Experimental Study on Prototype Optimisation Algorithms for Prototype-based Classification [1]

M. Lozano, J.M. Sotoca, J.S. Sánchez and F. Pla [a]
E. Pękalska and R.P.W. Duin [b]

[a] *Dept. Lenguajes y Sistemas Informáticos*
Universitat Jaume I, Campus Riu Sec, 12071 Castellón, Spain

[b] *Faculty of Electrical Engineering, Mathematics and Computer Science*
Mekelweg 4, 2628 CD Delft, Delft University of Technology, The Netherlands

Abstract

Prototype-based classification relies on the distances between the examples to be classified and carefully chosen prototypes. A small set of prototypes is of interest to keep the computational complexity low, while maintaining high classification accuracy. An experimental study of some old and new prototype optimisation techniques is presented, in which the prototypes are either selected or generated from the given data. These condensing techniques are evaluated on real data, represented in vector spaces, by comparing their resulting reduction rates and classification performance.

Usually the determination of prototypes is studied in relation with the nearest neighbour rule. We will show that the use of more general dissimilarity-based classifiers can be more beneficial. An important point in our study is that the adaptive condensing schemes here discussed allow the user to choose the number of prototypes freely according to the needs. If such techniques are combined with linear dissimilarity-based classifiers, they provide the best trade-off of small condensed sets and high classification accuracy.

Key words: dissimilarity, representation, prototype selection, adaptive condensing, EM algorithm, normal density based classifiers, nearest neighbour rule

Email addresses: {lozano,sotoca,sanchez,pla}@uji.es, {e.pekalska,r.p.w.duin}@ewi.tudelft.nl (E. Pękalska and R.P.W. Duin).

[1] This work has been partially supported by grants DPI2001-2956-C02-02 and TIC2003-08496 from the Spanish CICYT and project IST-2001-37306 from the European Union, as well as by the Dutch Organisation for Scientific Research (NWO).

1 Introduction

An intuitive way of determining the class of an unknown object is by analysing its similarity to a set of prototypes either selected or generated from a given Training Set (TS) of objects with known class labels. In general, similarities or dissimilarities can be computed either from the raw object observations or based on an intermediate feature representation. A small set of prototypes has the advantage of a low computational cost and small storage requirements, while leading to similar, or even improved, classification performance. Various ways of designing a prototype set can be studied in Euclidean vector spaces. Two families of such optimisation procedures are editing and condensing.

Editing is the step in a learning process in charge of increasing the accuracy of predictions, when there is substantial noise in the training data. A basic editing algorithm removes noisy instances, as well as close border cases, eliminating a possible overlap between the regions from different classes and leaving smoother decision boundaries. Wilson introduced the first editing method (32). Briefly, the k-Nearest Neighbour (k-NN) rule is used to estimate the class of each prototype in the TS and to remove those whose class labels do not agree with the ones judged by the k-NN rule. This algorithm tries to eliminate mislabelled objects from the TS as well as those near to the decision boundaries. Many researchers have addressed the problem of editing by proposing alternative schemes (1; 6; 9; 33).

The *condensing* step aims at selecting a small subset of prototypes without a significant degradation in the classification accuracy. Two main groups of condensing techniques can be distinguished. These are the selective schemes, which merely select a subset of the original training objects (1; 7; 13; 30; 31) and the adaptive schemes which modify them (2; 3; 5; 14; 16; 17).

This paper discusses prototype optimisation methods, such as editing and condensing, for feature-based representations of classes in the context of prototype-based classification. Traditionally, the 1-NN rule is used for this purpose. It classifies objects based on the minimum distance to the given prototypes. Here, we show that the prototype sets, optimised to guarantee good performance of the 1-NN rule, lead to a higher classification accuracy, when more general dissimilarity-based classifiers are considered, as recently proposed (27; 28). These are weighted linear or quadratic combinations of the (Euclidean) distances computed between the test objects and the prototype sets found by dedicated condensing algorithms. Linear classifiers are especially of interest, since their computational complexity is comparable to that of the 1-NN rule.

