PERGAMON

Pattern Recognition 33 (2000) 1999–2005

PATTERN RECOGNITION

THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Adaptive soft k-nearest-neighbour classifiers

Sergio Bermejo*, Joan Cabestany

Department of Electronic Engineering, Universitat Politècnica de Catalunya (UPC), Gran Capità s/n, C4 building, 08034 Barcelona, Spain

Received 24 March 1999; accepted 28 June 1999

Abstract

A novel classifier is introduced to overcome the limitations of the k-NN classification systems. It estimates the posterior class probabilities using a local Parzen window estimation with the k-nearest-neighbour prototypes (in the Euclidean sense) to the pattern to classify. A learning algorithm is also presented to reduce the number of data points to store. Experimental results in two hand-written classification problems demonstrate the potential of the proposed classification system.

Keywords: Soft nearest-neighbour classifiers; Online gradient descent; Hand-written character recognition

1. Introduction

Nearest-neighbour (NN) techniques are simple but powerful non-parametric classification systems. Since their introduction in the 1950s, many sophistications of the basic schema appeared, concerned with different topics as condensing, editing, learning or extensions to include rejection. (See Darasay [1] to review much of the existing literature.) In recent years, interest in these methods has flourished again in several fields (including statistics, machine learning and pattern recognition) since they remain the best choice in many classification problems.

This paper presents a novel classification learning architecture in the context of condensed k-NN classifiers with the goal of enhancing its generalisation properties. In learning systems, generalisation performance is affected by a trade-off between the number of training examples and the capacity (e.g. the number of parameters) of the learning machine. This global trade-off can be reinterpreted in local learning systems (like NN methods) as a trade-off between capacity and locality (how local the solution is) [2]. One of the implications of this is that the learning system must control the effective number of training samples available for training locally the system. K-NN techniques are good examples of such local learning systems. For every testing pattern, they estimate posterior class probabilities with a fixed number of training points. In this way, good generalisation is guaranteed since there is always enough data to compute the estimations. By contrast, these systems have poor approximation capabilities due to estimate using a ratio of integers. The quality of the k-NN approximations can be simply improved if we use instead of the usual ratio, a Parzen window estimate computed with the k-nearest-neighbour training samples. Hence, the resulting *soft* estimation could benefit from the virtues of k-NN and Parzen estimates, since it improves the approximation quality of crisp k-NN estimators by means of a smooth interpolation and it retains their control of the training points that contribute to the estimations. However, an important disadvantage of the method, inherited from their ancestors, would be that all of the training data must be retained to compute the estimations. So we also present a more sophisticated version of the algorithm to allow fewer data points to be used. This includes a learning algorithm to compute a condensed set of prototypes from the training data.

The organisation of this paper is as follows. Section 2 derives the soft k-NN classifier in the context of local learning algorithms [2]. In Section 3 we introduce an

* Corresponding author. Tel.: +34-934017488; fax: +34-934016756.

E-mail address: sbermejo@eel.upc.es (S. Bermejo).

0031-3203/00/$20.00
PII: S0031-3203(99)00186-7

adaptive version of the previous classification rule. The proposed learning algorithm designs a condensed set of prototypes that minimises the empirical average number of misclassifications. Section 4 shows experimental results that compares successfully the adaptive soft k-NN classifier with other algorithms using two NIST handwritten character databases [3]. Finally, in Section 5 conclusions are given.

2. The soft k-NN classifier

Let $c|X \to \{1, \ldots, C\}$ be a maximum classifier that assigns an input vector $\mathbf{x}$ to one of the C existing classes. This kind of classifiers is defined in terms of a set of discriminant functions $d_l(\mathbf{x}), l = 1, \ldots, C$. The classifier c then assigns $\mathbf{x}$ to class j if $d_j(\mathbf{x}) > d_l(\mathbf{x})$ for all $l \neq j$. In the case of equally misclassification costs, c achieves a minimum expected number of misclassifications, when d_l is the posterior probability of belonging to class l, $P(C_l \backslash \mathbf{x})$. Suppose we have N observations pairs $T = \{(\mathbf{x}_i, \mathbf{y}_i)\} i = 0, \ldots, N-1$ where $\mathbf{x}_i$ is a random sample taken from input space X that belongs to one of the classes and $\mathbf{y}_i$ is an indicator variable. If $\mathbf{x}_i$ belongs to the nth class, the nth coefficient of $\mathbf{y}_i$ is equal to 1 and the other coefficients are equal to 0. To ensure c is a good classification procedure each discriminant functions $d_l(\mathbf{x})$ must estimate the binary random variable y_l (that is equal to 1 if $\mathbf{x}$ belongs to the lth class and 0 otherwise). In local algorithms this estimation is defined for each neighbourhood around the testing pattern $\mathbf{x}$ as those, which minimises a weighted empirical average loss function of the following form [2, p. 892]:

$$I_{\text{local emp}}[d_l] = \frac{1}{M} \sum_{i=1}^{M} G(\mathbf{x}_i - \mathbf{x}; \gamma) J(y_{li}, d_l(\mathbf{x}_i; \gamma)),$$

$$d_l^*(\mathbf{x}; \gamma) = \arg\min I_{\text{local emp}}[d_l], \tag{1}$$

where kernel G is a bounded and even function on X that is peaked about 0, γ denotes the locality control parameters associated to G and $\{d_l^*(\mathbf{x}; \gamma), l = 1, \ldots, C\}$ are the optimal discriminant functions. If we use a constant approximation d_l (with respect to $\mathbf{x}$) of y_l and a quadratic loss $J(y_l, d_l) = (y_l - d_l)^2$, solving the equation $\partial I_{\text{local emp}}[d_l]/\partial d_l = 0; l = 1, \ldots, C$ yields

$$d_l^*(\mathbf{x}; \gamma) = \sum_{i=1}^{M} G(\mathbf{x}_i - \mathbf{x}; \gamma) y_{li} \Big/ \sum_{i=1}^{M} G(\mathbf{x}_i - \mathbf{x}; \gamma). \tag{2}$$

As we see from inspecting (2), this optimal local discriminant function computes kernel-based estimates of the posterior probability of the classes [4, 2.5]. The shape of the kernel G controlled by γ then determines different well-known solutions to the same original problem. If $G = G_{\text{H}}$ where G_{H} is a square kernel whose width is adjusted to contain exactly k samples then Eq. (2) is the k-NN algorithm. On the other hand, if $G = G_{\text{S}}$ where G_{S} is a smooth kernel with a width modulated by a locality parameter σ then Eq. (2) is the *Parzen window* algorithm.

The k-NN technique considers a hypersphere centred at the testing pattern $\mathbf{x}$ that exactly contains K training samples. This feature prevents the estimator from not having enough training data to ensure good generalisation. Although in practice, their estimates poorly approximate posterior probabilities, since the region around $\mathbf{x}$ is not small enough to keep the estimations, a ratio of integers (K_l/K), valid. By contrast, the Parzen window algorithm can estimate better since it is based on a smooth interpolation between training points, although it does not effectively control the trade-off between the capacity of the learning device and the number of training samples. This is due to using a fixed width parameter σ for all the training points. If the distribution of the training patterns in the input space is uneven; it will be impossible to ensure that there are enough training data that significantly contribute to the estimations for all testing pattern $\mathbf{x}$.

We propose an approach that benefits from the virtues of these two techniques and can be understood as a compromise between them: for each testing pattern $\mathbf{x}$, a Parzen window estimate is computed using the K nearest (in the Euclidean sense) training points. In fact, this algorithm results from using a composed kernel $G_{\text{soft}} = G_{\text{H}} G_{\text{S}}$ in Eq. (2). As we show in the appendix, the resulting equation also estimates the posterior class probabilities. Eq. (2) is then called the *soft k-NN* algorithm and can be written as follows:

$$d_l^*(\mathbf{x}; \gamma) = \sum_{i=1}^{K_l} G(\mathbf{x}_i^l - \mathbf{x}; \gamma) \Big/ \sum_{j=1}^{C} \sum_{u=1}^{K_j} G(\mathbf{x}_u^j - \mathbf{x}; \gamma), \tag{3}$$

where $C_{k-\text{NN of class } j} = \{\mathbf{x}_u^j, u = 1, \ldots, K_j\}$ are those prototypes of class j among the k nearest prototypes to $\mathbf{x}$ and $K = \sum_{j=1}^{C} K_j$.

We remark that the (crisp) k-NN algorithm in Eq. (3) 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.

