Available online at www.sciencedirect.com

SCIENCE DIRECT®

Pattern Recognition 39 (2006) 209–222

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Prototype reduction schemes applicable for non-stationary data sets[☆]

Sang-Woon Kim[a], B. John Oommen[b,*]

[a]Department of Computer Science and Engineering, Myongji University, Yongin 449-728, Korea
[b]School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6

Received 2 February 2005; received in revised form 27 June 2005; accepted 29 July 2005

Abstract

All of the prototype reduction schemes (PRS) which have been reported in the literature, process *time-invariant* data to yield a subset of prototypes that are useful in nearest-neighbor-like classification. Although these methods have been proven to be powerful, they suffer from a major disadvantage when they are utilized for applications involving non-stationary data, namely, time varying samples, typical of *video* and multimedia applications. In this paper, we suggest two PRS mechanisms which, in turn, are suitable for two *distinct* models of non-stationarity. In the first model, the data points obtained at discrete time steps, are individually assumed to be perturbed in the feature space, because of noise in the measurements or features. As opposed to this, in the second model, we assume that, at discrete time steps, new data points are available, and that these themselves are generated due to a non-stationarity in the *parameters* of the feature space. In both of these cases, rather than process all the data as a whole set using a PRS, we propose that the information gleaned from a previous PRS computation be enhanced to yield the prototypes for the current data set using an LVQ-3 type "fine tuning". The results are, to our knowledge, the first reported PRS results for non-stationary data, and can be summarized as follows: if the system obeys the first model of non-stationarity, the improved accuracy is as high as 90.98% for artificial data "Non_normal 2", and as high as 97.62% for the real-life data set, "Arrhythmia". As opposed to this, if the system obeys the second model of non-stationarity, the improved accuracy is as high as 76.30% for the artificial data, and as high as 97.40% for this real-life data set. These are, in our opinion, very impressive, considering that the data sets are *truly time-varying*.

Keywords: Prototype reduction schemes (PRS); Time varying samples (TVS); Non-stationary environments; Hybrid-type prototype reduction

1. Introduction

1.1. Overview

Over the past five decade numerous families and avenues of statistical pattern recognition (PR) systems have been developed. The general model of computation has been the following: The system is provided with a set of data represented in terms of its features. Using these so-called "training samples", the system builds a classifier which is, for example, of a parametric form, or the non-parametric form. Subsequently, data to be tested is provided, and the quality of the classifier is measured by quantifying the accuracy by which these samples are classified. We emphasize, though, that traditionally, all classifiers assume that the training samples and their underlying distribution are stationary.

In this paper, we suggest two time varying prototype reduction schemes (PRS) mechanisms which can be utilized for applications involving non-stationary data. Such data is typical of video and multimedia applications, in which the objects to be recognized move in space, or change with time. Although we have not specifically demonstrated this (because of the lack of suitable data), we believe that these methods can also be useful in medical diagnosis and

[☆] The work of the first author was done while visiting at Carleton University, Ottawa, Canada. The work of the first author was partially supported by KOSEF, the Korea Science and Engineering Foundation. The work of the second author was partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada. This research was also generously supported by the Korea Research Foundation Grant funded by the Korea Government (MOEHRD-KRF-2005-D00004).

* Corresponding author. Tel.: +1 613 520 4333; fax: +1 613 520 4334.
E-mail addresses: kimsw@mju.ac.kr (S.-W. Kim), oommen@scs.carleton.ca (B.J. Oommen).

bio-metric analysis, in cases where the measurements of the patients/"clients" change because of the variations in their conditions themselves, or because of variations in the equipment taking the measurements.

In particular, we propose two methods to tackle non-stationarity, which, in turn, are suitable for two *distinct* models of non-stationarity. In the first model, we assume that the data points obtained at discrete time steps, are individually perturbed in the feature space, because of *noise in the measurements* or features. In the second model, however, we assume that, at discrete time steps, new data points, which are themselves generated due to a non-stationarity in the *parameters* of the feature space, are available. The solution we advocate is the following: In both of these cases, rather than process all the data as a whole set using a PRS to yield a "time varying set of prototypes", we propose that the information gleaned from a previous PRS computation be enhanced to yield the prototypes for the current data set, and this enhancement is accomplished using an LVQ-3 type "fine tuning". This philosophy, though simple, is effective and quite powerful. Indeed, in cases such as the benchmark "non-normal" data set, we have observed that if the points are perturbed, the accuracy of a PR system which utilizes a fixed set of prototypes is quite unacceptable—it yields only about 50% accuracy. In this case, if the prototypes are continuously updated as per our strategy, the updated prototypes yield an accuracy of almost 75%. Thus, we believe that our methods can be used advantageously for medium and large non-stationary data sets.

The paper is organized as follows: after providing a brief introduction to the state-of-the-art PRSs, we briefly catalogue the contributions of this paper. In Section 2 we explain the difficulties involved in adequately modelling non-stationarity, and then proceed to describe both the models of non-stationarity that we propose. After proposing these models, namely, the so-called noisy measurement non-stationarity, and the noisy parameter non-stationarity in their general setting, we proceed in Section 3, to describe the schema for the proposed solutions, namely the process that will be invoked to *update* the PRSs for both these models. Section 4 details the experimental results for artificial and real-life benchmark data sets, where the specific advantages of our proposed methods are clearly highlighted. Section 5 concludes the paper.

1.2. State-of-the-art prototype reduction schemes

Let $T = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \in \mathscr{R}^p$ be a set of N feature vectors in p dimensions. We assume that T is a labeled data set, so that T can be decomposed into, say, c subsets $\{T_1, \ldots, T_c\}$ such that $T = \bigcup_{k=1}^{c} T_k$, $T_i \cap T_j = \phi$, $\forall i \neq j$. Our goal is to design a classifier with this *training data* set. Specifically, we are interested in classifiers of a nearest-neighbour (NN) or nearest prototype family [1]. Thus, we need one or more *prototypes* (vectors in $\mathscr{R}^p$) that will represent each T_k. The limiting case comprises using *all* of the input vectors as prototypes, but in many cases, this will impose an unacceptable computational burden on the classifier.

In non-parametric pattern classification which use the NN or the k-nearest neighbour (k-NN) rule, each class is described using a set of sample prototypes, and the class of an unknown vector is decided based on the identity of the closest neighbour(s) which are found among all the prototypes [1]. To enhance computation, it is advantageous to reduce the number of training vectors while simultaneously insisting that the classifiers that are built on the reduced design set perform as well, or nearly as well, as the classifiers built on the original data set. Various PRSs, which are useful in NN-like classification, have been reported in the literature—two excellent surveys are found in Refs. [2,3]. Bezdek et al. [3,25], who composed the second and more recent survey of the field, reported that there are "zillions!" of methods for finding prototypes (see Ref. [3, p. 1459]).

Rather than embark on yet another survey of the field, we mention here a *few* representative methods of the "zillions" that have been reported. One of the first of its kind is the condensed NN (CNN) rule [4]. The reduced set produced by the CNN, however, customarily includes "interior" samples, which can be completely eliminated, without altering the performance of the resultant classifier. Accordingly, other methods have been proposed successively, such as the reduced NN (RNN) rule [5], the prototypes for NN (PNN) classifiers [6,26], the selective NN (SNN) rule [7], two modifications of the CNN [8], the edited NN (ENN) rule [9], and the non-parametric data reduction method [10,11]. Besides these, in Ref. [12], the vector quantization (VQ) [18] and the Bootstrap [13] techniques have also been reported as being extremely effective approaches to data reduction. Recently, support vector machines (SVM) [14,24] have proven to possess the capability of extracting vectors that support the boundary between any two classes. Thus, they have been used satisfactorily to represent the global distribution structure.

In designing NN classifiers, however, it seems to be intuitively true that prototypes near the separating boundary between the classes play more important roles than those which are more interior in the feature space. In creating or selecting prototypes, vectors near the boundaries between the classes have to be considered to be more significant, and the created prototypes need to be moved (or adjusted) towards the classification boundaries so as to yield a higher performance. Based on this philosophy,[1] namely that of *selecting* and *adjusting* the reduced prototypes, we recently proposed a new hybrid approach that involved two distinct

[1] The reader will observe that we have not burdened him with unnecessary details of any of the "zillions" of PRSs. Indeed, they can be found in the literature. However, we have taken the pains to explain the latter scheme where we enhance a traditional PRS with an LVQ3-type fine-tuning phase, because we intend to propose an analogous strategy for our current problem.

phases [15,16]. In the first phase, initial prototypes are *selected* or *created* by any of the conventional reduction methods mentioned earlier. After this selection/creation phase, the technique in Refs. [15,16] suggests a second phase in which the proposed reduced prototypes are migrated to their "optimal" positions by *adjusting* them by invoking an LVQ3-type *learning* scheme. The relative advantages of the scheme in Refs. [15,16] have been demonstrated on both artificial and real-life data sets.

All the PRS methods reported in Refs. [2,3], (including the one proposed in Refs. [15,16]) are practical as long as the size of the data set is not "too large". The applicability of these schemes for large-sized data sets is limited because they all suffer from a major disadvantage—they incur an excessive computational burden encountered by processing *all the data points*. It should be noted, however, that points in the interior of the Voronoi space[2] of each class are usually processed for no reason—typically, they do not play any significant role in NN-like classification methods. Indeed, it is not unfair to state that processing the points in the "interior" of the Voronoi space becomes crucial only for "smaller" instantiations of the problem.

Since prototypes near the boundary play more important roles than the interior ones for designing NN classifiers, the points near the boundary are more important in selecting the prototypes. In all the previously reported PRS, however, points in the interior of the Voronoi space are processed for, apparently, no reason. Consequently, most reported PRS suffer from an excessive computational burden encountered by processing all the data, which becomes very prominent in "large" data sets. To overcome this disadvantage, a recursive PRS mechanism was proposed in Ref. [17]. In Ref. [17], the data set is sub-divided recursively into smaller subsets to filter out the "useless" internal points. Subsequently, a conventional PRS processes the smaller subsets of data points that effectively sample the entire space to yield *subsets* of prototypes—one set of prototypes for each subset. The prototypes, which result from each subset, are then coalesced, and processed again by the PRS to yield more refined prototypes. In this manner, prototypes which are in the interior of the Voronoi boundaries, and are thus ineffective in the classification, are eliminated at the subsequent invocations of the PRS. A direct consequence of eliminating the "redundant" samples in the PRS computations, is that the processing time of the PRS is *significantly* reduced.

