Pergamon

Pattern Recognition, Vol. 30, No. 2, pp. 281–288, 1997
Copyright © 1997 Pattern Recognition Society. Published by Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0031–3203/97 $17.00+.00

PII:S-0031-3203(96)00072-6

FINDING PROTOTYPES FOR NEAREST NEIGHBOUR CLASSIFICATION BY MEANS OF GRADIENT DESCENT AND DETERMINISTIC ANNEALING

CHRISTINE DECAESTECKER*
Institute of Interdisciplinary Research and Development in Artificial Intelligence (IRIDIA), Université Libre de Bruxelles-CP 194/6, 50 Av. F. Roosevelt, Brussels 1050, Belgium

(*Received* 13 *December* 1995; *in revised form* 25 *March* 1996; *received for publication* 13 *May* 1996)

Abstract—A new method is presented to find prototypes for a nearest neighbour classifier. The prototype locations are optimised through a gradient descent and a deterministic annealing process. The proposed algorithm also includes an initialisation strategy which aims to provide the maximum classification rate on the training set with the minimum number of prototypes. Experiments show the efficiency of this algorithm on both real and artificial data. Copyright © 1997 Pattern Recognition Society. Published by Elsevier Science Ltd.

Nearest neighbour   Classifier   Supervised learning   Prototypical reference vectors
Optimisation   Gradient   Annealing

1. INTRODUCTION

Nearest neighbour classification is one of the best known pattern classification methods. This is a distance-based technique which classifies an unknown case by looking at the classes of the nearest cases in a set of reference instances. It is well known that these classifiers suffer from some drawbacks such as needing a lot of memory and computational resources (for large data sets), and also from the difficulties in determining an optimal value for the number of nearest neighbours to consider in the allocation rule (especially in a noisy environment). This is why numerous studies have been carried out on this technique since the 70s, as is shown by the excellent survey by Dasarathy.(1) For example, editing and condensing methods have been developed to reduce the number of reference instances to be stored. However, these methods, which select some instances among the available ones, cannot avoid a degradation of the classifying performance.(1)

Another approach, initiated by Chang(2) and now developed in pattern classification as well as in neural networks,(3–5) consists of generating new prototype reference vectors instead of selecting reference cases from among the available instances. This approach is a kind of supervised (i.e. class-dependent) vector quantisation, called the "Learning Vector Quantisation (LVQ)" by Kohonen(4), where a discrimination purpose is taken into consideration. With computational efficiency (in the classification phase) as their objective, prototype-based classifiers also aim to improve generalisation performances. It should be noted that any classification approach based on nearest neighbour concepts makes the hypothesis that the proximity in the data space generally expresses the membership of the same class (and thus for each class, there are regions of instance space where the majority of cases belong to the given class). A data set which is in complete opposition to this condition cannot be treated by such an approach and will not be considered in the present paper.

The aim of the present work is to propose a new prototype-based classifier for which the prototype locations are optimised through a gradient descent mixed with a deterministic annealing process. This ensures the distribution of the prototypes within each class while minimising the total classification error rate at the end of the process. This classifier also has a "natural" implementation in terms of an artificial neural network similar to Kohonen's LVQ network.(4) This aspect will not be detailed in this paper (but can be found in a technical report(6)).

One important issue of this work is the control of the effective complexity of the model (here, the number of prototypes) to ensure good generalisation capacities. In practice, this means that the model complexity has to be optimised for a given training data set. One of the most frequently employed techniques (as in the case of neural networks) is cross-validation. However, this method may well require a large amount of processing time. In the present work, complexity control is realised without the use of cross-validation. Firstly, an initialisation strategy gives an initial set of prototypes as a function of the training data set. This heuristic process includes a pruning strategy aimed at providing the maximum classification rate on the training set with the minimum number of prototypes. Hence, redundant prototypes are

* Tel.: +32 2 650 3165; fax: +32 2 650 2715; e-mail: cdecaes@ulb.ac.be.

eliminated and the remaining prototypes are forced to cover larger areas in the pattern space. The second one is a consequence of the optimisation process itself. This process is able to completely inactivate some prototypes. These useless prototypes must be suppressed in order to realise the classification task.

Experiments show that the classification rate (on test data sets) of the proposed prototype-based classifier is as good in noise-free context as the nearest neighbour classifier performances and, indeed, better in the case of noisy data. In relation to traditional neural networks, Multilayer Perceptron (MLP), Radial Basis Function (RBF) and LVQ networks, and symbolic classifiers (Decision tree), the proposed classifier gives a degree of accuracy comparable with the best classifier. Furthermore, the complexity of the proposed model (number of prototypes) is generally at its lowest when compared to the other classifiers (e.g. the number of hidden units for neural networks and the number of rules for decision trees).