3. The adaptive soft k-NN classifier

Given a training set $\mathbf{T}$, we can consider to apply Eq. (2) using G_{soft}, classifying a new input to those classes whose discriminant function has the highest value. However, when training set size is large, storage and processing requirements make this simple algorithm unusable in engineering applications. In the case of (crisp) k-NN classifiers, many heuristic sophistication of the basic algorithm allows to use fewer data points (e.g. Hart's

Condensed NN [5]). By contrast, replacing the local kernel estimation with a simplified adaptive mixture model [6] can simply extend our approach. Suppose we define for each class density function a mixture as follows:

$$f_{X|C_l}(\mathbf{x})P(C_l) = \frac{1}{M}\sum_{j=1}^{M} f_{X|C_{l,j}}(\mathbf{x}), \tag{4}$$

where $\{f_{X\setminus Cl}, l = 1, \ldots, C\}$ are the class density probabilities, $\{P(C_l), l = 1, \ldots, C\}$ are the priors and $\{f_{X\setminus Cl,i} i = 1, \ldots, M, l = 1, \ldots, C\}$ are known parametric models restricted to the form $G(\mathbf{x} - \mathbf{w}_i^l; \gamma)$ with $G(\mathbf{u})$ a decreasing function on U and peaked about $\mathbf{u} = 0$. Applying Bayes' theorem, we arrive at the following expression for the posterior probabilities:

$$P(C_l|\mathbf{x}) = \frac{\sum_{j=1}^{M} f_{X|C_{l,j}}(\mathbf{x})}{\sum_{i=1}^{C}\sum_{u=1}^{M} f_{X|C_{i,u}}(\mathbf{x})}. \tag{5}$$

Since each model of the mixture is locally defined, we can approximate Eq. (5) using only the k-nearest-models to the testing pattern $\mathbf{x}$. If we use this simplification to construct the classifier, then the discriminant functions give

$$P(C_l|\mathbf{x}) \approx \frac{\sum_{j=1}^{k_i} f_{X|C_{l,j}}{}^{k}(\mathbf{x})}{\sum_{i=1}^{C}\sum_{u=1}^{k_i} f_{X|C_{i,u}}{}^{k}(\mathbf{x})}, \tag{6}$$

where $\{f_{X|C_{i,u}}{}^{K}(\mathbf{x}), i = 1, \ldots, C, u = 1, \ldots, K_i\}$ are the k-nearest-models to $\mathbf{x}$. Like the mixture models are locally defined around their centres $\mathbf{w}_i^j$, $P(C_l\setminus\mathbf{x})$ can be approximated by

$$P(C_l|\mathbf{x}) \approx \frac{\sum_{i=1}^{K_l} G(\mathbf{m}_i^l - \mathbf{x}; \gamma)}{\sum_{j=1}^{C}\sum_{u=1}^{K_j} G(\mathbf{m}_u^j - \mathbf{x}; \gamma)}, \tag{7}$$

where $\{\mathbf{m}_u^j, u = 1, \ldots, K_j, j = 1, \ldots, C\}$ are the K_j-nearest-mixture-centres to $\mathbf{x}$ and $K = \sum_{j=1}^{C} K_j$.

The total number of mixture models (M*C) will be typically much smaller than the number of training points (N), and the mixture models centres ($\mathbf{w}_i^j$) are no longer constrained to coincide with the training points. The parameters of the mixture models could be estimated using unsupervised procedures as the EM algorithm [7], although the followed approach here make use of a supervised learning procedure based on minimising the number of misclassifications.

3.1. The loss function

Consider a training set of random pairs $\mathbf{T} = \{(\mathbf{x}_i, \mathbf{y}_i)\} i = 0, \ldots, N-1$. If the pattern $\mathbf{x}_i$ belongs to the nth class, the nth coefficient of $\mathbf{y}_i$ is equal to 1 and the other coefficients are equal to 0. As a loss function we propose a modification of the cross-entropy error function for multiple classes (See Section 6.9. in Bishop [4]):[1]

$$L = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{l=1}^{C} y_{li} d_l(\mathbf{x}_i). \tag{8}$$

The absolute minimum with respect to the $\{d_l(\mathbf{x}_i)\}$ occurs when all the training points are correctly classified, that is when $d_l(\mathbf{x}_i) = y_{li}$ for all l and i. In the limit in which the size N of the training set goes to infinity, we can substitute the sum over patterns with the following integral:

$$\langle L\rangle = \lim_{N\to\infty} L = -\int_{\mathscr{R}^d}\sum_{l=1}^{C} y_l(\mathbf{x}) d_l(\mathbf{x}) f_X(\mathbf{x})\,\mathrm{d}\mathbf{x}, \tag{9}$$

where f_X is the probability density of the input space. Since $y_l = \mathbf{1}(\mathbf{x}\in \text{class } l)$ where $\mathbf{1}$ is the indicator function