This overview of the state-of-the-art of PRSs should be sufficient to help us proceed in formulating our solution to the problem at hand.

[2] Typically, the Voronoi hyperplane between two classes is an equi-bisector of the space, partitioning the points of each class on either side. Classification is achieved by assigning a class index to a sample being tested, and in our context, this is done by computing the location of the tested sample in the Voronoi space, for example, by determining the class of *its* NN using any well-established metric.

1.3. Contributions of the paper

The main contribution of this paper is we have proposed two models by which non-stationarity of data can be modeled. The second contribution is that in each of these cases, we have presented a technique by which the prototypes can be computed at any time instant by processing the information in the prototypes available at the *previous* time instant, and augmenting it with the currently available data. Thus, we show that there is a distinct advantage in taking the non-stationarity into consideration. Finally, we have shown that these results lead to superior PR systems where the data is truly non-stationary.

The results presented here are of a preliminary sort, as there is not much benchmark data currently available to describe problems of this sort. But we believe that our methods have potential in real-life problems which deal with multimedia applications, and in domains that reckon with medical and bio-metric data. Although the contribution is modest, to the best of our knowledge, there is currently no reported time varying PRS suitable for handling non-stationarity.

2. Models of non-stationarity

In this section we explain the two different models of non-stationarity that we propose.

Let us suppose that at time "t" the system is presented with a data set $S_i(t)$, which represents the samples of class ω_i as measured at "t". The jth sample in this set is $x_{i,j}(t)$. We propose the following two models for capturing non-stationarity.

2.1. Noisy measurement non-stationarity

In this model of non-stationarity we assume that the sample $x_{i,j}$ is obtained by a noisy perturbation on the sample at time "t". This perturbation can be perceived as the inclusion of some additional noise[3] $\theta_1(t+1)$, and thus we write

$$x_{i,j}(t+1) = x_{i,j}(t) + \theta_1(t+1), \tag{1}$$

$$S_i(t+1) = \bigcup x_{i,j}(t+1). \tag{2}$$

Typically, each data point $x_{i,j}(t+1)$ is in the neighbourhood of $x_{i,j}(t)$ as shown in Fig. 1. We present an example for the two-dimensional data set referred to as "random", which is generated randomly with a uniform distribution, but with irregular decision boundaries [17]. In this case, the points are generated uniformly, and the assignment of the points to the respective classes is achieved by *artificially* assigning them to the region they fall into, as per the manually created "irregular decision boundary". The set of just

[3] $\theta_1(\cdot)$ and $\theta_2(\cdot)$ refer to the noise generation random variables associated with Models 1 and 2, respectively.

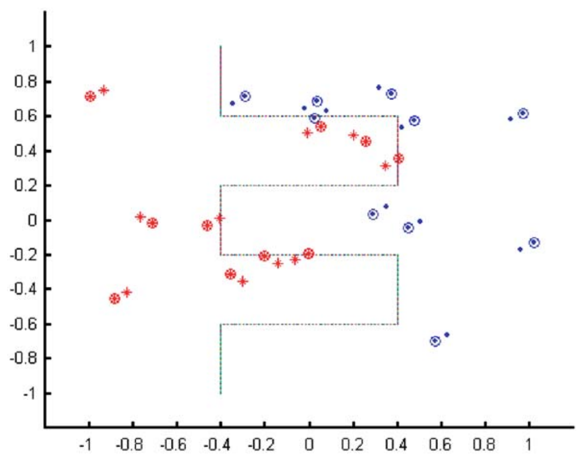

Fig. 1. The set of data points generated with the "noisy measurement model" of non-stationarity, where the $x_{i,j}(t)$, $i = 1, 2$, $j = 1, \ldots, 10$, samples are represented by "$*$" and "$\cdot$", and the $x_{i,j}(t+1)$, $i = 1, 2$, $j = 1, \ldots, 10$, sample vectors are represented by "$\otimes$" and "$\odot$", respectively. Note that the discriminant function also changes with time, but that the set of points bear the semblance to their prior versions.

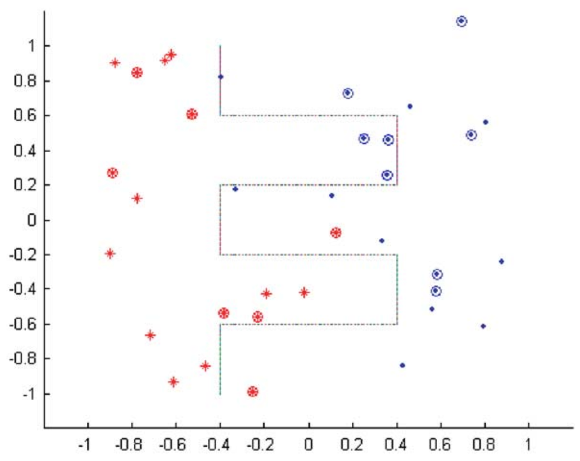

Fig. 2. The set of data points generated with the "noisy parameter model" of non-stationarity, where the $x_{i,j}(t)$, $i = 1, 2$, $j = 1, \ldots, 10$, samples are represented by "$*$" and "$\cdot$", and the $x_{i,j}(t+1)$, $i = 1, 2$, $j = 1, \ldots, 10$, sample vectors are represented by "$\otimes$" and "$\odot$", respectively. Note that the discriminant function also changes with time, but that the set of points bear *no semblance* to their prior versions.

10 sample vectors is generated for the samples of the Class "1" (which are represented by "$*$" in the picture) and the same number of 10 sample vectors for points of Class "2" (which are represented by "$\cdot$" in the picture).

To demonstrate the properties of the mechanism, after generating the $x_{i,j}(t)$, $i = 1, 2$, $j = 1, \ldots, 10$, samples, we again generate "time-varying" points, $x_{i,j}(t+1)$, $i=1, 2$, $j = 1, \ldots, 10$, using the model of non-stationarity, which are represented by "$\otimes$" and "$\odot$", respectively. Observe that in this example, the discriminant function also changes with time, but is essentially a perturbed variation of the discriminant function at the previous time instant.

2.2. *Noisy parameter non-stationarity*

There is a more fundamental model of non-stationarity, in which the data at time "$t+1$" is not just generated from perturbing the data at time "t". Rather, in this second model, the *entire* set at time "$t+1$" is obtained, as it were, by being generated by a random sample generator, whose *parameters* are perturbed versions of the parameters from time "t".

To show how this model works, let us suppose that at time "t" the system is presented with a data set $S_i(t)$, which represents the samples of class ω_i as measured at "t". Again, the jth sample in this set is $x_{i,j}(t)$. In this model of non-stationarity, we assume that the entire sample *set* $S_i(t+1)$ is obtained by a noisy perturbation on the sample *set* $S_i(t)$ at time "t". This perturbation can be perceived as the inclusion of some additional noise, $\theta_2(t+1)$ (which is typically a vector) on the *parameters* of the distribution of $S_i(t)$. Thus we write

$$Parameters[S_i(t+1)] = Parameters[S_i(t)] + \theta_2(t+1). \tag{3}$$

We again present an example for the two-dimensional data set referred to as "random", which is generated randomly with a uniform distribution, but with irregular decision boundaries. Typically, in this case, the data point $x_{i,j}(t+1)$ need not be in the neighbourhood of $x_{i,j}(t)$ as shown in Fig. 2, although the parameters (the mean and the covariance) at time "$t+1$" are perturbed versions of the same parameters at time "t".

3. Schema for the proposed solutions

As mentioned earlier, in this paper we deal with the non-parametric model of PR. Thus, we seek to attain to the classification by a NN-like decision rule. In this setting, the most naive method to handle non-stationarity is to not consider any variation at all.[4] In this case, the prototypes[5] obtained from $S_i(0)$ are used to achieve the classification at all future time instants. Thus, if $P_i(0)$ is the set of prototypes representing class ω_i, the classification is achieved by a NN-like decision rule involving $P_i(0)$, for all i, and for all t.

A more sophisticated method to handle non-stationarity would be to update $P_i(t)$ from the samples of the classes currently available. Thus, when the new set of data $S_i(t+1)$ is available, the system would invoke a PRS afresh for the new data, and effectively treat the problem as a brand new pattern

[4] Pattern recognition systems which use traditional PRS methods would, typically, resort to such a philosophy.

[5] Throughout this section, we assume that the user has access to any of the previously mentioned PRSs. Thus, s/he may chose to use the CNN, the PNN, the HYB or any "pet" PRS scheme from the ones mentioned earlier to yield his/her current prototypes. Our intention is to enhance the prototypes obtained by invoking *this* PRS, by incorporating the information in $S(t+1)$.

classification problem. Thus, if $P_i(t)$ is the set of prototypes representing class ω_i, at time t, at time $t+1$ the classification is achieved by a NN-like decision rule involving $P_i(t+1)$, for all i, where the latter is obtained by performing a PRS on $\bigcup S_i(t+1)$. We believe that this is an expensive strategy. Furthermore, we believe that any "intelligent" system must be capable of utilizing the information in $P_i(t)$ in its effort to compute $P_i(t+1)$. This is what we attempt to accomplish here by invoking an LVQ3-type [19–23] scheme.

In LVQ3, two code-book vectors m_i and m_j, which are the two NN to x, are simultaneously updated, where x and m_j belong to the same class, and x and m_i belong to different classes. Moreover, x must fall into a zone of values called the "window", which is defined around the mid-plane of m_i and m_j. Assume that d_i and d_j are the Euclidean distances of x from m_i and m_j, respectively. Then x is defined to fall in a window of relative width w if $\min(di/dj, dj/di) > ((1-w)/(1+w))$.