Although the algorithms are run in vector spaces, the distances are by no means restricted to the Euclidean metric. Other distances, such as the l_p-

distances, $d(\boldsymbol{x},\boldsymbol{y}) = (\sum_i |x_i - y_i|^p)^{\frac{1}{p}}$, $p > 0$ (metric for $p \geq 1$), or inner-product based distances, $d(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{\boldsymbol{x}^T\boldsymbol{y}}{||\boldsymbol{x}||||\boldsymbol{y}||}$, can be used. Moreover, the selective techniques can easily be applied to non-vectorial data (such as strings, graphs or shapes), provided that the pairwise dissimilarities are derived. This aspect has already been confirmed in earlier studies. There, the prototypes were often chosen at random (22; 23; 24; 25). Other object selection techniques, based on feature selection or clustering, were also investigated and appeared to be good for small prototype sets (27; 28).

In this paper we will show that more general dissimilarity-based classifiers may successfully replace the 1-NN rule, especially when small sets of prototypes are needed. We will focus on Euclidean distances to maintain the connection with the traditionally used dissimilarities for selective condensing schemes in vector spaces. We will also compare these with adaptive reduction algorithms. Consequently, we focus on (Euclidean) distances in vector spaces and the use of condensing schemes determined in favour of the 1-NN rule. Adaptive schemes are especially of interest, as they can only be used when vectorial representations are available. Consequently, as they offer more flexibility, they should be more beneficial in vector spaces than the selective approaches.

Four condensing techniques are studied. Experiments are conducted to compare their ability to reduce the training size, while maintaining the discriminative power of the optimised prototypes. The classification performance is judged by the 1-NN rule and two more general dissimilarity-based classifiers.

The paper is organised as follows. Section 2 briefly reviews a number of condensing techniques which are used in our study. These are *MaxNCN* (16; 29), *Reconsistent* (16), *LVQ* (15) and *MixtGauss* (17). Section 3 briefly describes the framework of the prototype-based classification methods used for the evaluation of the derived condensed sets. In Section 4, the data sets are presented and the experiments are described, providing quantitative results and a further discussion. Finally, the main conclusions are summarised in Section 5.

2 Condensing methods to compare

Assume a training set (TS) of N instances, $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, representing J classes, $C = \{c_1, \ldots, c_J\}$. Each instance $\boldsymbol{x}_i$ is a point (a vector) in an n-dimensional feature space $\Re^n$. A condensed set consists of r, $r \ll n$, prototypes, which are either selected or generated from the examples of X. They are determined to represent efficiently the distributions of the classes and be well discriminative, when used to classify the training objects. Their cardinality should be sufficiently small to reduce both the storage and evaluation time.

A condensed set is said to be *consistent* with respect to a TS if the classification error, estimated by assigning all objects from the TS to the classes of their nearest neighbours in the condensed set, is small; see (6; 13). Therefore, given a set of prototypes representing the class distribution, the classification rate of the TS can be used to measure the consistency of this set. As the consistent condition is maintained, the smaller the number of prototypes in the condensed set, the better the final result is.

Four condensing algorithms are presented below. These are two selective schemes, *MaxNCN* and *Reconsistent*, and two adaptive schemes, *LVQ* and *MixtGauss*.

2.1 MaxNCN

The geometrical distribution of objects in a vector space may be more informative than the distances between them. The so-called *surrounding neighbourhood-based rules* (29) try to improve the traditional nearest neighbour approach by making use of such information, especially in relation to the training objects which are nearby the decision boundaries. This can be achieved by taking into account not only the proximity of objects to a given input sample but also their *symmetrical distribution* around it. As a result, the surrounding neighbours of an object $\boldsymbol{p}$ should satisfy two complementary conditions:

(1) Distance criterion: the neighbours should be as close as possible to $\boldsymbol{p}$.
(2) Symmetry criterion: the neighbours should be distributed as homogeneously as possible around $\boldsymbol{p}$.

Chaudhuri (4) proposed the concept of a Nearest Centroid Neighbourhood (NCN), which can be viewed as a particular realisation of the surrounding neighbourhood. Let $\boldsymbol{p}$ be a given object whose k-NCN (k Nearest Centroid Neighbours) should be found in a TS. These k neighbours can be determined by the following iterative procedure:

(1) Find the nearest neighbour of $\boldsymbol{p}$. Choose it as the first NCN, $\boldsymbol{q}_1$.
(2) Select the i-th NCN, $\boldsymbol{q}_i$, $i \geq 2$, from X such that the centroid defined by $\boldsymbol{q}_i$ and the previously selected NCNs, $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_i$ is the closest to $\boldsymbol{p}$.