$$1(\text{condition}) = \begin{cases} 1 & \text{if condition is true,} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

and $f_X(\mathbf{x}) = \sum_{i=1}^{C} f_{X|C_i}(\mathbf{x})P(C_i)$, Eq. (9) can be easily rewritten as

$$\langle L\rangle = -\sum_{i=1}^{C}\left(\int_{R_l} d_l(\mathbf{x}) f_{X|C_i}(\mathbf{x})P(C_i)\,\mathrm{d}\mathbf{x} + \int_{R_l}\sum_{\substack{i=1\\ i\neq l}}^{C} d_i(\mathbf{x}) f_{X|C_i}(\mathbf{x})P(C_i)\,\mathrm{d}\mathbf{x}\right), \tag{11}$$

where R_l are the decision region of class l, that is the input region in which d_l has the highest value among discriminant functions. $\langle L\rangle$ would be minimised when $d_l = 1$ for all $\mathbf{x}$ in R_l and $f_{X\setminus Cl}P(C_l) = \max f_{X\setminus C_i}P(C_i)$ for all R_l. But as we know the best-possible choice for d_l is $P(C_l\setminus\mathbf{x})$. If we use an optimisation method to minimise $\langle L\rangle$, the algorithm will try to minimise the number of misclassifications it will be possible a convergence of d_l to $P(C_l\setminus\mathbf{x})$ (at least near Bayes Borders). In this case if class overlapping is moderate, the maximum expected number of correct classifications could approximate the absolute value of $\langle L_{\min}\rangle$,

$$|\langle L_{\min}\rangle| \approx \sum_{l=1}^{C}\int_{R_l}\hat{P}(C_l|\mathbf{x}) f_{X|C_l}(\mathbf{x})P(C_l)\,\mathrm{d}\mathbf{x}. \tag{12}$$

This latter result gives us, when N is large enough, a possible tendency of the system from minimising Eq. (8) during the learning phase.

3.2. The learning algorithm

In real-life pattern recognition problems training sets usually are large and high dimensional. So algorithmic simplicity of the learning procedure, or the optimisation

[1] We will see in the next subsection, when we derive the learning equations, why Eq. (8) can be of more practical interest than cross-entropy error.

process, is a requirement in order to achieve a solution with moderated computational resources. Online gradient descent algorithms [8] are one of the simplest computational approaches to the learning problem. Other features of these algorithms include better convergence speed than their batch counterparts in redundant training sets (e.g. real-world databases) and good mathematical characterisation [8,9].

3.2.1. Derivation of the on-line gradient learning algorithm

We restrict the mixture model parameters to centres and a global defined parameter σ that controls locality.

$$H(\mathbf{w}_i^j, \mathbf{x}) = \begin{cases} -d_W(\mathbf{x})(1 - d_j(\mathbf{x}))(\mathbf{x} - \mathbf{w}_i^j) & \text{if } \mathbf{w}_i^j \in C_{k-NN} \text{ and } j = \text{class index } (\mathbf{x}), \\ d_W(\mathbf{x}) d_j(\mathbf{x})(\mathbf{x} - \mathbf{w}_i^j) & \text{if } \mathbf{w}_i^j \in C_{k-NN} \text{ and } j \neq \text{class index } (\mathbf{x}), \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

So the goal in the learning phase is reduced to compute a set of labelled codebooks $C_j = \{\mathbf{w}_i^j, i = 1, \ldots, M_j\}$, $j = 1, \ldots, C$, one for each class, that minimises an estimator of the expected loss function given in Eq. (9):

$$\langle L \rangle(\{C_j, j = 1, \ldots, C\}) = E_X[Q(\mathbf{x}, \{C_j, j = 1, \ldots, C\})]. \tag{13}$$

Since $E_X[\bullet]$ is unknown, an empirical estimator is formed with a (training) set of random samples pairs $\mathbf{T} = \{(\mathbf{x}_i, \text{class index } (\mathbf{x}_i))\} i = 0, \ldots, N-1$. The online gradient learning algorithm estimates using the instantaneous loss function $Q(\mathbf{x}, \{C_j, j = 1, \ldots, C\})$. Each step of this algorithm consists of picking up (cyclically or randomly) one sample pair from the training set $\mathbf{T}$ and applying the following update equation:

$$\mathbf{w}_i^j[k] = \mathbf{w}_i^j[k-1] - \eta_i^j[k] H(\mathbf{w}_i^j[k-1], \mathbf{x}[k]), \tag{14}$$