The updating rules for m_i and m_j ensure that the code-book vectors continue to approximate the respective class distributions and simultaneously enhance the quality of the classification boundary. These rules are given in Ref. [15] (see Eqs. (5) and (6) on p. 1087 of Ref. [15]), and are not included here to avoid repetition. In these equations, t is the discretized (synchronized) time index, and $\alpha(t)$ and $\varepsilon(t)$ are called the *learning rate* and *relative learning rate*, respectively. This is explained presently.

The accuracy achievable in any classification task to which the LVQ3 is applied, and the time needed for learning, depend on the following factors listed below:

- An approximately-optimal number of code-book vectors assigned to each class and their initial values.
- The initialization of the code-book vectors.
- Setting the parameters, namely the learning rate, the relative learning rate, and the number of iteration steps.

These concepts are essentially common for all LVQ3-based PRS strategies. The rules used for obtaining these are explained in detail in Ref. [15] (see Ref. [15, p. 1088]) and are omitted here in the interest of brevity.

3.1. Updating PRS for the noisy measurement non-stationarity

For the noisy measurement non-stationarity, we propose that the new prototypes at time $t+1$ be obtained by fine-tuning the prototypes $\{P_i(t+1)\}$ by invoking an LVQ-3 type enhancement on $\{P_i(t)\}$. This fine-tuning is achieved by moving the prototypes in the feature space without invoking a PRS on $S_i(t+1)$. Thus, the set $S_i(t+1)$ is directly presented to the existing prototypes, $P_i(t)$, and they, in turn, are migrated so as to optimize the recognition accuracy of the testing samples. Fig. 3 shows the key idea of the proposed mechanism for this model, and the formal algorithm follows.

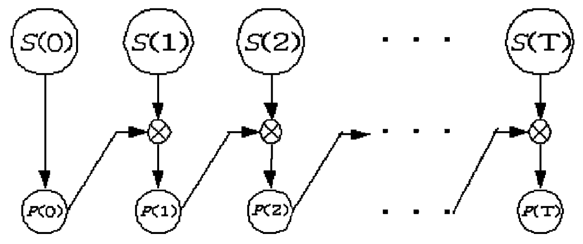

Fig. 3. The processing diagram of the proposed time-variant PRS method. Here, the $S(1), S(2), \ldots, S(T)$ are the subsets of time varying samples, which are generated as mentioned previously from the data sets, $S(0), S(1), \ldots, S(T-1)$, respectively, and the $P(1), P(2), \ldots, P(T)$ are the prototype vectors obtained from invoking the *adjusting* process. The adjusting is performed as $P(t+1) \leftarrow P(t) \bigotimes S(t+1), (t=0, 1, \ldots, T-1)$, which means that the prototype subset $P(t)$ was adjusted with the time-variant data set measured at time $t+1$, $S(t+1)$, using an LVQ3-type algorithm after selecting initial prototypes, $P(0)$, from a given data set, $S(0)$, by using a conventional PRS method.

3.1.1. Determining the relevant LVQ3 parameters

The algorithm[6] that we propose consists of two steps. We first *select* or *create* initial prototypes by any one of the conventional reduction methods described earlier. After this selection/creation phase, we invoke a phase in which the optimal positions are learned with an LVQ3-type scheme. To achieve this, we assume that for every class, i and for the time $t+1$, we are given the new observations $S_i(t+1)$, which, in turn, can be partitioned into two subsets, the *training* set, $S_{i,T}(t+1)$, and *validation* set, $S_{i,V}(t+1)$ at time $t+1$. Our aim is to update the prototypes $P_i(t+1)$, utilizing the information in $P_i(t)$, and the set $S_i(t+1)$.

We first partition the *training* set, $S_{i,T}(t+1)$, into two subsets, called the *placement* set, $S_{i,P}(t+1)$, and the *optimizing* set, $S_{i,O}(t+1)$, where, $S_{i,T}(t+1) = S_{i,P}(t+1) \cup S_{i,O}(t+1)$. The intention is that the *placement* set is used to *position* the condensed prototypes using the LVQ3-type algorithm, and the parameters of the LVQ3-type algorithm are, in turn, optimized by testing the classification efficiency of the *current* placement on the optimizing set, $S_{i,O}(t+1)$. Thus, the training set plays a triple role: (a) first of all, it is used to obtain the initial condensed vectors; (b) secondly, one portion of this set is used by the LVQ3-type algorithm to migrate the condensed vectors; (c) finally, the other portion of the training set serves the purpose of "pseudo-testing", so as to obtain the best parameters for the LVQ3-type algorithm. Using these sets[7] the procedure is formalized as below for each class.

[6] The reader will observe that the actual LVQ3 algorithms are quite similar to the ones described in Ref. [15]. However, the specific sets, namely the *placement* and the *optimizing* sets are quite different, and so we believe that it is necessary to specify what they are (in this non-stationary setting) so that other researchers can effectively use our new methods.

[7] Specific distinct indices j and i are used just for ease of notation. The training sets are first specified in terms of the index j, but then the placement and optimizing sets are used for every class i.

1. For every class, j, set the initial prototype set $P_{j,Test}(t+1)$ to be the prototypes obtained at time t, using any one of the PRS methods described earlier, and the entire training sets, $S_{i,T}(t+1)$.
2. Set $P_{Test}(t+1) = \bigcup P_{j,Test}(t+1)$, which is the set of the training samples of all the classes.
3. Using $P_{Test}(t+1)$ as the set of condensed prototype vectors, do the following using the *placement* sets, $S_{i,P}(t+1)$, and the optimizing sets, $S_{i,O}(t+1)$ for all the classes:

   (a) perform LVQ3 using the points in the *placement* set, $S_{i,P}(t+1)$. The parameters of the LVQ3 are spanned by considering increasing values of w from 0.0 to 0.5, in steps of Δw. The sets $P_{j,Test}(t+1)$ (for all j) and $P_{Test}(t+1)$ are updated in the process. Select the best value w_0 after evaluating the accuracy of the classification rule on $S_{i,O}(t+1)$, where the NN-classification is achieved by the adjusted $P_{Test}(t+1)$;
   (b) perform LVQ3 using the points in the *placement* set, $S_{i,P}(t+1)$. The parameters of the LVQ3 are spanned by considering increasing values of ε from 0.0 to 0.5, in steps of $\Delta\varepsilon$. The sets $P_{j,Test}(t+1)$ (for all j) and $P_{Test}(t+1)$ are updated in the process. Select the best value ε_0 after evaluating the accuracy of the classification rule on $S_{i,O}(t+1)$, where the NN-classification is achieved by the adjusted $P_{Test}(t+1)$;
   (c) repeat the above steps with the current w_0 and ε_0, till the best values w^* and ε^* are obtained.
4. Determine the best prototype set $P_{Final}(t+1)$ by invoking the LVQ3 process η times with the data in $S_{i,P}(t+1)$, and where the parameters are w^* and ε^*, and where the "pseudo-testing" is achieved by using the optimizing set, $S_{i,O}(t+1)$.

The actual classification accuracy is obtained by testing the classifier using the final prototype set, $P_{Final}(t+1)$, and the original testing (validation) data points, $S_{i,V}(t+1)$.

From Fig. 3, we can see that the prototype vectors of the various time varying samples are obtained from adjusting the previous version instead of extracting them from the current time varying samples. That is, the $S(1), S(2), \ldots, S(T)$ are the subsets of time varying samples, which are generated as mentioned previously from the data sets, $S(0), S(1), \ldots, S(T-1)$, respectively, and the $P(1), P(2), \ldots, P(T)$ are the prototype vectors obtained from invoking the *adjusting* process. The adjusting is performed as $P(t+1) \leftarrow P(t) \bigotimes S(t+1), (t=0, 1, \ldots, T-1)$, which means that the prototype subset $P(t)$ was adjusted with the time-variant data set measured at time $t+1$, $S(t+1)$, using an LVQ3-type algorithm after selecting initial prototypes, $P(0)$, from a given data set, $S(0)$, by using a conventional PRS method.

In Fig. 3, the *adjusting* processes are performed as in Fig. 4, where the "$\circ$" and "$\times$" are prototype vectors of class i and class j, respectively. The prototypes adjusted by using an LVQ3-type algorithm are represented by "$\odot$" and "$\otimes$", respectively. In the processing, the interior vectors of the Voronoi spaces will be relatively ineffective in the classification, (they are represented by the large region described with "*X*" in the regions associated with both classes in the figure), and will thus be eliminated from yielding more refined prototypes.

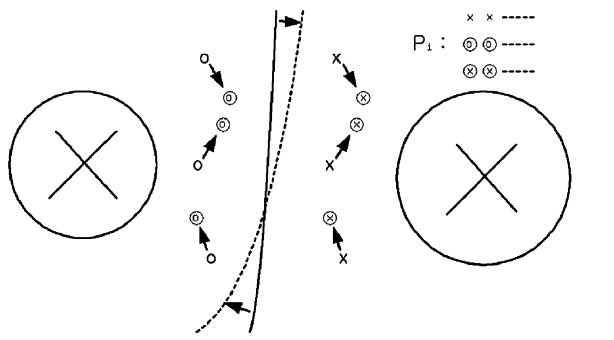

Fig. 4. The key idea of the adjusting mechanism. Here, the "$\circ$" and "$\times$" are prototype vectors of class i and class j, respectively. Then, the adjusted prototypes using an LVQ3-type algorithm are represented by "$\odot$" and "$\otimes$", respectively. The time varying samples which are in the interior of the Voronoi spaces and thus ineffective in the classification, (they are represented with X at the both classes in the figure), were eliminated from yielding more refined prototypes.

3.2. Updating PRS for the noisy parameter non-stationarity

For updating PRS for the noisy parameter non-stationarity, we have two methods as follows:

1. Using the same PRS method applied to the noisy measurement non-stationarity, which was described in Fig. 3.
2. Using a new PRS method in which the prototypes are obtained from adjusting the previous prototypes, $P(t)$, with the current ones, $P(t+1)$, not with $S(t+1)$.