2. PROTOTYPE OPTIMISATION

2.1. *Initialisation strategy*

As stated in Section 1, an initialisation strategy is used to propose initial prototypes which will be adapted during the optimisation process. This strategy tries to generate a "reasonable" initial number of prototypes in accordance with the complexity control purpose explained above. The strategy operates in three phases, mixing an unsupervised search for prototypes in each class with a supervised elimination of inaccurate prototypes. Firstly, a standard k-means is performed separately on each class, with a large number of training instances randomly chosen as initial centroids (in practice, between 10 and 20 per class, depending on the training set size; see the experiments). This gives a good chance that each cluster in the training set will be represented by at least one prototype. Secondly, an elimination rule proposed by Kohonen *et al.*(7) is applied to discard each non-representative prototype, i.e. if the majority of the k nearest neighbours of the prototype in the training set does not have the same class label (in practice, $k = 3$). Thirdly, a second elimination rule proposed by Van de Merckt(8) discards redundant prototypes, i.e. ones which can be eliminated without decreasing the classification quality of the training set (with respect to a nearest neighbour classifier). To do so, the prototypes are sorted as functions of the size of their corresponding cluster (number of patterns attracted in the training set). Following the increasing order so defined, the classification rate is computed on the training set in both the presence and the absence of each prototype. The prototypes which do not contribute to a strict increase in the classification rate are eliminated.

2.2. *Optimisation process*

Let $\boldsymbol{P}_1, \boldsymbol{P}_2, \dots, \boldsymbol{P}_i, \dots, \boldsymbol{P}_n$ be the location vectors (coordinates in the data space) of the prototypes resulting from the initialisation strategy. Each prototype $\boldsymbol{P}_i$ is representative of a cluster of patterns C_i inside a given class *a priori* defined in the data set. As is used by Rose *et al.*(9–11) in clustering, the probability that a pattern $\boldsymbol{X}$ belongs to the cluster C_i is defined as follows:

$$Z_i(\boldsymbol{X}) = \frac{\exp[-d_i^2/T]}{\sum_k \exp[-d_k^2/T]}, \qquad (1)$$

where T is a "temperature" parameter that decreases during training (see Section 2.3) and d_i is the Euclidean distance (or another) between the two vectors $\boldsymbol{X}$ and $\boldsymbol{P}_i$ ($d_i = \|\boldsymbol{X} - \boldsymbol{P}_i\|$). Equation (1) defines a Gibbs distribution for all the possible allocations of a pattern to one of the n clusters defined by the prototypes. Hence the following property holds:

$$\sum_i Z_i(\boldsymbol{X}) = 1. \qquad (2)$$

We now define a cost function to optimise the prototype locations. As is usual in neural network approaches, the cost function is a measure of the classification error and is defined as a function of some "desired" answers imposed on the model. Similarly, desired probabilities $Z_i^*(\boldsymbol{X})$ can be defined for our prototype-based model. They express the desired condition that a pattern, $\boldsymbol{X}$, belongs to cluster C_i with zero probability if $\boldsymbol{X}$ and $\boldsymbol{P}_i$ do not belong to the same class. Hence, let us define $C(\boldsymbol{X}) = \{k \mid \text{prototype } \boldsymbol{P}_k \text{ belongs to the class of } \boldsymbol{X}\}$. Desired probabilities $Z_i^*(\boldsymbol{X})$ are then

$$Z_i^*(\boldsymbol{X}) = \begin{cases} 0 & \text{if } i \notin C(\boldsymbol{X}), \\ \left(\frac{\exp[-d_i^2/T]}{\sum_{k \in C(X)} \exp[-d_k^2/T]}\right) & \text{if } i \in C(\boldsymbol{X}). \end{cases} \qquad (3)$$

The above definition preserves property (2) by redistributing the probabilities not equal to zero to the prototypes of the correct class. In the following section we shall see that, as the temperature decreases, the desired output of the prototype nearest to the correct class gradually comes closer to 1.

Now let us define the cost function in terms of the differences between observed and desired probabilities:

$$E(\boldsymbol{X}) = \frac{1}{2} T \sum_i (Z_i^*(\boldsymbol{X}) - Z_i(\boldsymbol{X}))^2, \qquad (4)$$

where temperature, T, is introduced in order to simplify the gradient expression (see equation (6)).