As designed, the NCN search method is incremental and the objects around a given sample have a geometrical distribution that tends to surround it. Moreover, the region of influence of the NCN is larger than that of the NN, in general. This is illustrated in Figure 1, where the regions defined by the 4-NCN and the 4-NN are shown for the given point $\boldsymbol{p}$.

The *MaxNCN* technique is based on the concept of NCN and relies on the NCN search algorithm, as presented above. A set of prototypes is selected from the

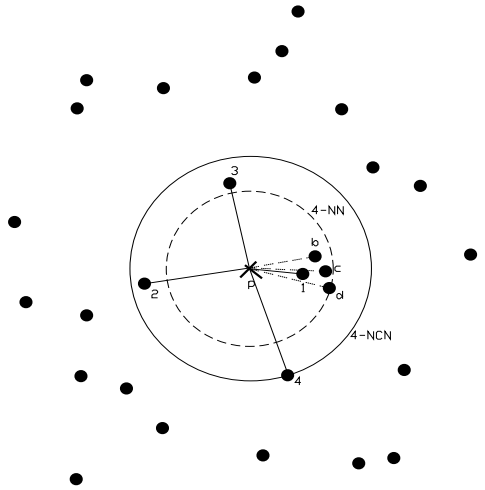

Figure 1. Illustration of the NCN concept.

Algorithm 1. *MaxNCN*

for $i = e ach\_prototype(TS)$ **do**
  $neighbours\_number[i] = 0$
  $neighbour = next\_neighbour(i)$
  while $neighbour.class == i.class$ **do**
    $neighbours\_vector[i] = Id(neighbour)$
    $neighbours\_number[i] + +$
    $neighbour = next\_neighbour(i)$
  end while
end for
while $Max\_neighbours() > 0$ **do**
  $EliminateNeighbours(id\_Max\_neighbours)$
end while

TS to guarantee their optimal geometrical distribution with respect to their NCNs. The use of the NCN of a given sample can provide local information about the shape of the probability class distribution, which depends on the nature and class of its NCN, that is, on the nature of the prototypes in its surrounding area.

The rationale behind this approach is that the prototypes belonging to the same class are located in a neighbouring area and can be replaced by a single representative without significantly affecting the original boundaries. The main reason to employ the NCN instead of the NN is to benefit from its properties, i.e. that the NCN covers a bigger region than the NN, and that these neighbours are located in the area of influence around a given sample which is compensated in terms of their geometrical distribution.

Initially, all training objects are considered as prototypes. The algorithm attempts to replace a group of neighbouring prototypes of the same class by a representative. In order to decide which group of prototypes should be re-

placed, the NCN of each prototype $\boldsymbol{p}$ in the TS is computed until reaching a neighbour of a class different than the class of $\boldsymbol{p}$. The prototype with the largest number of NCNs is defined as a representative of its corresponding group. Since this group lies in the area of influence defined by the NCN distribution, consequently, all its members can now be removed from the TS. Next, given the remaining prototypes, the algorithm updates the number of their neighbours (if some were previously eliminated) as belonging to the group of an already existing representative. This is repeated until there is no group of prototypes to be replaced by a representative. This basic scheme is called *MaxNCN*, and is presented in Algorithm 1.

Algorithm 2. *Reconsistent*

for $i = each\_prototype(TS)$ **do**
  $neighbours\_number[i] = 0$
  $neighbour = next\_neighbour(i)$
  while $neighbour.class == i.class$ **do**
    $neighbours\_vector[i] = Id(neighbour)$
    $neighbours\_number[i] + +$
    $neighbour = next\_neighbour(i)$
  end while
end for
while $Max\_neighbours() > 0$ **do**
  $EliminateNeighbours(id\_Max\_neighbours)$
end while
$count = 0$
for $i = each\_prototype(TS)$ **do**
  if $Classify(i)! = i.class$ **then**
    $incorrect\_class[count + +] = i$
  end if
end for
for $i = each\_prototype(incorrect\_class[])$ **do**
  $neighbours\_number\_inc[i] = 0$
  $neighbour\_inc = next\_neighbour\_inc(i)$
  while $neighbour\_inc.class == i.class$ **do**
    $neighbours\_vector\_inc[i] = Id(neighbour\_inc)$
    $neighbours\_number\_inc[i] + +$
    $neighbour\_inc = next\_neighbour\_inc(i)$
  end while
end for
while $Max\_neighbours\_inc() > 0$ **do**
  $EliminateNeighbours\_inc(id\_Max\_neighbours\_inc)$
end while
$AddCondensedIncToCondensedTS()$