In the second method itemized above, the new prototypes at time $t+1$ are obtained by two steps: first, we obtain crude values of $\{P_i(t+1)\}$ for every class i, and then we fine-tune these crude prototypes $\{P_i(t+1)\}$ by invoking an LVQ-3 type enhancement on $\{P_i(t)\}$. This fine-tuning is done by moving the prototypes at time t in the feature space by using the information found in the crude values, $P_i(t+1)$ and not the entire set $S_i(t+1)$. Thus, the crude set $P_i(t+1)$ is directly presented to the existing prototypes, $P_i(t)$, and they, in turn, are migrated so as to optimize the recognition accuracy of the testing samples.

The key idea of the proposed mechanism for this model (Model 2), and the formal algorithm are quite parallel to the one given above for Model 1. Rather than duplicate the various steps we highlight the primary difference:

- At every step of the algorithm, the fine-tuning is achieved by invoking an LVQ3 scheme, and moving the prototypes at time t in the feature space by using the information found in the crude values, $P_i(t+1)$, rather than invoking the LVQ3 scheme, and moving the prototypes at time t (in the feature space) by using the information found in the entire set $S_i(t+1)$.

As in the case of Model 1, the actual classification accuracy is obtained by testing the classifier using the final prototype set, $P_{Final}(t+1)$, and the original testing (validation) data points, $S_{i,V}(t+1)$.

4. Experimental results

4.1. Experimental data

The *time-varying* PRS has been tested fairly extensively, and compared with many conventional PRS. This was done by performing experiments on a number of "medium-sized" data sets, both real and artificial, as summarized in Table 1. The time-varying data sets consist of *four* subsets which have been generated randomly with two kinds of non-stationary data models, *Models* 1 and 2.

The data set named "Non_normal", which has been also employed in Refs. [11–13] as a benchmark experimental data set, first of all, was generated from a mixture of four 8-dimensional Gaussian distributions as follows:

1. $p_1(x) = \frac{1}{2}N(\mu_{11}, I_8) + \frac{1}{2}N(\mu_{12}, I_8)$ and
2. $p_2(x) = \frac{1}{2}N(\mu_{21}, I_8) + \frac{1}{2}N(\mu_{22}, I_8)$,

where $\mu_{11} = [0, 0, \ldots, 0]$, $\mu_{12} = [6.58, 0, \ldots, 0]$, $\mu_{21} = [3.29, 0, \ldots, 0]$ and $\mu_{22} = [9.87, 0, \ldots, 0]$. In these expressions, I_8 is the eight-dimensional *identity* matrix.

The data sets "Non_normal 2" and "Non_normal 3" were generated randomly with the normal distribution. However, the data sets "Arrhythmia" and "Adult4", which are real benchmark data sets, are cited from the UCI Machine Learning Repository [27].

The "Arrhythmia" data set contains 279 attributes, 206 of which are real-valued and the rest are nominal. In our experiments, the nominal features were replaced by zeros. The aim of the pattern recognition exercise was to distinguish between the presence or absence of cardiac arrhythmia, and to classify the feature into one of the 16 groups. In our case, in the interest of uniformity, we merely attempted to classify the total instances into two categories, namely, "normal" and "abnormal".

The "Adult4" data set was extracted from a census bureau database [27]. The aim of the PR task here is to separate people by incomes into two groups; in the first group the salary is more than 50K dollars, and in the second group the salary is less than or equal to 50K dollars. Each sample vector has 14 attributes. Some of the attributes, such as the age, hours-per-week, etc., are continuous numerical values. The others, such as education, race, etc., are nominal symbols. In this case, the total number of sample vectors is 33,330. Due to time considerations, we randomly selected 8336 samples—approximately 25% of the set.

Before we can test the algorithms, we are faced with a fundamental problem, namely that of obtaining data sets which can be used for testing. As far as we know, such time-varying data sets are not available (even the standard benchmark data sets merely specify time-invariant data). Thus, we have opted to modify the existing artificial and real-life benchmark data sets so that they indeed demonstrate time-varying phenomena. Observe that to achieve this we have to make some assumptions, namely those which specify how the non-stationarity is present. Due to the lack of any better mechanism,[8] we have chosen to artificially generate them from their non-stationary counterparts using the two models explained in Section 2.

Thus, the above artificial and real-life benchmark data sets are used to generate their four *time-varying* data sets, $S(t)$, $t=1, \ldots, 4$, using the two kinds of non-stationary data models as follows. In the noisy measurement non-stationarity (which is referred to Model '1' in the experimental results of next sections), their four *time-varying* data sets are generated as follows: At first, we calculate the variance, *Var*, of a given data set, $S(0)$, of Table 1. Then, we randomly generate its first *time-varying* data set, $S(1)$ as: $S(1) \leftarrow S(0) + Var * (1 + r\ and)$; Here, the function *rand* is to generate an array of random numbers whose elements are normally distributed with mean 0 and variance 1. Finally, we repeat the above procedures to generate *time-varying* data sets $S(t+1)$ from $S(t)$, $t = 1, \ldots, 3$. The *time-varying* data sets of "Model 1" for the other data sets of Table 1, are also generated with the same process.

In the noisy parameter non-stationarity (which is referred to as Model "2" in the experimental results), the four *time-varying* data sets are generated as follows: at first, for every class i, we calculate the mean and standard deviation, μ_i and σ_i, of a given data set, $S(0)$, of Table 1. Then, we randomly generate its first *time-varying* data set, $S(1)$ as: $\mu_i \leftarrow \mu_i * (1 + rand)$; $\sigma_i \leftarrow \sigma_i * (1 + rand)$; $S(1) \leftarrow$

Table 1
The artificial and real-life benchmark data sets using which the four *time-varying* data sets, $S(t)$, $t=1, \ldots, 4$, are generated randomly with the two kinds of non-stationary data models. For every class, i and the time t, the vectors are divided into two subsets of equal size, $S_{i,T}(t)$ and $S_{i,V}(t)$, and used for training and validation, alternatively

Data set types	Data set names	Pattern no. of the given data set, $S(0)$	No. of features	No. of classes
Artificial	Non_normal 2	1000 (500; 500)	8	2
	Non_normal 3	10000 (5000; 5000)	8	2
Real-life	Arrhythmia	452 (226; 226)	279	16
	Adult4	8336 (4168; 4168)	14	2

[8] We are open to input from the community as to how we can, in the absence of real-life data, obtain more "real-life-like" data.

$Normal(\mu_i, \sigma_i)$, where, the *Normal* is a function to generate normal (Gaussian) random numbers with the mean μ_i and the standard deviation σ_i. Finally, we repeat the above procedures to generate *time-varying* data sets $S(t+1)$ from $S(t)$, $t = 1, \ldots, 3$. The *time-varying* data sets of "Model 2" for the other data sets of Table 1, are also generated with the same process mentioned above.

We now highlight some of the real-time properties of the data sets. In the case of Model 1, it turns out that the variations of the generated data points are not so significant. They are generally speaking, in the neighborhood of the original points. The cardinality of the sets is also the same—for each original point we can generate a new point using this model. As opposed to this, in the case of Model 2, as explained earlier, the variations of the generated data points can be quite significant. The new points generated need not be in the neighborhood of the original points because the non-stationary variation occurs at the "parameter" level. Furthermore, although the cardinality of the generated set can be different, in the interest of uniformity, we have kept it to be the same. Thus, for each original point we have generated a new non-stationary point at the next time instant.

In the above data sets, all of the vectors were normalized using their standard deviations. Also, for every class i, the data set for the class was randomly split into two subsets, $S_{i,T}(t)$ and $S_{i,V}(t)$ of equal size. One of them was used for choosing initial code-book vectors and training the classifiers as explained earlier, and the other subset was used in the validation (or testing) of the classifiers. The roles of these sets were later interchanged.

In this case, we employed just three PRSs, namely the CNN,[9] PNN and HYB[10] so as to evaluate the strengths and weaknesses of the current algorithms.

4.1.1. Experimental results: the LVQ3 parameters

In this set of experiments, the $P(t+1)$ is obtained by adjusting $P(t)$ using an LVQ3-type algorithm with $S(t+1)$, where the LVQ3 parameters such as w, ε, α and η play an important role in the classification task. So, the optimal or near-optimal parameters of w^*, ε^*, α^* and η^* should be selected with $S_P(t)$ and evaluated with $S_O(t)$. However, in this experiment, we set $S_O(t)$ with $S_P(t)$ for simplicity, which is a simple so-called, a *re-substitution* evaluation method.

As mentioned earlier, in the algorithm mentioned previously, the relevant optimal parameters were determined by repeating the "pseudo-testing" increasing values of w from 0.0 to 1.0 in steps of Δw, ε from 0.0 to 0.5 in steps of $\Delta\varepsilon$, α from 0.0 to 0.5 in steps of $\Delta\alpha$, η from 100 to 1000 in steps of $\Delta\eta$, till the best values are obtained. However, this process is a time-consumption task. So, in this experiment, we *heuristically* selected the best values of the parameters by adjusting the Δw, $\Delta\varepsilon$, $\Delta\alpha$ and setting $\eta = 100$.

The LVQ3 parameters employed for the *proposed* PRS algorithm for the *four* data sets, namely, "Non_n2", "Non_n3", "Arrhy" and "Adult4", are shown in Table 2, and are included here *to enable repeatability*.

4.2. Experimental results: the classification accuracy

We report below the classification accuracy rates of the *proposed* PRS algorithm for some "medium-sized" time varying data sets. The experimental results of the CNN, PNN and HYB methods for the "Non_normal 2", "Non_normal 3", "Arrhythmia" and "Adult4" data sets are shown in Tables 3, 4, 5 and 6, respectively. In these tables, the abbreviations *Ex*1, *Ex*2 and *Ex*3 are evaluation methods for the classification accuracy employed in this experimentation. In *Ex*1, the prototypes $P(0)$ of the source data set $S(0)$, are constantly used as the prototypes of the four "non-stationary" data sets, $S(t)$, $t = 1, \ldots, 4$. Then, in *Ex*2, the prototypes $P(t)$ at time t are extracted directly from the corresponding data set $S(t)$ using a conventional PRS algorithm. Finally, *Ex*3 is obtained using the "*time-varying*" PRS algorithms proposed in this paper.