The learning rates $\eta_i^j[k]$ are positive numbers and are usually made to decrease monotonically with discrete time k, although they can be also constant with time. The term $H(\mathbf{w}_i^j, \mathbf{x})$ is defined as

$$H(\mathbf{w}_i^j, \mathbf{x}) = \begin{cases} \nabla_{\mathbf{w}_i^j} Q(\mathbf{x}, \{C_j, j = 1, \ldots, C\}) & \text{when differentiable,} \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Given a training set of infinite size, the necessary (but not sufficient, see Section 5 in Bottou [8]) condition to ensure the convergence of the iterative (14) to a local minimum of $\langle L \rangle$ is that

$$E_X[H(\mathbf{w}_i^j, \mathbf{x})] = \nabla_{\mathbf{w}_i^j} \langle L \rangle(\{C_j, j = 1, \ldots, C\}). \tag{16}$$

Since Q is not differentiable on a finite number of points and the iterative algorithm has a zero probability to reach them, this condition is met (see Bottou [8, p. 10]). In the case of using a constant η, we might ensure that η is small enough (see Benveniste [9, p. 48]). Practically, an optimal value of η can be estimated using a validation set.

Let $G(\mathbf{x} - \mathbf{w}_i^j)$ be $\exp(-\|\mathbf{x} - \mathbf{w}_i^j\|^2/\sigma)$, $\langle L \rangle' = \sigma/2\langle L \rangle$ and $\eta_i^j[k] = \eta$. We have modified the loss function for analytical convenience since the added constant does not affect the desired points of convergence. Hence Eq. (15) yields

where $C_{k-\mathrm{NN}}$ are those prototypes of class j among the k nearest prototypes (in the Euclidean sense) to $\mathbf{x}$ and d_W is equal to

$$d_W(\mathbf{x}) = \frac{\exp(-\|\mathbf{x} - \mathbf{w}_i^j\|^2/\sigma)}{\sum_{m=1}^{C} \sum_{u=1}^{K_m} \exp(-\|\mathbf{x} - \mathbf{w}_u^m\|^2/\sigma)}. \tag{18}$$

In case σ goes to infinity $\exp(\bullet/\sigma)$ tends to the unity. So H can be simplified since d_W tends to $1/K$ and d_j to K_j/K. Finally, the learning equation gives

$$\begin{aligned} &\text{if } \mathbf{w}_i^j[k-1] \in C_{k-\mathrm{NN}}(\mathbf{x}[k]) \text{ and } j = \text{class index } (\mathbf{x}[k]) \\ &\quad \mathbf{w}_i^j[k] = \mathbf{w}_i^j[k-1] + \eta d_W(\mathbf{x}[k])\,(1 - d_j(\mathbf{x}[k]))\,(\mathbf{x}[k] \\ &\qquad - \mathbf{w}_i^j[k-1]), \\ &\text{if } \mathbf{w}_i^j[k-1] \in C_{k-\mathrm{NN}}(\mathbf{x}[k]) \text{ and } j \neq \text{class index } (\mathbf{x}[k]) \\ &\quad \mathbf{w}_i^j[k] = \mathbf{w}_i^j[k-1] - \eta d_W(\mathbf{x}[k]) d_j(\mathbf{x}[k])\,(\mathbf{x}[k] \\ &\qquad - \mathbf{w}_i^j[k-1]), \end{aligned} \tag{19}$$

otherwise,

$$\mathbf{w}_i^j[k] = \mathbf{w}_i^j[k-1],$$

where $\mathbf{x}[k] = \mathbf{x}_{(k-1) \bmod N}$, $k \geqslant 0$ if we pick up cyclically.

3.2.2. Properties of the learning equation

As we pointed out before, the proposed loss function has the same minimum points than the cross-entropy error function

$$L_{\text{cross-entropy}} = -\frac{1}{N} \sum_{i=0}^{N-1} \sum_{l=1}^{C} y_{li} \ln d_l(\mathbf{x}_i). \tag{20}$$

If we derive the on-line gradient equations for this error function, we obtain Eq. (17) divided by $d_j(\mathbf{x})$. So practical differences between performing on-line gradient descent using one loss function or another must be determined with an analysis of how this term can affect to the

evolution of the optimisation process. Suppose that several outliers are present in the training set. This is a typical situation in real-world databases. When we pick up one outlier of class j, the amount of change in a winning prototype w_i^j will be

$$\Delta \mathbf{w}_{i\ \mathrm{cross-entropy}}^{j} = d_W(\mathbf{x}) \frac{1 - d_j(\mathbf{x})}{d_j(\mathbf{x})} (\mathbf{x} - \mathbf{w}_i^j). \tag{21}$$