2.2 Reconsistent

The *Reconsistent* algorithm is an important modification of the *MaxNCN* algorithm towards obtaining a consistent condensed set (16). The primary idea is that the consistency of a subset with respect to the TS should lead to a better classification. By using the *MaxNCN* algorithm some prototypes close to the decision boundaries are removed because of the order in which the instances are taken during the condensing process. The *Reconsistent* approach, proposed by us, tries to address this issue.

The procedure starts by applying the *MaxNCN* technique to a TS, yielding a reduced set, RS. Then, each object in the TS is tested by the 1-NN rule with respect to the current prototype set, RS. All misclassified objects form a new group, which is condensed using the RS set as the reference. In the end, this new condensed set is added to the RS, resulting in the final condensed set. The *Reconsistent* procedure is presented in Algorithm 2.

2.3 LVQ

In statistical pattern recognition, Learning Vector Quantisation (LVQ) is a popular competitive learning algorithm, used in many applications. Its goal is to approximate the distribution of classes by using a reduced set of prototypes, while minimising the classification error (14). It is, therefore, an adaptive condensing technique. In general, the classes can be described by a relatively small number of prototypes $\boldsymbol{p}_i$, placed within each class region of the decision boundary by means of measures of neighbourhood.

In the initialisation step, the prototypes are placed in the TS, maintaining the same number of prototypes in each class. The class borders are represented in a piecewise linear way by segments of midplanes between the prototypes of neighbouring classes (the borders of the so-called Voronoi tessellations). This may seem to be a good strategy for approximating the class borders, using the fact that the average distances between the neighbouring prototypes should be the same on both sides of the borders.

Nevertheless, the optimal placement of the prototypes is not known. Therefore, their distances and their optimal cardinality, k, cannot be determined beforehand. To fix the value of k for each case, the number of prototypes has to be chosen first. A combination of editing and condensing is used for this purpose. Therefore, first the Wilson's editing (k of the k-NN used in the editing scheme, as explained in Subsection 4.1) is used, and then followed by the Hart's condensing algorithm (13). The resulting size of the prototype set is close to ideal (9). Using this number of prototypes, different values of k were

tested, and the one leading to the highest classification accuracy was finally chosen. Different set sizes were used for each data set, in order to compare the results for different cases.

In the experiments, a variant of the original LVQ algorithm is used, namely the *Optimised-Learning-Rate*, *OLVQ1*, (15). The basis of this algorithm is the *LVQ1* (15) such that an individual learning rate $\alpha_i(t)$ is selected for each prototype $\boldsymbol{p}_i$. Several prototypes are assigned to each class such that the layout of the prototypes minimise approximately the misclassification errors in the 1-NN classification. The following equations define this process:

$$\begin{aligned}
&\boldsymbol{p}_c(t+1) = \boldsymbol{p}_c(t) + \alpha_c(t)[\boldsymbol{x}(t) - \boldsymbol{p}_c(t)], \text{ if } \boldsymbol{x} \text{ and } \boldsymbol{p_c} \text{ are in the same class,} \\
&\boldsymbol{p}_c(t+1) = \boldsymbol{p}_c(t) - \alpha_c(t)[\boldsymbol{x}(t) - \boldsymbol{p}_c(t)], \text{ if } \boldsymbol{x} \text{ and } \boldsymbol{p}_c \text{ are in different classes,} \\
&\boldsymbol{p}_i(t+1) = \boldsymbol{p}_i(t), \qquad\qquad\qquad\qquad\qquad \text{if } i \neq c
\end{aligned} \tag{1}$$

where $\boldsymbol{x}(t)$ is an input sample and $\boldsymbol{p}_c$ is the nearest $\boldsymbol{p}_i$ to $\boldsymbol{x}$. In (15), the "optimal" values of $\alpha_i(t)$ are determined by recursion as:

$$\alpha_c(t) = \frac{\alpha_c}{1 + s(t)\alpha_c(t-1)} \tag{2}$$

where $s(t) = +1$ if the classification of the prototype p_c is correct and $s(t) = -1$ if the classification is wrong. Therefore, it is necessary to stop the learning process after some "optimal" number of steps. Typically, the *OLVQ1* may be stopped after 200 iterations.