First of all, consider the results obtained for the "artificial" data. Tables 3 and 4 show the experimental results of the conventional PRSs and the proposed PRSs for the time-varying artificial data sets, namely, "Non_normal 2" (in short, "Non_n2") and "Non_normal 3" (in short, "Non_n3"), respectively. In Table 3, the first column, '1' and '2', is the results for the "noisy measurement model" and the "noisy parameter model", respectively. Also, the values written in the first row for each PRS algorithm, that is, $P(t)$, are the numbers of prototype vectors extracted from the data sets of $S_T(t)$ and $S_V(t)$, $t = 0, \ldots, 4$, respectively.

Consider the CNN method for the "1" model of the "Non_n2" data set. The training and the testing sets and their *four* non-stationary subsets were processed individually as a separate set using the CNN method. First of all, the training 500 samples and the testing 500 samples, namely, $S_T(0)$ and $S_V(0)$, were reduced into 64 and 66, respectively. Also the first non-stationary data sets, $S_T(1)$ and $S_V(1)$, were truncated into 64 and 83, respectively. By the same way, the prototypes of the other non-stationary subsets, namely, $S_T(t)$ and $S_V(t)$, $t = 2, 3, 4$, are 73, 107 ($t = 2$), 105, 112 ($t = 3$) and 103, 118 ($t = 4$), respectively.

[9] It appears from the literature that the CNN method by Hart is not the *best* competitor for prototype selection in terms of both accuracy *and* effectiveness. We have chosen this method over the methods surveyed in Refs. [2,3] because of its relative simplicity and ease of implementation.

[10] Here, we employed SVM as a *pre-processing* PRS of the HYB method. As is well known, the SVM does reduce the set of prototypes, but not for the NN method. This means that the set of prototypes, which are also the *support vectors* obtained through the SVM method could be absolutely useless with 1-NN. All the other methods considered in this paper are supposed to select a reference set suitable for the 1-NN method. Thus, from this perspective, SVM belongs to a completely different group! Thus, although it is, in one sense, inappropriate for testing it as a basic PRS method, it has advantages if it is used recursively. This is the rationale for including it in our test suite.

Table 2
The LVQ3 parameters employed for the *proposed* PRS algorithm for the "Non_normal 2" (in short "Non_n2"), "Non_normal 3" (in short "Non_n3"), "Arrhythmia" (in short "Arrhy") and "Adult4" data sets

# of models	Data sets	PRS methods	*Values* of LVQ3 parameters			
			w_T, w_V	ε_T, ε_V	α_T, α_V	η_T, η_V
1	Non_n2	CNN	0.9, 0.9	0.10, 0.15	0.40, 0.20	100, 100
		PNN	0.9, 0.8	0.10, 0.10	0.35, 0.50	100, 100
		HYB	0.9, 0.8	0.10, 0.10	0.50, 0.45	100, 100
	Non_n3	CNN	0.9, 0.9	0.05, 0.05	0.05, 0.05	100, 100
		PNN	0.9, 0.9	0.10, 0.10	0.50, 0.35	100, 100
		HYB	1.0, 1.0	0.10, 0.15	0.20, 0.10	100, 100
	Arrhy	CNN	0.1, 0.1	0.05, 0.05	0.05, 0.05	100, 100
		PNN	0.1, 0.1	0.05, 0.05	0.05, 0.05	100, 100
		HYB	0.1, 0.1	0.05, 0.05	0.05, 0.05	100, 100
	Adult4	CNN	0.8, 0.9	0.05, 0.35	0.05, 0.05	100, 100
		PNN	0.8, 0.9	0.05, 0.10	0.10, 0.10	100, 100
		HYB	1.0, 1.0	0.15, 0.05	0.05, 0.10	100, 100
2	Non_n2	CNN	0.9, 0.9	0.15, 0.10	0.05, 0.35	100, 100
		PNN	0.5, 0.9	0.05, 0.05	0.45, 0.45	100, 100
		HYB	0.9, 0.9	0.50, 0.50	0.50, 0.50	100, 100
	Non_n3	CNN	0.9, 0.9	0.05, 0.05	0.15, 0.25	100, 100
		PNN	0.9, 0.9	0.10, 0.10	0.50, 0.50	100, 100
		HYB	1.0, 1.0	0.05, 0.15	0.45, 0.30	100, 100
	Arrhy	CNN	0.1, 0.1	0.05, 0.05	0.05, 0.05	100, 100
		PNN	0.1, 0.1	0.05, 0.05	0.05, 0.05	100, 100
		HYB	0.1, 0.1	0.05, 0.05	0.05, 0.05	100, 100
	Adult4	CNN	0.9, 0.9	0.10, 0.15	0.15, 0.10	100, 100
		PNN	1.0, 0.9	0.10, 0.15	0.15, 0.15	100, 100
		HYB	0.8, 0.8	0.20, 0.20	0.15, 0.05	100, 100

Here, w_T, w_V, ε_T, ε_V, α_T, α_V and η_T, η_V are the window widths, the relative learning rates, the learning rates and the number of iterations of training and validation data sets, respectively. Each value has been selected with a *heuristic* method in which Δw, $\Delta\varepsilon$, $\Delta\alpha$ have been adjusted heuristically after fixing the η as 100.

The resulting classification accuracies, *Acc*'s, have been evaluated with the three methods: *Ex*1, *Ex*2 and *Ex*3. In *Ex*1, the prototypes of the training and the testing samples, $P_T(0)$ and $P_V(0)$, are used as the prototypes of all of the non-stationary data sets, $S_V(t)$ and $S_T(t)$, $t=0,\dots,4$. First of all, the classification accuracy rates of $P_T(0)$ and $P_V(0)$ to $S_V(0)$ and $S_T(0)$ are 92.60% and 91.20%, respectively. Also, the classification accuracy rates of $P_T(1)$ $(=P_T(0))$ and $P_V(1)$ $(=P_V(0))$ to $S_V(1)$ and $S_T(1)$ are 90.20% and 90.20%, respectively. By the same way, the other accuracies of $P_T(t)$ $(=P_T(0))$ and $P_V(t)$ $(=P_V(0))$, $t=2,3,4$, to $S_V(t)$ and $S_T(t)$, $t=2,3,4$, are 86.00, 84.20% $(t=2)$, 82.80, 78.20% $(t=3)$ and 77.00, 74.20% $(t=4)$, respectively.

In *Ex*2, the prototypes of each training and testing data set, namely, $P_T(t)$ and $P_V(t)$, $t=0,\dots,4$, are extracted directly from the corresponding non-stationary data sets, $S_T(t)$ and $S_V(t)$, $t=0,\dots,4$. After that, the classification accuracies of the $P_T(t)$ and $P_V(t)$ are evaluated by $S_V(t)$ and $S_T(t)$, respectively. From Table 3, the classification accuracies of $P_T(t)$ and $P_V(t)$ to $S_V(t)$ and $S_T(t)$ are 92.60, 91.20% $(t=0)$, 91.60, 90.20% $(t=1)$, 90.00, 89.40% $(t=2)$, 89.00, 88.80% $(t=3)$ and 88.40, 88.80% $(t=4)$, respectively.

In *Ex*3,[11] the prototypes of the non-stationary data sets are generated by adjusting *existing* prototype vectors using an LVQ3-type algorithm. As explained previously, instead of extracting individually from the corresponding non-stationary data sets, the prototypes of each training and testing data set, namely, $P_T(t)$ and $P_V(t)$, $t=1,\dots,4$, are generated by adjusting the $P_T(t-1)$ and $P_V(t-1)$, $t=1,\dots,4$, with the $S_T(t)$ and $S_V(t)$, $t=1,\dots,4$, using the LVQ3-type algorithm. Then, the classification accuracies of the $P_T(t)$, $P_V(t)$, $t=1,\dots,4$, are evaluated by $S_V(t)$, $S_T(t)$, $t=1,\dots,4$, respectively. From Table 3, the *Acc*'s of $P_T(t)$ and $P_V(t)$ to $S_V(t)$ and $S_T(t)$ are 93.00, 93.00% $(t=1)$, 92.40, 90.40% $(t=2)$, 90.40, 90.60% $(t=3)$ and 89.00, 89.00% $(t=4)$, respectively.

Finally, the averaged classification accuracies, $\overline{Acc}$, of the *Ex*1, *Ex*2 and *Ex*3, which have been calculated from the *eight Acc*'s of the corresponding non-stationary data sets, are 82.85, 89.53 and 90.98%, respectively.

To highlight the advantage, we also considered the results for "real-life" data sets, namely, "Arrhythmia" and "Adult4" as shown in Tables 5 and 6. From the results of the tables,

[11] This method is the one advocated in this paper.