Since it is probable that $d_j(\mathbf{x})$ takes a low value when $\mathbf{x}$ is an outlier of class j, the term $(1 - d_j(\mathbf{x}))/d_j(\mathbf{x})$ tends to infinity. In this way $\Delta \mathbf{w}_{i\ \mathrm{cross-entropy}}^{j}$ can be arbitrarily large and the normal evolution of the learning process can be enormously degraded by the presence of outliers. By contrast, if we use our proposed loss function, outliers have a little impact since the term d_w dominates the amount of change and tends to zero since the distance between the winning prototype and the outlier can be large.

4. Empirical results

In this section, we compare Kohonen's LVQX algorithms [10] (where X is 1, 2 and 3)) and K-NN classifiers with our proposal. We have used the NIST (uppercase and lowercase) hand-written characters database [3], which can be found in directories train/hsf_4, train/hsf_6 and train/hsf_7. We have split directory train/hsf_7 in two sets: one for validation (1100 samples) and one for test (over 10 000 samples) preserving the original class distribution in data. The other two directories were chosen for training (above 24 000 samples). These images (32×32 pixels) have been pre-processed with a principal component analyser that extracts 64 components from each image (NIST's mis2evt utility). The correlation matrix of the PCA was computed using the training set.

4.1. Simulations

Ten runs for each classification problem, learning algorithm and several codebook sizes have been made. We have initialised the codebooks using the LVQ_PAK's program [11]. Then we have applied every learning algorithm until classification error in validation set increases or stands. (We have monitored this error every 200 000 steps. In the case of LVQX algorithms, every time we monitored the error, we restart its α value). Optimal parameters of algorithms have been estimated using the validation set.

4.2. Results

Two main aspects are of interest in this section. The first one would be concerned with the comparison of our classifier and their crisp counterparts when our system is not constrained to use also the crisp k-NN estimation. The second one would compare the classifier when all of them make use of this classification rule. In this way we can evaluate if the use of a more complex classification procedure, the soft k-NN estimation, is of practical interest and, secondly, what happens when we force our system to compete with other well-established methods to design prototypes sets for crisp k-NN classifiers.

Tables 1 and 2 summarise the average classification error on uppercase and lowercase test sets. Each result is the average of 10 trials for each learning algorithm (except in the case of k-NN classifiers that the test error is computed once using the whole training set as the prototypes set). In Table 1, we show the average misclassification rate of our proposed classifier in four different cases. In the first case, we fix the number of nearest-prototypes (k) equal to 2 and σ to infinity. While in the second case with k fixed to 2, we make use of the best σ value estimated by cross-validation. In these two first cases, the resulting classification systems are crisp 1-NN classifiers since only two nearest-prototypes contribute to form the soft estimations. In the third case, we fix σ to infinity and make use of the best-estimated k value (with k greater than 2). As above, our classifier then is the crisp k-NN classifier. Finally, in the fourth case, we constrain k to be greater than 2 and allow σ to take a finite value. Both parameters are estimated with the validation set using

Table 1
Experimental results for our proposed classifier. We show for each algorithm the average test misclassification rate over 10 runs. The number in parenthesis denotes the value of k

Codebook size	Upper				Lower			
	$\sigma \to \infty$ $K = 2$	Best $\sigma < \infty$ $K = 2$	$\sigma \to \infty$ Best $K > 2$	Best $\sigma < \infty$ Best $K > 2$	$\sigma \to \infty$ $K = 2$	Best $\sigma < \infty$ $K = 2$	$\sigma \to \infty$ Best $K > 2$	Best $\sigma < \infty$ Best $K > 2$
100	18.2	14.52	25.95(3)	11.94(5)	24.7	20.92	31.01(3)	18.8 (11)
200	10.98	10.63	15.8(3)	9.01(5)	17.62	17.26	22.69(3)	15.93(5)
400	7.36	7.97	10.09(3)	6.53(5)	13.3	15.28	17.13(3)	14.02(5)
800	5.45	7.22	5.74(3)	5.38(11)	11.28	14.11	11.82(3)	12.54(5)

Table 2
Experimental results for LVQ-based and k-NN classifiers. We show for each algorithm the average test misclassification rate over 10 runs except in the case of k-NN classification. In k-NN classifiers the number in parenthesis denotes the value of k

Codebook size	Upper				Lower			
	LVQ1	LVQ2	LVQ3	k-NN	LVQ1	LVQ2	LVQ3	k-NN
100	21.5	15.82	16.02	—	27.89	22.4	22.29	—
200	15.3	9.941	11.4	—	20.85	16.81	18.39	—
400	11.8	8.24	9.08	—	17.27	14.84	15.54	—
800	9.61	8.23	8.26	—	15.53	14.06	14.43	—
Whole database	—	—	—	$5.84_{(1)}$	—	—	—	$13.1_{(7)}$

these constraints. Table 2 shows classification results of 1-NN classifiers whose codebooks are designed using Kohonen's LVQ algorithms and crisp k-NN that use the whole training database as the prototypes set.