Finally, the optimisation of prototype locations $\boldsymbol{P}_i$ is performed through an adaptive gradient procedure which minimises the error function given in equation (4). After each presentation of pattern $\boldsymbol{X}_{t+1}$ we modify each prototype location, $\boldsymbol{P}_i$, according to

$$\boldsymbol{P}_i(t+1) = \boldsymbol{P}_i(t) + \Delta\boldsymbol{P}_i(t+1), \qquad (5)$$

with $\Delta\boldsymbol{P}_i(t+1) = -\eta(t+1)\nabla_i E(\boldsymbol{X}_{t+1}) + \mu\Delta\boldsymbol{P}_i(t)$, where $\nabla_i E$ is the gradient of E with respect to $\boldsymbol{P}_i$, $\eta(t)$ a gradient step which decreases over time and μ a momentum term (usually fixed at 0.9).

While desired probabilities Z^* are computed as functions of the prototype locations (see equation (3)), they are not considered in this way in the gradient process. The fact that the answers desired of a classifier depend on the values of the adaptive parameters (here, the prototype locations) does not make sense in an optimisation process. In our algorithm, it is only a device to obtain desired values which are coherent with the model and which allow the smooth adaptation of the parameters. In the gradient calculation these desired values are thus considered as constants independent of the prototype locations. In accordance with this hypothesis, we obtain (using the Kronecker notation δ_{ij})

$$-\nabla_i E = (\boldsymbol{X} - \boldsymbol{P}_i) Z_i(\boldsymbol{X}) \sum_j (Z_j^*(\boldsymbol{X}) - Z_j(\boldsymbol{X}))(\delta_{ij} - Z_j(\boldsymbol{X})). \tag{6}$$

Let us try to interpret the adaptation rule defined by equation (6). Note that each prototype P_j makes a contribution to $-\nabla_i E$, in the direction of $(\boldsymbol{X} - \boldsymbol{P}_i)$. Let β_j be this contribution, we have

$$\beta_j = (Z_j^*(\boldsymbol{X}) - Z_j(\boldsymbol{X}))(\delta_{ij} - Z_j(\boldsymbol{X})).$$

Thus,

(a) if $j = i$ and $j \in C(\boldsymbol{X})$ then $\beta_j > 0$;
(b) if $j = i$ and $j \notin C(\boldsymbol{X})$ then $\beta_j < 0$;
(c) if $j \neq i$ and $j \in C(\boldsymbol{X})\}$ then $\beta_j < 0$;
(d) if $i \neq j$ and $j \notin C(\boldsymbol{X})$ then $\beta_j > 0$.

Rule (4) thus expresses three complementary effects on the prototypes. First, there is a classical attraction–repulsion effect between $\boldsymbol{P}_i$ and $\boldsymbol{X}$ (according to whether or not they belong to the same class, see (a) and (b)). The second effect (see (c)) introduces a competition between the prototypes of the correct class and the third (see (d)) expresses a discrimination effect between the different classes.

2.3. *Deterministic annealing*

Annealing is introduced by varying T in the above optimisation process, i.e. starting with a high value and ending with a low one ($T > 0$).

We have the following properties:

$$\lim_{T \to 0} Z_i(\boldsymbol{X}) = \delta_{ip}, \tag{7}$$

where $\boldsymbol{P}_p$ is the nearest prototype to $\boldsymbol{X}$.

$$\lim_{T \to 0} Z_i^*(\boldsymbol{X}) = \delta_{it}, \tag{8}$$

where $\boldsymbol{P}_t$ is the nearest prototype to $\boldsymbol{X}$ and belongs to the same (i.e. correct) class.

The annealing process starts at high T, for which all the probabilities, Z_i, are more or less similar. The same observation can be made for desired probabilities Z_i^* not equal to zero in equation (3). Each input pattern thus influences all the prototypes of its class in a similar way, and (in another way) all the prototypes of the other classes (because $Z^* = 0$). During training, the gradual decrease of T increases the bias in favour of the nearest prototype belonging to the correct class (see equation (8)). Thus, at low temperatures we obtain a drastic classifier, and error function E measures the total error rate of classification on the training set (by substituting equations (7) and (8) in equation (4)).

Expression (1) can also be interpreted as a fuzzy membership function (of the cluster defined by prototype P_i). The annealing process then realises a progressive "defuzzing" of the prototype areas of influence which become "crisp" (Voronoi diagram) when the temperature is sufficiently low (near zero). However, the training process is stopped if the value of the error function (cumulated on the whole training set) remains stationary (or increases again), possibly before total defuzzing (see the algorithm given in Section 2.4). This depends on the data configuration.

In conclusion, the process distributes the prototypes in each class while minimising the total error rate of classification. Another property of annealing is that it helps to avoid local minima in function optimisation; this has been shown by previous experiments in clustering techniques.(9,10,11)

2.4. *Outline of the complete optimisation algorithm*

As stated in Section 1, the proposed optimisation process mixing a gradient descent and a deterministic annealing process is able to automatically prune the set of prototypes by "deactivating" some of them. This is principally due to the deterministic annealing component which progressively transforms the classifier into a nearest neighbour one. When T is sufficiently low, only the nearest prototype has a membership probability above zero (see equation (8)). Some prototypes may thus become inactive (they do not attract any training pattern) at the end of the optimisation process. These prototypes are then considered useless for the classification task and are eliminated.