Table 3
The classification accuracy rates (%) of the time-varying PRSs for the artificial data set, "Non_normal 2"

# of model	PRS	Evalu. method	*Acc* of *original* data	*Acc* of *non-stationary* data sets				$\overline{Acc}$
				$S_T(1), S_V(1)$	$S_T(2), S_V(2)$	$S_T(3), S_V(3)$	$S_T(4), S_V(4)$	
1	CNN	$P(t)$	64, 66	64, 83	73, 107	105, 112	103, 118	
		$Ex1$	92.60, 91.20	90.20, 90.20	86.00, 84.20	82.80, 78.20	77.00, 74.20	82.85
		$Ex2$	92.60, 91.20	91.60, 90.20	90.00, 89.40	89.00, 88.80	88.40, 88.80	89.53
		$Ex3$	92.60, 91.20	93.00, 93.00	92.40, 90.40	90.40, 90.60	89.00, 89.00	90.98
	PNN	$P(t)$	53, 55	55, 68	64, 98	87, 104	99, 103	
		$Ex1$	91.20, 91.20	89.80, 88.40	86.00, 83.40	83.20, 77.60	76.60, 74.20	82.40
		$Ex2$	93.60, 91.20	90.20, 91.00	88.80, 89.00	88.60, 87.80	85.80, 88.40	88.70
		$Ex3$	91.20, 91.20	93.60, 94.00	91.60, 92.80	89.40, 90.40	88.00, 89.40	91.15
	HYB	$P(t)$	254, 256	258, 266	260, 270	270, 272	272, 278	
		$Ex1$	93.00, 91.40	89.40, 90.40	85.40, 85.40	82.20, 78.40	76.60, 74.80	82.83
		$Ex2$	93.00, 91.40	91.60, 91.20	71.80, 89.60	66.00, 89.40	88.60, 76.40	83.08
		$Ex3$	93.00, 91.40	90.00, 90.80	88.00, 89.60	86.00, 83.60	81.80, 81.00	86.35
2	CNN	$P(t)$	64, 66	169, 150	236, 217	246, 216	237, 241	
		$Ex1$	92.60, 91.20	72.00, 70.00	60.60, 57.00	52.80, 53.60	53.80, 53.20	59.13
		$Ex2$	92.60, 91.20	82.80, 80.20	71.60, 73.20	70.60, 69.00	67.40, 65.60	72.55
		$Ex3$	92.60, 91.20	81.60, 84.80	74.20, 78.80	72.60, 75.60	70.80, 72.00	76.30
	PNN	$P(t)$	53, 55	153, 140	224, 200	233, 208	228, 229	
		$Ex1$	91.20, 91.20	73.00, 70.20	61.40, 57.80	52.80, 53.80	53.60, 53.20	59.48
		$Ex2$	91.20, 91.20	81.00, 78.20	71.40, 72.20	70.20, 68.00	65.80, 64.40	71.40
		$Ex3$	91.20, 91.20	45.20, 82.60	75.20, 63.00	78.80, 70.80	74.40, 50.00	67.50
	HYB	$P(t)$	254, 256	250, 238	282, 262	278, 249	295, 264	
		$Ex1$	93.00, 91.40	73.00, 70.60	60.60, 57.00	52.80, 53.60	53.60, 53.20	59.30
		$Ex2$	93.00, 91.40	73.00, 60.60	33.20, 58.80	43.60, 35.40	70.20, 55.00	53.73
		$Ex3$	93.00, 91.40	75.00, 73.80	61.00, 58.60	53.00, 53.80	54.00, 53.80	60.38

The evaluations had been performed with the "non-stationary" data sets generated by the "Model 1" and "Model 2". Here, the two *Acc* values of *original* data set refer to the results when the source training and testing sets, $S_T(0)$ and $S_V(0)$, are then interchanged. Also, the *Acc* of *non-stationary* data sets are the classification accuracies (%) for the corresponding data sets, $S_T(t)$ and $S_V(t)$, $t = 1, \ldots, 4$, respectively. The results reported in the final column, $\overline{Acc}$, are the average *Acc* rates (%). Finally, the two "integer" numerics of the $P(t)$ row of each PRS, are the numbers of prototype vectors extracted from the data sets of $S_T(t)$ and $S_V(t)$, $t = 0, \ldots, 4$, respectively.

we can see a comparison of the results obtained with the *Ex*1, *Ex*2 and *Ex*3 evaluation methods for the CNN, the PNN and the HYB methods.

Consider the CNN method for the "Arrhythmia" data set in Table 5. First of all, for the "1" model, if the *non-stationary* data sets were evaluated with the *Ex*1, the resulting averaged classification accuracy, $\overline{Acc}$, is 78.76%. Then, in *Ex*2, the $\overline{Acc}$ is 91.81%. However, in *Ex*3, the $\overline{Acc}$ is 97.62%. Again, in the cases of the PNN and HYB, we can see the *increasing* accuracies such as 78.82%, 95.63%, 98.40% and 87.83%, 96.85%, 97.24%, respectively. For the "2" model, the same characteristics can be observed. The details are omitted here in the interest of compactness.

For the PNN and HYB methods, the same accuracy characteristics can be observed for both models "1" and "2" in all the tables. From the above considerations, we can see a comparison of the results obtained with the *Ex*1, *Ex*2 and *Ex*3 methods. The comparison demonstrates that the prototype vectors of the *non-stationary* data sets can be extracted efficiently by employing the proposed philosophy. Indeed, such accuracy results are also typical of all the data sets used.

To give a fair perspective of the advantage gained by our method, it is pertinent to point out that there are many performance aberrations which we are unable to explain. For example, in Table 4, there are a few places where *Ex*2 performs better than the proposed method, *Ex*3. It should be emphasized that these cases are not the norm, but rather the exceptions, and probably occur due to the fact that the PRS invoked on the *current* data set (without invoking the characteristics of the data sets for their previous time instants) is adequate when it concerns the accuracy, although the time involved is much more. Similarly, in Tables 3 and 4, in some entries, a sudden decline of the accuracy can be observed. For example, in Table 3, when the PNN is used for Model 2, the accuracy abruptly falls to 45.20% from 91.20%, and in Table 4, when the HYB is used for Model 2, the accuracy again abruptly falls to 25.64% from 93.32%. We believe that these are a consequence of the *specific data-related* properties of the prototypes selected, and are thus not the norm. It should however, be mentioned that the PRS that seems to be the most "troublesome" for artificial data sets is the PNN.[12]

[12] We are grateful to the anonymous Referee who pointed out these issues to us.

Table 4
The classification accuracy rates (%) of the time-varying PRSs for the artificial data set, "Non_normal 3"

# of model	PRS	Evalu. method	*Acc* of *original* data	*Acc* of *non-stationary* data sets				$\overline{Acc}$
				$S_T(1), S_V(1)$	$S_T(2), S_V(2)$	$S_T(3), S_V(3)$	$S_T(4), S_V(4)$	
1	CNN	$P(t)$	503, 477	5000, 5000	5000, 5000	5000, 5000	5000, 5000	
		$Ex1$	91.74, 91.88	89.44, 90.84	85.82, 87.22	80.12, 81.10	75.26, 76.66	83.31
		$Ex2$	91.74, 91.88	90.48, 90.80	89.20, 89.44	86.90, 87.02	85.40, 85.72	88.12
		$Ex3$	91.74, 91.88	89.78, 90.80	87.46, 88.68	85.54, 85.96	85.26, 84.28	87.22
	PNN	$P(t)$	4891, 3356	603, 3542	768, 900	885, 1019	2692, 1174	
		$Ex1$	92.10, 92.04	89.86, 91.00	86.08, 87.32	80.34, 81.00	75.60, 76.60	83.48
		$Ex2$	92.10, 92.04	90.14, 89.86	88.54, 87.66	85.76, 85.46	84.96, 83.92	87.04
		$Ex3$	92.10, 92.04	90.44, 91.42	87.40, 88.56	84.68, 84.88	80.86, 80.72	86.12
	HYB	$P(t)$	1194, 1218	1290, 1330	1374, 1450	1472, 1554	1592, 1666	
		$Ex1$	92.74, 93.32	90.92, 91.74	87.36, 87.74	81.28, 81.62	75.64, 76.04	84.04
		$Ex2$	92.74, 93.32	55.28, 91.34	87.48, 54.92	87.86, 88.00	53.74, 22.00	67.58
		$Ex3$	92.74, 93.32	91.80, 92.02	89.76, 89.54	86.94, 86.24	86.74, 81.56	88.08
2	CNN	$P(t)$	503, 477	5000, 5000	5000, 5000	5000, 5000	5000, 5000	
		$Ex1$	91.74, 91.88	73.30, 73.42	60.08, 59.18	54.96, 54.38	52.50, 52.76	60.07
		$Ex2$	91.74, 91.88	81.46, 81.64	75.54, 76.20	72.02, 71.40	70.14, 69.58	74.75
		$Ex3$	91.74, 91.88	82.48, 84.48	81.82, 81.24	78.34, 78.94	78.68, 76.00	80.25
	PNN	$P(t)$	4891, 3356	1347, 4739	1870, 1780	2135, 2069	2316, 2139	
		$Ex1$	92.10, 92.04	73.90, 73.62	60.30, 59.54	55.10, 54.38	52.54, 52.82	60.28
		$Ex2$	92.10, 92.04	80.26, 81.10	72.96, 74.40	70.94, 69.32	67.78, 67.84	73.08
		$Ex3$	92.10, 92.04	76.94, 82.26	62.14, 73.72	55.56, 61.48	53.00, 55.54	65.08
	HYB	$P(t)$	1194, 1218	1686, 1736	2102, 2068	2286, 2227	2476, 2331	
		$Ex1$	92.74, 93.32	72.68, 73.38	58.92, 59.04	53.34, 54.40	51.32, 52.76	59.48
		$Ex2$	53.22, 58.02	92.74, 93.32	81.48, 25.64	28.98, 71.18	55.74, 56.02	63.14
		$Ex3$	92.74, 93.32	84.14, 78.38	81.88, 65.96	52.98, 66.46	51.76, 63.70	68.16

The evaluations had been performed with the "non-stationary" data sets generated by the "Model 1" and "Model 2". The rest of the nomenclature is as described in the caption for Table 3.