The condensed soft k-NN classifiers (the fourth case) always outperforms crisp 1-NN classifiers based on LVQ algorithms for each tested codebook size. In uppercase and lowercase recognition, best results are achieved when codebook size is 800. In this case, our classification systems also outperform k-NN classifiers that use the whole training database as the prototypes set. On average, our learning system correctly recognised 94.62% of the uppercase hand-written letters and 87.46% of the lowercase hand-written characters, in front of 91.77 and 85.94%, respectively for LVQ2, the best of the LVQ algorithms. Differences between our procedure and LVQ algorithms are notably greater as codebook size decreases. We also note that the evolution of the misclassification rate, as codebook size grows, is less abrupt in our procedure. This behaviour suggests that our procedure achieves a better graceful degradation in the classification accuracy as the codebook size is smaller.

Our best crisp 1-NN classifiers, the best solutions for the first two cases, clearly outperform LVQ-based crisp 1-NN classifiers except in one case. When the codebook size is equal to 200, codebooks designed with the LVQ2 algorithm are better than ours. It is interesting to compare the classification accuracy between cases 1 and 2 since both learning algorithms design codebooks for crisp 1-NN classifier although they differ in the way of doing it. While the first algorithm uses an infinite σ, the second one do not and can better define how local the solution is, since σ regulate how many training data contribute to update each prototype during the learning phase. In both classification problems we observe the same behaviour. For small codebooks (100 and 200), the effect of using a finite σ allows to achieve an optimal solution. However, for medium codebooks, the use of an infinite σ works better.

Classifiers of the third case only achieve better classification accuracy than k-NN classifiers (that use over 25 000 prototypes) when their codebook size is 800. Although these classifiers are inferior to the best solutions achieved in the above crisp cases. Finally, we compare the best solutions of cases 1, 2 and 4. We can observe that the soft estimation is, in 6 out of 8 cases superior than the crisp estimation. Only in lowercase hand-written recognition, the crisp estimation is superior on average when the codebook sizes are 400 and 800.

5. Summary

Crisp k-nearest-neighbour classifiers estimate posterior class probabilities using a fixed number of training data. This feature prevents them, unlike other approaches, from having in some input regions enough data to ensure good generalisation, although the quality of their estimations (a simple ratio of integers) is too poor to properly approximate the true probabilities when finite data sets are employed.

An extension of k-NN estimation was presented in order to overcome this limitation: for each testing pattern $\mathbf{x}$, a Parzen window estimate is computed using the K nearest (in the Euclidean sense) training points. This simple algorithm also estimates the posterior class probabilities although all the training data points must be retained in order to compute the estimations.

An improvement of this approach can be achieved, in terms of storage and processing requirements, by replacing the local kernel estimation with a simplified adaptive mixture model: since each model of the mixture is locally defined, for each testing pattern we only take into account its k-nearest models. The total number of mixture models will be typically much smaller than the number of training points and the mixture models centres are no longer constrained to coincide with the training points. We have proposed a supervised learning procedure to estimate the mixture models' parameters based on minimising a modification of the cross-entropy error function. This error function has the same minimum points as that of the cross-entropy error function while it allows

the derived on-line gradient learning algorithm to have a more robust behaviour in presence of outliers in the training set.

Experimental results in two hand-written classification problems demonstrate the potential and versatility of the proposed classification system. Since the soft k-NN classifier includes as a special case the crisp version, in numerical simulations we always find an optimal solution that achieves better classification accuracy than crisp 1-NN classifiers designed with LVQ algorithms and k-NN classifiers.

Acknowledgements

The first author is actually supported by CIRIT FI grant 97/0621. This research was supported in part by Spanish CICYT action TIC96-0889.