By way of recapitulation, the outline of the prototype optimisation algorithm is as follows:

1. Choose an *arbitrary* set of patterns in each class as an initial set of prototypes.
2. Apply the initialisation strategy (detailed in Section 2).
3. Set gradient step $\eta(0)$ at a *low* value and temperature T at a *high* one.
4. Make one iteration on the whole training set in accordance with equation (5).
5. Decrease $\eta(t)$. If $T > T_{\min}$, decrease T.
6. If the process has converged (stabilisation of the error function or $\eta(t)$ is too small) go to step 7, otherwise go to step 4.
7. Detect and eliminate the inactivate prototypes.

Certain issues must be specified. Hence the number of initial prototypes in each class depends on the size of the data set (see experiments). This number must be sufficient to allow some eliminations in the initialisation strategy. While the initial value of the gradient step must be small (e.g. $\eta(0) = 0.05$), the precise value is not important because it decreases over time. In all our experiments, the decrease was fixed at 50% for each iteration ($\eta(t+1) = 0.5\eta(t)$). The initial T value is modulated as a function of the difficulty of the classification task (evaluated on the training set). If the classification rate on the training set is already high (e.g. $> 90\%$) after the initialisation strategy, the initial temperature must not be very high (between 20 and 50), and conversely. In our experiments, an initial temperature of 100 has generally given good results for tasks of average difficulty. The decrease in T was fixed at 10% for each iteration in all our experiments. A non-zero $T_{\min}$ value is numerically required and depends on the computational capacities of the computer used. The algorithm did not really appear to be sensitive to all these internal parameter values. For example, other decreasing rates were tested for the gradient step and temperature without any significant modifications in the results being observed. The values proposed here seemed sufficiently good and did not require fine tuning for the various experiments.

Theoretically, all the prototypes of a class could be eliminated. This would mean that this class could not be distinguished from any other without a decrease in the classification rate over the training set. One conclusion is that the data set had not been constructed well enough (regarding the choice of instances and/or descriptive variables) to realise the classification task by a prototype-based method as stated in Section 1, since for these data spatial proximity does not express membership of the same class. The user can possibly insert into the algorithm a rule fixing a minimum number of prototypes by class. (This was never required in our numerous experiments on real as well as artificial data sets.)

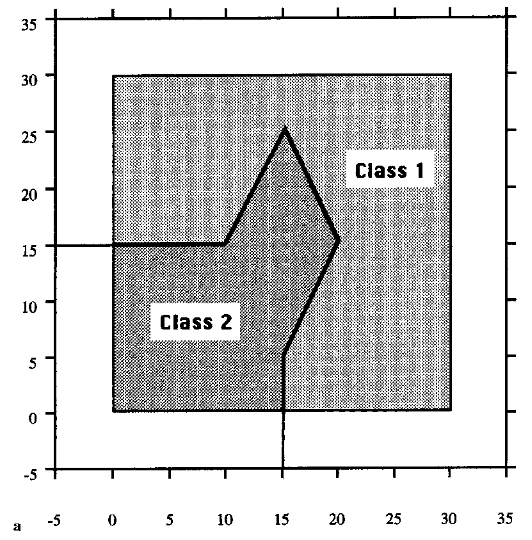
3. EXPERIMENTS

A number of experiments have been conducted on both simulated and real data. The first series concerned the behaviour of the proposed prototype-based classifier (Sections 3.1 and 3.2). The second one concerned the comparisons on performances with other classifiers such as neural networks and symbolic inductive systems (Section 3.3). We do not give experimental results related to the contribution of each component of the algorithm (initialisation, annealing process, etc.). They are detailed elsewhere.(6)

3.1. Problem decomposition and cognitive interpretation

The purpose of this experiment was to illustrate both how the proposed algorithm produces piecewise-linear decision boundaries by breaking a problem down into convex areas localised by the prototypes, and how the prototype locations can be interpreted from a cognitive point of view. As can be seen from the problem presented in Fig. 1, each prototype draws a convex decision surface in the pattern space. This surface results from the nearest neighbour competitive process. The combination of these individual regions constructs a final region for each class. The difference between the real frontier and the decision surface produced by the algorithm results from the lack of training patterns in some areas of the pattern space (only 200 patterns in each class). An extension of the proposed algorithm (see Section 4) will give a solution to this kind of problem.

