniques, IEEE Computer Society Press, Los Alamos, CA, 1991.
[4] P.A. Devijver, J. Kittler, Pattern Recognition: A Statistical Approach, Prentice-Hall, Englewood Cliffs, NJ, 1982.
[5] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence, vol. 2, Montreal, Canada, August 20–25, 1995, pp. 1137–1145.
[6] T. Kohonen, Self-Organizing Maps, Springer, Berlin, Heidelberg, Germany, 1995.
[7] T. Kohonen, J. Kangas, J. Laaksonen, K. Torkkola, LVQ_PAK: The learning vector quantization program package, Technical Report A30, Laboratory of Computer and Information Science, Helsinki University of Technology, FIN-02150 Espoo, Finland, 1996.
[8] J. Laaksonen, E. Oja, Classification with learning k-nearest neighbors, in: Proceedings of the International Conference on Neural Networks, vol. 3, Washington, DC, USA, June 3–6, 1996, pp. 1480–1483.
[9] J.S. Sánchez, F. Pla, F.J. Ferri, On the use of neighbourhood-based non-parametric classifiers, Pattern Recognition Lett. 18 (11–13) (1997) 1179–1186.

J.S. Sánchez is an Associate Professor in the Department of Programming Languages and Information Systems at Universitat Jaume I (Castelló de la Plana, Spain) since 1992, and is currently the head of the Pattern Recognition Section in the Computer Vision Lab. He received a B.Sc. in Computer Science from the Universidad Politécnica de Valencia in 1990 and a Ph.D. in Computer Science Engineering from Universitat Jaume I in 1998. He is author or co-author of more than 70 scientific publications, co-editor of two books and guest editor of three special issues in international journals. His current research interests lie in the areas of pattern recognition and machine learning, including nonparametric classification, feature and prototype selection, ensembles of classifiers, and decision tree induction.

A.I. Marqués is an Assistant Professor in the Department of Business Management and Marketing at Universitat Jaume I (Castelló de la Plana, Spain) since 1999. She received a Diploma in Business Studies and a B.Sc. in Business Administration and Management, both from Universitat Jaume I, in 1997 and 2002, respectively. She is currently working towards a Ph.D. degree on Logistics. Her main research interests include the application of statistical pattern recognition to logistics.