Appendix A. The soft k-NN kernel estimation

Consider a training set of random pairs $\mathrm{T} = \{(\mathbf{x}_i, \text{class label } (\mathbf{x}_i))\}$, $i = 0, \ldots, N-1$ where $\mathbf{x}_i$ is a random sample belonging to one of the C existing classes. Suppose we take the region $R(\mathbf{x})$ to be a small hypersphere centred at the point $\mathbf{x}$ that contains precisely k training data points. Let us denote R as the event X belongs to $R(X)$. This event is always true due to any vector which belongs to a region centred at itself. So the following equation can be written:

$$P(C_l|\mathbf{x}) = P(C_l|\mathbf{x}, R). \tag{22}$$

Making use of Bayes' theorem in the right hand-side of Eq. (A.1) we have

$$P(C_l|\mathbf{x}, R) = \frac{f_{X|C_l,R}(\mathbf{x})P(C_l|R)}{f_{X|R}(\mathbf{x})}. \tag{23}$$

If the training data belonging to class l that fall in $R(\mathbf{x})$ are denoted by $\{(\mathbf{x}_i^l)\} i = 1, \ldots, K_l l = 1, \ldots, C$, we obtain the following Parzen window estimates [4, Eq. (2.57)] for the probability density functions $f_{X\setminus R}$ and $f_{X\setminus C_l,R}$:

$$f_{X|C_l,R}(\mathbf{x}) \approx \frac{1}{K_l}\sum_{i=1}^{K_l}\frac{1}{h^d}G\left(\frac{\mathbf{x}-\mathbf{x}_i^l}{h}\right),$$

$$f_{X|R}(\mathbf{x}) \approx \frac{1}{K}\sum_{j=1}^{C}\sum_{m=1}^{K_j}\frac{1}{h^d}G\left(\frac{\mathbf{x}-\mathbf{x}_m^j}{h}\right). \tag{24}$$

Since the probability of belonging to class l conditioned to belong to region R, $P(C_l\setminus R)$ can be approximated by K_l/K, we have as an estimation of $P(C_l\setminus x)$ as

$$P(C_l|\mathbf{x}) \approx \frac{\sum_{i=1}^{K_l} G((\mathbf{x}-\mathbf{x}_i^l)/h)}{\sum_{j=1}^{C}\sum_{m=1}^{K_j} G(\mathbf{x}-\mathbf{x}_m^j)/h)}. \tag{25}$$

References

[1] B.V. Darasay (Ed.), Nearest neighbor pattern classification techniques, IEEE Computer Society Press, Los Alamitos, LA, 1991.
[2] L. Bottou, V. Vapnik, Local learning algorithms, Neural Comput. 4 (6) (1992) 888–890.
[3] M. Garris et al., NIST form-based handprint recognition system (Release 2.0), National Institute of Standards and Technology, 1997.
[4] C.M. Bishop, Neural networks for pattern recognition, Oxford University Press, Oxford, 1995.
[5] P.E. Hart, The condensed nearest neighbor rule, IEEE Trans. Inform. Theory 14 (1968) 515–516.
[6] G.J. McLachlan, K.E. Basford, Mixture models, Inference and applications to clustering, Marcel Dekker, New York, 1988.
[7] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. Roy. Statist. Soc. Ser. B 39 (1977) 1–38.
[8] L. Bottou, Online learning and stochastic approximations, Technical Report, AT & T Labs-Research, 1996.
[9] A. Benveniste, M. Métivier, P. Priouret, Adaptive algorithms and stochastic approximations, Springer, New York, 1990.
[10] T. Kohonen, Self-organizing Maps, 2nd Edition, Springer, New York, 1996.
[11] T. Kohonen et al., LVQ_PAK. The learning vector quantization program package. Version 3.1, Helsinki University of Technology, 1995.

About the Author—SERGIO BERMEJO received the M.Sc. Degree in Telecommunication Engineering in 1996 from Universitat Politècnica de Catalunya (UPC). Since then he has been a research assistant at the Department of Electronic Engineering of the UPC. He is a holder of a CIRIT FI fellowship from Generalitat of Catalonia and is currently working towards a Ph.D. degree. His research interests include statistical pattern recognition and machine learning.

About the Author—JOAN CABESTANY holds currently a Professor position at the Department of Electronic Engineering of the Universitat Politècnica de Catalunya (UPC). He obtained the M.Sc. Degree and Ph.D. Degrees in Telecommunication Engineering in 1976 and 1982, respectively, both from the Universitat Politècnica de Catalunya. His research interests include Analog and Digital Electronic Systems Design, Configurable and Programmable Electronic Systems, and Neural Networks Models and applications.