From a cognitive point of view, what a prototype really is remains an open question (semantically, of course). In case of their use in classification functions, Medin and Barsalou(12) saw prototypes as a special case of reference points with which they associated an interpretation bearing on their location in the instance space. They distinguish three kind of locations: (i) at the boundaries of categories—a reference point entails salient attribute values that occur at boundaries between



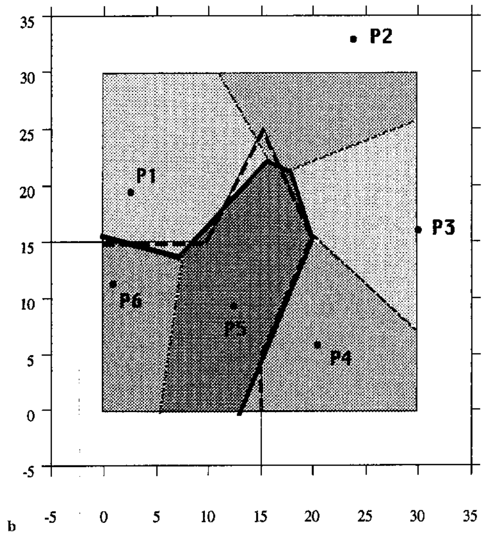


Fig. 1. (a) A two-class problem with a uniform pattern distribution (200 training patterns in each class). (b) Prototypes and surface breakdown produced by the algorithm.

categories; (ii) at the centre of categories—a prototype represents the central tendency of its category; (iii) at ideal locations—a prototype represents an ideal exemplar that may never be observed (e.g. "things to eat on a diet" refers to a zero-calorie food which is ideal, but impossible).

It is surprising to see that the semantic interpretation of the prototypes produced in Fig. 1 fits the view of Medin and Barsalou. Hence P4 and P5 are *centre* prototypes and P2 and P3 are clear cases of *ideal* prototypes (they are located outside the range of possible values) while P1 and P6 are *discriminant reference points* that encode the separation line between instances around value 15 on the vertical axis. It is interesting to see that initial prototypes derived from a k-means and thus are centre prototypes only. The fact that optimised prototypes represent the three types of reference points is a good indication that the process neatly adapts the prototype locations to the nature of the data. This was confirmed by the experimental results on various data sets.(6)

3.2. Identification of optimal prototypes

This experiment tests the capacity of the proposed algorithm to identify the optimal locations of prototypes (when they exist). A three class problem (shown in Fig. 2) was used for this purpose. The examples in each class were artificially generated by Gaussian distributions (several per class). Each Gaussian centre is shown by ▲; each standard deviation is proportional to the distance between the Gaussian centre and the centre of the nearest Gaussian distribution. The examples generated were attributed to the class corresponding to the nearest centre. The decision surfaces therefore appear as shown in Fig. 2 and the centre ▲ can be considered as optimal locations of prototypes to describe each class.

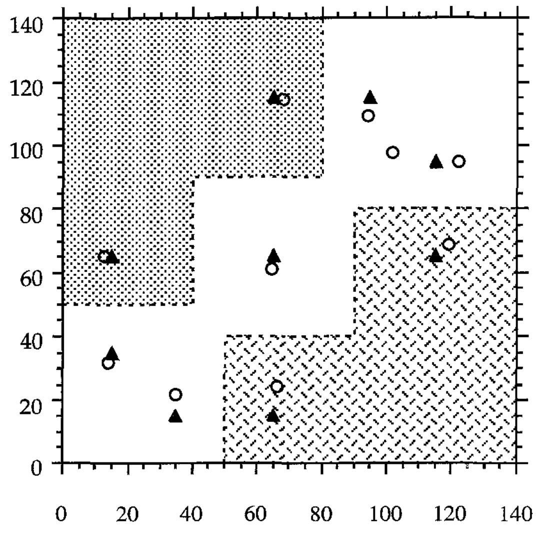


Fig. 2. A three-class problem with the locations of optimal prototypes (▲) and those found (– o –) by the algorithm proposed.

This problem was submitted to the proposed algorithm with eight prototypes per class (a total of 24) to begin the initialisation strategy. Finally the algorithm only retained 10 prototypes (shown by – o – in Fig. 2) distributed near the 9 centres ▲.

3.3. Comparisons with other classifiers

The proposed algorithm was compared to other classifiers from different approaches such as the classical linear (logistic regression), the neural network and the symbolic inductive learning ones (decision tree produced by the C4.5 algorithm of Quinlan(13)) without forgetting the classical k-nearest neighbour algorithm (with $k = 1$). The neural network classifiers include the MLP (with one hidden layer) and the RBF networks (classical RBF and normalised version RBFN, see Decaestecker and Saerens(14)) together with Kohonen's LVQ3 network(4) (another prototype-based algorithm).