Table 5
The classification accuracy rates (%) of the time-varying PRSs for the real-life data set, "Arrhythmia"

# of model	PRS	Evalu. method	*Acc* of *original* data	*Acc* of *non-stationary* data sets				$\overline{Acc}$
				$S_T(1), S_V(1)$	$S_T(2), S_V(2)$	$S_T(3), S_V(3)$	$S_T(4), S_V(4)$	
1	CNN	$P(t)$	31, 31	34, 33	27, 31	40, 44	40, 42	
		$Ex1$	95.58, 97.79	87.61, 95.58	78.32, 93.36	67.70, 78.32	61.95, 67.26	78.76
		$Ex2$	95.58, 97.79	93.81, 92.04	91.59, 95.13	89.82, 92.92	88.50, 90.71	91.81
		$Ex3$	95.58, 97.79	98.67, 99.12	98.23, 97.79	96.90, 96.02	97.79, 96.46	97.62
	PNN	$P(t)$	3, 8	4, 8	6, 7	9, 6	4, 12	
		$Ex1$	98.67, 96.02	96.90, 97.79	90.71, 85.40	71.24, 70.80	57.08, 60.62	78.82
		$Ex2$	98.67, 96.02	97.79, 95.13	98.67, 95.13	95.13, 92.92	97.35, 92.92	95.63
		$Ex3$	98.67, 96.02	99.12, 98.23	99.56, 97.35	99.56, 96.90	100.0, 96.46	98.40
	HYB	$P(t)$	190, 180	188, 175	190, 176	188, 176	189, 179	
		$Ex1$	99.12, 99.12	96.90, 98.23	92.04, 95.58	84.51, 89.82	72.57, 73.01	87.83
		$Ex2$	99.12, 99.12	97.79, 97.79	97.79, 96.90	97.35, 97.79	95.58, 93.81	96.85
		$Ex3$	99.12, 99.12	98.23, 99.56	96.90, 97.79	96.90, 97.35	94.25, 96.90	97.24
2	CNN	$P(t)$	31, 31	31, 33	34, 37	55, 25	37, 31	
		$Ex1$	95.58, 97.79	80.53, 80.53	64.16, 65.49	57.97, 62.83	58.85, 57.08	65.93
		$Ex2$	95.58, 97.79	95.13, 93.36	92.04, 84.96	87.61, 84.96	92.48, 81.42	89.00
		$Ex3$	95.58, 97.79	99.56, 98.67	99.12, 97.79	98.23, 97.35	96.46, 92.04	97.40
	PNN	$P(t)$	3, 8	16, 9	3, 15	18, 11	25, 6	
		$Ex1$	98.67, 96.02	78.76, 73.45	57.52, 57.08	54.43, 56.64	54.43, 54.43	60.84
		$Ex2$	98.67, 96.02	92.92, 92.92	98.23, 89.38	90.27, 92.92	92.04, 86.28	91.87
		$Ex3$	98.67, 96.02	98.67, 96.46	99.12, 94.69	96.46, 94.25	95.13, 93.36	96.02
	HYB	$P(t)$	190, 180	146, 130	113, 104	124, 112	154, 159	
		$Ex1$	99.12, 99.12	88.50, 82.30	64.60, 63.27	56.20, 54.87	56.20, 54.87	65.10
		$Ex2$	99.12, 99.12	96.02, 97.79	99.56, 90.27	99.12, 97.35	92.48, 91.59	95.52
		$Ex3$	99.12, 99.12	99.15, 98.23	98.67, 96.02	98.23, 95.13	96.46, 93.81	96.96

The evaluations had been performed with the "non-stationary" data sets generated by the "Model 1" and "Model 2". The rest of the nomenclature is as described in the caption for Table 3.

Table 6
The classification accuracy rates (%) of the time-varying PRSs for the artificial data set, "Adult4"

# of model	PRS	Evalu. method	*Acc* of *original* data	*Acc* of *non-stationary* data sets				$\overline{Acc}$
				$S_T(1), S_V(1)$	$S_T(2), S_V(2)$	$S_T(3), S_V(3)$	$S_T(4), S_V(4)$	
1	CNN	$P(t)$	743, 737	772, 766	837, 818	831, 797	841, 850	
		$Ex1$	91.10, 91.55	91.48, 91.65	93.16, 91.94	93.57, 92.06	94.22, 92.27	92.54
		$Ex2$	91.10, 91.55	90.19, 88.65	89.25, 89.11	89.23, 88.55	88.48, 89.08	89.07
		$Ex3$	91.10, 91.55	92.35, 91.19	93.50, 92.25	93.45, 92.95	93.78, 92.78	92.78
	PNN	$P(t)$	660, 659	591, 592	606, 586	633, 605	592, 601	
		$Ex1$	88.72, 89.97	89.78, 90.09	91.39, 90.35	93.04, 91.12	93.47, 92.01	91.41
		$Ex2$	88.72, 89.97	86.99, 85.43	86.99, 86.97	85.67, 85.22	85.60, 85.84	86.09
		$Ex3$	88.72, 89.97	94.62, 92.87	94.82, 93.38	94.62, 93.64	93.45, 93.52	93.87
	HYB	$P(t)$	497, 470	484, 473	487, 472	488, 462	481, 485	
		$Ex1$	95.10, 93.55	95.34, 94.31	95.32, 94.74	95.03, 94.53	94.77, 94.53	94.82
		$Ex2$	95.10, 93.55	93.31, 92.20	92.63, 92.01	92.08, 91.41	91.55, 91.46	92.08
		$Ex3$	95.10, 93.55	95.27, 93.98	94.86, 94.02	94.58, 93.74	94.62, 93.57	94.33
2	CNN	$P(t)$	743, 737	592, 536	581, 655	645, 615	638, 665	
		$Ex1$	91.10, 91.55	93.23, 92.01	94.55, 90.69	94.91, 93.57	94.88, 90.52	93.05
		$Ex2$	91.10, 91.55	89.92, 88.34	88.75, 87.62	88.67, 88.31	87.43, 87.45	88.31
		$Ex3$	91.10, 91.55	93.09, 93.98	94.43, 93.62	94.53, 93.98	94.62, 94.43	94.09
	PNN	$P(t)$	660, 659	592, 536	581, 655	645, 615	638, 665	
		$Ex1$	88.72, 89.97	92.85, 88.00	94.50, 76.03	94.82, 71.71	94.94, 72.59	85.68
		$Ex2$	88.72, 89.97	86.73, 84.67	84.83, 83.15	84.62, 83.54	84.21, 83.18	84.37
		$Ex3$	88.72, 89.97	94.62, 93.43	94.82, 93.95	94.62, 94.41	93.45, 94.46	94.22
	HYB	$P(t)$	497, 470	483, 465	465, 473	501, 492	590, 573	
		$Ex1$	95.10, 93.55	87.71, 75.43	92.66, 46.63	93.88, 33.26	94.32, 26.88	68.85
		$Ex2$	95.10, 93.55	90.40, 88.65	89.51, 88.99	90.07, 88.99	86.97, 86.75	88.79
		$Ex3$	95.10, 93.55	93.57, 93.59	94.26, 94.53	94.60, 94.24	94.67, 94.17	94.20

The evaluations had been performed with the "non-stationary" data sets generated by the "Model 1" and "Model 2". The rest of the nomenclature is as described in the caption for Table 3.

4.3. Experimental results: the time complexity

We report below the time complexity of the *proposed* PRS algorithm for the "medium-sized" time varying data sets.

In *Ex*1, the prototype set of the "original" data set is used for all the four "non-stationary" data sets. So, no additional time is required for extracting the prototype vectors. On the other hand, in the *Ex*2 evaluation method, each prototype set has been extracted from the corresponding non-stationary data set through a conventional PRS method. In *Ex*3, the prototypes of the "current" non-stationary data set, $S(t+1)$, were selected from the "previous" data set, namely, $P(t)$, by adjusting the $P(t)$ with the $S(t+1)$ using an LVQ3-type algorithm. Therefore, the processing CPU-time of the *Ex*3 method is merely the time required for adjusting the prototypes using the LVQ3. However, in the adjusting process, the additional times involve those required for selecting optimal or near optimal values for the parameters such as the window length, w, the learning rate, α, the relative learning rate, ε, and the number of iteration steps, η, are required.

The processing CPU-times of the PNN and HYB[13] for the "Non_n2", "Non_n3", "Arrhy" and "Adult4" data sets are shown in Tables 7 and 8, respectively.

[13] The time for using the CNN as the kernel is excessive and so is not included here.

In the cases of the PNN and HYB methods, (unlike for the CNN), the situation is different as shown in Tables 7 and 8. From the results of the tables, we can see a comparison of the results obtained with the *Ex*2 and *Ex*3 for the "artificial" and "real-life" data sets.

In Table 7, consider the PNN method for the *four* data sets, namely, "Non_n2", "Non_n3", "Arrhy" and "Adult4". First of all, for "1" model, the averaged CPU-times for the *Ex*2 and *Ex*3 are 0.30, 0.01, 315.69, 4.79, 1.05, 0.02 and 394.58, 1.09 min, respectively. Then, for '2' model, the averaged CPU-times for the *Ex*2 and *Ex*3 are 0.35, 0.01, 428.54, 4.72, 1.01, 0.02 and 184.23, 1.10 min, respectively. For both "1" and "2" models in Table 8, the same characteristics can be observed from the results of the *four* data sets. The details are omitted here in the interest of compactness.

From these considerations, the reader should observe that the proposed philosophy of *Ex*3 needs less time than that of *Ex*2 in the cases of the PNN and HYB methods.

5. Conclusions

All of the prototype reduction schemes (PRS), which have been reported in the literature, process the time unvarying stationary data to yield a subset of prototypes that are useful in nearest-neighbor-like classification. In this paper we have

Table 7
The processing CPU-time (minutes) of the PNN method for the “Non_normal 2” (in short “Non_n2”), “Non_normal 3” (in short “Non_n3”), “Arrhythmia” (in short “Arrhy”) and “Adult4” data sets

# of model	Data set	Evalu. methods	*T* of *non-stationary* data sets				$\overline{T}$
			$S_T(1), S_V(1)$	$S_T(2), S_V(2)$	$S_T(3), S_V(3)$	$S_T(4), S_V(4)$	
1	Non_n2	$Ex2$	0.27, 0.29	0.30, 0.31	0.30, 0.30	0.31, 0.30	0.30
		$Ex3$	0.01, 0.01	0.01, 0.01	0.01, 0.01	0.01, 0.01	0.01
	Non_n3	$Ex2$	195.16, 507.20	200.22, 243.95	204.50, 252.75	645.27, 276.43	315.69
		$Ex3$	6.30, 3.31	6.30, 3.30	6.26, 3.30	6.24, 3.29	4.79
	Arrhy	$Ex2$	1.03, 1.05	1.03, 1.05	1.04, 1.08	1.08, 1.06	1.05
		$Ex3$	0.01, 0.02	0.01, 0.02	0.01, 0.02	0.01, 0.02	0.02
	Adult4	$Ex2$	186.68, 1890.65	184.48, 176.82	167.32, 164.42	186.76, 199.47	394.58
		$Ex3$	1.08, 1.11	1.08, 1.11	1.07, 1.11	1.07, 1.11	1.09
2	Non_n2	$Ex2$	0.31, 0.31	0.30, 0.39	0.39, 0.30	0.37, 0.40	0.35
		$Ex3$	0.01, 0.01	0.01, 0.01	0.01, 0.01	0.01, 0.01	0.01
	Non_n3	$Ex2$	220.87, 582.53	284.98, 308.84	608.84, 466.62	457.35, 498.29	428.54
		$Ex3$	6.25, 3.28	6.20, 3.26	6.07, 3.26	6.17, 3.26	4.72
	Arrhy	$Ex2$	0.98, 1.01	1.03, 1.00	1.00, 1.02	0.98, 1.03	1.01
		$Ex3$	0.01, 0.02	0.01, 0.02	0.01, 0.02	0.01, 0.02	0.02
	Adult4	$Ex2$	162.34, 183.65	187.90, 187.98	193.97, 178.70	190.68, 188.65	184.23
		$Ex3$	1.08, 1.12	1.08, 1.11	1.08, 1.11	1.07, 1.11	1.10