These comparisons were conducted on artificial and real data sets with non-linear class boundaries, class overlapping and noise:

- *Iris* is Fisher's well-known set containing three classes of 50 instances each, where a class refers to a type of Iris plant.
- *Diabetes* contains 145 records of three different diagnoses of diabetes based on five numerical variables representing chemical tests.
- *Geometrical Data* is a two-class problem defined in a two-dimensional space. The classes are circumscribed by two circles in a square (see Fig. 3). Class 1 is represented by grey areas and class 2 by a white ring. Instances are uniformly distributed over the whole surface of the square.
- *Waveforms*(15) consist of three classes, each of which is a linear combination of three distinct waveforms. Each instance consists of a vector of 21 continuous values. These data are thus distributed over a large multidimensional space. Without the introduction of noise, this data set is easy to classify, and classifiers generally realise good performances (near 100% of accuracy). In contrast, this classification is known to be a difficult problem when noise is present (in this latter instance the Bayes rule gives a classification error rate of 14%).

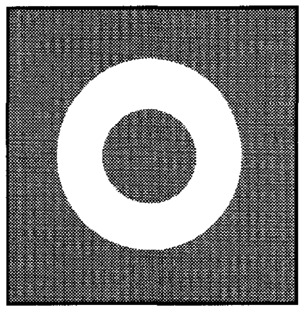


Fig. 3. Geometrical data (two-class problem).

Table 1. Experimental conditions for the four data sets tested

Data	Dim	Classes	Training	Test	Centroids
Iris	4	3	30/75	120/75	9/24
Diabetes	5	3	29/71	116/74	9/24
Geometrical	2	2	115/575	1000/1000	12/24
Waves	21	3	30/300	5000/5000	9/24

Table 2. Comparative results (1–NN, PNN and the best other classifier) on *non-noisy* data

	Small		Large	
Model	Cplx (Train)	Acc% ± Sdt	Cplx (Train)	Acc% ± Sdt
Iris				
1–NN	30.0	94.2±2.39	75.0	94.7±1.65
MLP	10.0 (98.3)	96.8±1.60	10.0 (97.9)	96.0±2.15
PNN	5.3 (99.3)	93.8±3.21	5.4 (98.5)	95.6±2.15
Diab				
1–NN	29.0	95.7±1.78	72.0	97.3±1.45
RBFN	6.5 (97.9)	95.8±2.16	15.0 (98.9)	98.0±1.84
PNN	3.0 (100.0)	98.3±1.16	3.0 (100.0)	99.1±0.87
Geom				
1–NN	115.0	90.3±1.36	575.0	95.5±0.58
RBF	17.7 (96.6)	92.0±2.03	22.4 (95.6)	95.0±2.42
PNN	13.4 (99.7)	90.4±1.72	18.9 (98.7)	95.2±0.65
Waves				
1–NN	30.0	92.5±2.79	300.0	99.4±0.19
MLP	10.0 (99.2)	92.1±3.60	10.0 (99.6)	98.9±0.59
PNN	4.8 (100.0)	94.4±2.48	3.1 (100.0)	99.6±0.17

Each data set was tested with two different amounts of training data (small and "reasonable" in quantity, see Table 1), before and after the introduction of noise. A Gaussian noise on each attribute was added artificially, except for geometrical data for which noise was introduced by overlapping the classes.

For each data set, 10 experiments (or runs) were performed with independent training and test sets (cross-validation-like process). The results given were averaged over the 10 runs (see Tables 2 and 3). Table 1 gives a summary of the experimental conditions relating to the dimension of the pattern space (Dim), the number of classes, the two different sized training and test sets and the corresponding numbers of initial centroids submitted to the initialisation strategy. Our initialisation strategy was also used for the LVQ3, and slightly adapted for the RBF, networks.[14]

In the interest of clarity, Tables 2 and 3 only give the results obtained with the proposed algorithm (labelled PNN for "prototype-based nearest neighbour"), the

Table 3. Comparative results (1–NN, PNN and the best other classifier) on *noisy* data

	Small		Large	
Model	Cplx (Train)	Acc% ± Sdt	Cplx (Train)	Acc% ± Sdt
Iris				
1–NN	30.0	79.3±3.94	75.0	80.7±2.51
MLP	10.0 (86.3)	81.5±1.19	10.0 (83.6)	83.0±1.34
PNN	5.3 (95.0)	82.0±1.76	5.7 (91.2)	83.0±1.13
Diab				
1–NN	29.0	82.1±3.53	72.0	84.1±1.40
RBFN	5.5 (88.6)	83.7±3.74	12.7 (92.3)	87.0±1.39
PNN	3.3 (93.1)	83.9±3.69	4.3 (94.1)	87.7±0.90
Geom				
1–NN	115.0	80.4±2.84	575.0	82.0±0.85
RBF	16.6 (88.4)	82.2±2.79	21.2 (87.0)	85.1±1.47
PNN	11.9 (94.4)	81.6±2.13	15.5 (88.8)	84.4±1.07
Waves				
1–NN	30.0	71.7±2.79	300.0	76.4±0.79
RBFN	6.7 (94.0)	79.4±3.30	20.7 (87.5)	84.9±0.50
PNN	4.8 (100.0)	80.2±2.76	12.7 (96.3)	83.8±0.88

nearest neighbour (1–NN) and the best classifier (in terms of classification accuracy on test sets) selected from among the other ones. The results include the complexity of the model (Cplx, the number of prototypes for the nearest neighbour algorithms and the number of hidden units for the neural networks), the percentage of correct classifications over the training sets given in brackets (Train, which illustrates the network's ability to represent the data set), accuracy (Acc%, the percentage of correct classifications over the *test* sets) and its standard deviation over the 10 runs ($\pm$Sdt).

Note that the best classifier selected in Tables 2 and 3 varies from one problem to another (from MLP to RBF to RBFN). As can been seen, PNN offered performances comparable to those of the best classifier—sometimes slightly inferior, sometimes slightly superior. To be more precise, the set of classifiers tested did not include one which was always better than PNN whatever the experimental conditions (data set, training set size, the absence or presence of noise). In contrast, the complexity of the PNN model (i.e. the average number of prototypes retained) was always smaller than for the model selected. Surprisingly, the low level of complexity of the PNN models did not restrict their abilities to represent the training data sets, as is shown by the performances given in brackets (in Tables 2 and 3). These performances over the training sets were always better for PNN than for the best selected classifier. Compared to 1–NN, the reduced complexity of the PNN did not decrease its ability to generalise (unlike the classical editing methods, see Section 1). Hence PNN performances (over test sets) were generally at least as good as the 1–NN ones in a noise-free context and better in a noisy one.

4. CONCLUSIONS AND FUTURE WORKS

In the context of nearest neighbour classifiers, a lot of work has been carried out since the 70s to improve in the fields of computation and generalisation efficiency.(1) This work is also related to more recent studies in Machine Learning (Instance Based Learning). Besides reducing the number of stored instances, these algorithms also aim to be noise-resistant.(16) The majority of the above methods are based on the selection of a subset of training patterns. In the present work, we propose a prototype-based approach which consists in generating prototypical reference vectors instead of selecting some instances from among those available. This method is based on an optimisation process which mixes a classical gradient descent with a deterministic annealing process. This kind of optimisation process cannot be used in selecting approaches because in this latter instance, the search space is discrete.

The recent work by Yan(5) also followed an optimisation approach by using a two-layer perceptron. However an additive process was required to map the network back to a nearest neighbour classifier. Furthermore, no investigation has ever been made in choosing the number and initial locations of the prototypes. In contrast, one important issue in the present work is the control of the effective complexity of the model (here, the number of prototypes) to ensure good generalisation capacities. One of the most traditional techniques for doing this is cross-validation. It is used to compare a range of models with different number of adaptive parameters. The disadvantage of such an approach is that it requires the optimisation (or training) process to be repeated several times which, under some circumstances, could lead to the need for large amounts of processing time. In the present work, we develop heuristics to avoid this requirement. Firstly, an initialisation strategy generated a reasonable set of initial prototypes. This strategy was biased in order to provide a good classification rate on a training set with a minimum number of prototypes. Secondly, by its deterministic annealing component, the optimisation process was able to prune useless prototypes automatically. Our initialisation strategy can also be used for initialising other prototype-based classifiers for which the initialisation problem was not really taken into account (e.g. Kohonen's LVQ networks(4) or Yan's MLP(5)), and can be easily adapted for RBF networks.(14)

Experiments have shown that the proposed algorithm behaves as expected and satisfies the double objectives of computational efficiency (by drastically reducing the number of reference vectors to store) and generalisation ability (by providing performances comparable to those of the best classifier chosen from among a large set of candidates).

Some extensions can be made to the algorithm presented. Hence, in a recent work,(17) a reject option has been proposed to exclude the (test) cases for which the classification between two classes is too ambiguous. The reject rule is based on an entropy criterion and can easily be adapted to other prototype-based classifiers. Some experiments have shown the efficiency of this approach in relation to near boundary unsafe areas generated by non-linear class boundaries, class overlapping and noise. In these cases, the error rate can be reduced to half with a reasonable number of omissions.