The evaluations were obtained with the “non-stationary” data sets generated by “Model 1” and “Model 2”. Here, the *T* of *non-stationary* data sets are the processing CPU-time for the corresponding data sets, $S_T(t)$ and $S_V(t)$, $t = 1, \ldots, 4$, respectively. Also, the two *T* values of each data set refer to the results when the source training and testing sets, $S_T(0)$ and $S_V(0)$, are interchanged. The results written in the final column, $\overline{T}$, are the average processing times.

Table 8
The processing CPU-time (seconds) of the HYB method for the “Non_normal 2” (in short “Non_n2”), “Non_normal 3” (in short “Non_n3”), “Arrhythmia” (in short “Arrhy”) and “Adult4” data sets

# of model	Data set	Evalu. methods	*T* of *non-stationary* data sets				$\overline{T}$
			$S_T(1), S_V(1)$	$S_T(2), S_V(2)$	$S_T(3), S_V(3)$	$S_T(4), S_V(4)$	
1	Non_n2	$Ex2$	2.16, 2.21	5.85, 2.24	2.29, 2.26	2.27, 2.51	2.72
		$Ex3$	1.98, 2.00	1.95, 2.00	1.98, 2.05	1.98, 2.03	2.00
	Non_n3	$Ex2$	74.00, 85.14	85.52, 84.55	220.00, 91.54	98.10, 123.17	107.75
		$Ex3$	74.48, 76.02	74.39, 75.95	74.38, 75.92	74.36, 76.08	75.07
	Arrhy	$Ex2$	17.86, 16.69	18.04, 16.76	17.83, 16.69	17.94, 17.11	17.37
		$Ex3$	17.14, 16.59	17.17, 16.33	17.23, 16.36	17.23, 16.38	16.80
	Adult4	$Ex2$	49.74, 48.86	151.23, 48.84	50.16, 47.66	49.48, 50.99	62.12
		$Ex3$	48.55, 45.92	48.48, 45.97	48.59, 45.97	48.52, 46.17	47.27
2	Non_n2	$Ex2$	2.05, 1.95	2.22, 2.22	2.36, 2.10	2.43, 2.35	2.21
		$Ex3$	1.97, 2.00	2.03, 2.08	1.98, 2.00	2.00, 2.05	2.01
	Non_n3	$Ex2$	75.65, 109.53	78.94, 232.25	142.52, 136.31	157.68, 154.16	135.88
		$Ex3$	69.58, 75.98	69.48, 76.38	69.42, 76.11	69.48, 76.17	72.83
	Arrhy	$Ex2$	14.04, 12.60	10.99, 10.25	12.01, 10.93	14.73, 15.61	12.65
		$Ex3$	17.13, 16.22	17.16, 16.19	17.11, 16.20	17.13, 16.20	16.67
	Adult4	$Ex2$	53.10, 48.53	49.39, 49.96	53.43, 51.40	61.72, 59.57	53.39
		$Ex3$	48.48, 45.94	48.52, 46.02	48.50, 45.97	48.59, 45.95	47.25

The rest of the nomenclature is as described in the caption for Table 7.

proposed a mechanism applicable for the non-stationary data sets. In the proposed time varying PRS method, the prototypes of the non-stationary data versions of a given data set, can be obtained by adjusting the *previous* prototypes with the *current* non-stationary data set using an LVQ3-type algorithm.

The proposed method was tested on both artificial and real-life benchmark time varying data sets, and compared with a few representative conventional methods. The experimental results for small and medium-sized non-stationary data sets demonstrate that the proposed algorithm can improve the reduction rate of the conventional PRSs such as the CNN, PNN and HYB methods, and that their classification accuracies are comparable, although they require almost the same or less CPU-times.

Acknowledgements

The authors are very grateful to the anonymous Referee who gave us numerous comments which improved the quality of this paper. Thanks a lot!

References

[1] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: a review, IEEE Trans. Pattern Anal. Machine Intell. PAMI-22 (1) (2000) 4–37.

[2] B.V. Dasarathy, Nearest Neighbour (NN) Norms: NN Pattern Classification Techniques, IEEE Computer Society Press, Los Alamitos, 1991.

[3] J.C. Bezdek, L.I. Kuncheva, Nearest prototype classifier designs: an experimental study, Int. J. Intell. Syst. 16 (12) (2001) 1445–1473.

[4] P.E. Hart, The condensed nearest neighbour rule, IEEE Trans. Inform. Theory IT-14 (1968) 515–516.

[5] G.W. Gates, The reduced nearest neighbour rule, IEEE Trans. Inform. Theory IT-18 (1972) 431–433.

[6] C.L. Chang, Finding prototypes for nearest neighbour classifiers, IEEE Trans. Comput. C-23 (11) (1974) 1179–1184.

[7] G.L. Ritter, H.B. Woodruff, S.R. Lowry, T.L. Isenhour, An algorithm for a selective nearest neighbour rule, IEEE Trans. Inform. Theory IT-21 (1975) 665–669.

[8] I. Tomek, Two modifications of CNN, IEEE Trans. Syst. Man and Cybern. SMC-6 (6) (1976) 769–772.

[9] P.A. Devijver, J. Kittler, On the edited nearest neighbour rule, in: Proceedings of the Fifth International Conference on Pattern Recognition, December 1980, pp. 72–80.

[10] K. Fukunaga, J.M. Mantock, Nonparametric data reduction, IEEE Trans. Pattern Anal. Machine Intell. PAMI-6 (1) (1984) 115–118.

[11] K. Fukunaga, Introduction to Statistical Pattern Recognition, second ed., Academic Press, San Diego, 1990.

[12] Q. Xie, C.A. Laszlo, R.K. Ward, Vector quantization techniques for nonparametric classifier design, IEEE Trans. Pattern Anal. Machine Intell. PAMI-15 (12) (1993) 1326–1330.

[13] Y. Hamamoto, S. Uchimura, S. Tomita, A bootstrap technique for nearest neighbour classifier design, IEEE Trans. Pattern Anal. Machine Intell. PAMI-19 (1) (1997) 73–79.

[14] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining Knowledge Discovery 2 (2) (1998) 121–167.

[15] S.-W. Kim, B.J. Oommen, Enhancing prototype reduction schemes with LVQ3-type algorithms, Pattern Recognition 36 (5) (2003) 1083–1093.

[16] S.-W. Kim, B.J. Oommen, A Brief Taxonomy and Ranking of Creative Prototype Reduction Schemes, Pattern Anal. Appl. J. 6 (3) (2003) 232–244.

[17] S.-W. Kim, B.J. Oommen, Enhancing Prototype Reduction Schemes with Recursion: A Method Applicable for “Large” Data Sets, IEEE Trans. Syst. Man Cybern.—Part B SMC-34 (3) (2004) 1384–1397.

[18] Y. Linde, A. Buzo, R. Gray, An algorithm for vector quantizer design, IEEE Trans. Commun. COM-28 (1) (1980) 84–95.

[19] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, em SOM_PAK: the Self-Organizing Map Program Package. Technical Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, FIN-02150 Espoo, Finland, 1996. Can also be downloaded (as of June 2005) from http://cochlea.hut.fi/research/som_lvq_pak.shtml.

[20] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, K. Torkkola, LVQ_PAK: the Learning Vector Quantization Program Package, Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, FIN-02150 Espoo, Finland, 1996. Can also be downloaded (as of June 2005) from http://cochlea.hut.fi/research/som_lvq_pak.shtml.

[21] T. Kohonen, Self-Oganizing Maps, Springer, Berlin, 1995.

[22] N. Aras, B.J. Oommen, I.K. Altinel, The Kohonen network incorporating explicit statistics and its application to the travelling salesman problem, Neural Networks (1999) 1273–1284.

[23] N. Aras, I.K. Altinel, B.J. Oommen, A Kohonen-like decomposition method for the traveling salesman problem—KNIES_DECOMPOSE, in: Proceedings of the ECAI’2000, the 14th European Conference on Artificial Intelligence, Berlin, Germany, August 2000, pp. 261–265.

[24] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.

[25] J.C. Bezdek, T.R. Reichherzer, G.S. Lim, Y. Attikiouzel, Multiple-prototype classifier design, IEEE Trans. Syst. Man Cybern.—Part C SMC-28 (1) (1998) 67–79.

[26] L.I. Kuncheva, J.C. Bezdek, Nearest prototype classification: clustering, genetic algorithms or random search?, IEEE Trans. Syst. Man Cybern.—Part C SMC-28 (1) (1998) 160–164.

[27] C.L. Blake, C.J. Merz, UCL Machine Learning Databases, Department of Information and Computer Science, University of California, Irvine, CA. Can also be downloaded (as of June 2005) from http://www.ics.uci.edu/mlearn/MLRepository.html.

Further reading

[28] B.V. Dasarathy, Minimal Consistent Set (MCS) identification for optimal nearest neighbor decision systems design, IEEE Trans. Syst. Man Cybern. 24 (3) (1994) 511–517.