While distance-based classifiers (including nearest neighbour ones and RBF networks) have significant advantages over others (such as MLP, see the study by Kramer and Leonard(18) for example), there is still a critical problem encountered with distance-based classifiers. This problem is related to the presence of irrelevant (or unimportant) variables in the cases where it is impossible to distinguish relevant from irrelevant variables *a priori*. In these cases MLP is efficient because the backpropagation algorithm is able to ignore irrelevant variables. In contrast, when distance-based classifiers have to cover a region artificially inflated by an irrelevant variable (with substantial variance), their required number of prototypes (or reference training patterns or RBF units) may dramatically increase, as may the complexity of the classification task, and consequently the classification error rate produced by the classifiers. To solve this critical problem, different approaches focussing the selection of useful features are investigated in the literature(19,20) and could be adapted for the algorithm proposed. They consist principally of determining appropriate feature weighting during the optimisation process.

REFERENCES

1. B. V. Dasarathy (ed.), *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California (1991).
2. C. L. Chang, Finding prototypes for nearest neighbor classifiers, *IEEE Trans. Comput.* **C-23**, 1179–1184 (1974).
3. D. L. Reilly, L. N. Cooper and C. Elbaum, A neural model for category learning, *Biol. Cybernetics* **45**, 35–41 (1982).
4. T. Kohonen, The self-organizing map, *Proc. IEEE* **9**, 1464–1480 (1990).
5. H. Yan, Prototype optimization for nearest neighbor classifiers using a two-layer perceptron, *Pattern Recognition* **2**, 317–324 (1993).
6. C. Decaestecker, A prototype-based neural net classifier, BENELEARN-94 (Proc. Belgian-Dutch Conference on Machine Learning), Technical Report EUR-CS-94-05, Department of Computer Sciences, Erasmus University Rotterdam, 217–229 (1994).
7. T. Kohonen, J. Kangas, J. Laaksonen and K. Torkkola, LVQ_PAK: A program package for correct application of learning vector quantization algorithms, *Proc. IEEE Int. Joint Conf. on Neural Networks*, Baltimore, I.725–I.730 (1992).
8. T. Van de Merckt, NFDT: A system that learns flexible concepts based on decision trees for numerical attributes, in *Machine Learning (Proceedings of the 9th International Workshop)*, D. Sleeman and P. Edwards, eds, pp. 322–331. Morgan Kaufmann, Los Altos, California (1992).
9. K. Rose, E. Gurewitz and G. Fox, Statistical mechanics and phase transitions in clustering, *Phys. Rev. Lett.* **8**, 945–948 (1990).
10. K. Rose, E. Gurewitz and G. Fox, A deterministic annealing approach to clustering, *Pattern Recognition Lett.* **11**, 589–594 (1990).
11. K. Rose, E. Gurewitz and G. Fox, Deterministic annealing, constrained clustering and optimization, *Proc. IEEE Int. Joint Conf. on Neural Net.*, Singapore, 2515–2520 (1991).
12. D. L. Medin and L. W. Barsalou, Categorization processes and categorical perception, in *Categorical Perception: The Groundwork of Cognition*, S. Harnard, ed., pp. 455–490. Cambridge University Press, Cambridge (1987).
13. J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California (1993).
14. C. Decaestecker and M. Saerens, Comparisons of different RBF networks for pattern classification, *Proc. Int. Conf. on Artificial Neural Networks (ICANN'95)*, Paris, 591–596 (1995).
15. L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*. Wadsworth and Brooks, Pacific Grove, California (1984).
16. D. W. Aha, D. Kibler and K. M. Albert, Instance-based learning algorithms, *Machine Learning* **6**, 37–66 (1991).
17. C. Decaestecker and T. Van de Merckt, How to "secure" the decisions of a NN classifier, *Proc. 1994 IEEE Int. Conf. on Neural Networks*, Orlando, U.S.A., 263–268 (1994).
18. M. A. Kramer and J. A. Leonard, Diagnosis using backpropagation neural networks—Analysis and criticism, *Comput. Chem. Engng.* **14**, 1323–1338 (1990).
19. D. G. Lowe, Similarity metric learning for a variable-kernel classifier, *Neural Comput.* **7**, 72–85 (1995).
20. E. Hartman and J. D. Keeler, Predicting the future: Advantages of semilocal units, *Neural Comput.* **3**, 566–578 (1991).

About the Author — CHRISTINE DECAESTECKER obtained a Master's degree in Mathematics in 1984 from the Université Libre de Bruxelles, where she also obtained a Ph.D. in Pure Sciences in 1991. In 1992, she joined the Institute of Interdisciplinary Research and Development in Artificial Intelligence (IRIDIA, ULB) as a researcher, and is currently taking part in the BELON project (Integrated Research on Quantified Beliefs, Doxatic Logics and Neural Networks). Her current research interests include data analysis, pattern recognition, neural networks and machine learning.