2.4 MixtGauss

MixtGauss is an adaptive condensing algorithm proposed by some of us in (17). It is considered in the framework of mixture modelling by Gaussian distributions, while assuming a statistical independence of features. The prototypes are chosen as the mean vectors of the optimised Gaussians, whose mixtures are fit to model each of the classes. The details are given below.

In general, given a TS, we assume that each class follows a spatial distribution according to its class-conditional probability density function (*pdf*) $P(\boldsymbol{x}|c_j)$ and the respective *a priori* probability $P(c_j)$, $c_j \in C$. These class-conditional *pdf*s do not need to have a specific structure, in general. In practice, however, it is necessary to obtain their density estimations.

A natural way to deal with a density estimator is to consider a mixture density of modes. One approach to retain the capacity of $P(\boldsymbol{x}|c_j)$ is by reflecting the local structure of the distribution by means of mixture modes $P_m(\boldsymbol{x}|c_j)$, where

each mode is estimated by the product of probabilities in each feature as (21):

$$P(\boldsymbol{x}|c_j) = \sum_{m=1}^{M} \alpha_{m|j} P_m(\boldsymbol{x}|c_j) = \sum_{m=1}^{M} \alpha_{m|j} \prod_{k=1}^{n} N(x_k; \mu_{m|kj}, \sigma_{m|kj}) \qquad (3)$$

where M is the number of modes and $\alpha_{m|j}$ is *a priori* probability of the m-th mode in the class c_j.

In our condensing algorithm, each class in the TS is modelled by a probability distribution. A general shape of classes as well as the decision boundaries is maintained, when each class is described by a mixture of multivariate Gaussian distributions. By our additional assumption of the statistical independence of features, the m-th component in a mixture modelling the class c_j is a multivariate Gaussian distribution expressed by a product of univariate normal distributions, $N(\mu_{m|kj}, \sigma_{m|kj})$. Note that this is equivalent to a multivariate elliptic Gaussian distribution $N(\boldsymbol{\mu}_{m|j}, \mathrm{diag}(\sigma_{m|j})$. The standard method used to fit finite mixture models to the observed data, hence to estimate the parameters of the class distributions, is the well-known EM algorithm (18; 19). This is a general maximum likelihood optimisation procedure for problems with hidden variables or missing data (8). The final prototypes of the sought condensed set are the mean vectors of the Gaussian distributions determined by the EM algorithm.

Note that, although the EM algorithm is widely used, one needs to be aware of its drawbacks (12). As a method working in local neighbourhoods, it is sensitive to initialisation, as the likelihood function of a mixture model is not unimodal. Another important problem in mixture modelling is the selection of the number of components. With too many components, the mixture may overfit the data, while a mixture with too few components may not be flexible enough to approximate the true underlying model.

2.4.1 Algorithm details

The more Gaussian components are included in a mixture, the more accurate the representation of the classes and the decision boundaries is. Taking this into account, M Gaussians have to be specified to represent each class distribution. This number corresponds to the number of prototypes generated in each class for the final condensed set. Initially, each class is represented by M Gaussians located at the mean of that class. By adding random disturbances, different mean vectors are created and the Gaussians are shifted away. So, a mixture of Gaussians can be obtained per class. For each Gaussian, the initial variance is set up to 1/10 of the range in each dimension.

After the initialisation stage, the EM algorithm is used to determine an optimal location of the mixture of Gaussians. This iterative optimisation proce-

Algorithm 3. *MixtGauss*

```
(* initialisation *)
for c = 1..number_of_classes do
  mean[c] = Calculate_Mean(c, TS)
  for g = 1..number_of_Gaussians_per_class do
    Gaussians[c, g] = mean[c] + Random_Disturbance()
  end for
end for
(* optimisation *)
repeat
  previous_Gaussians[c, g] = Gaussians[c, g]
  Gaussians[c, g] = EMstep()
  previous_accuracy = current_accuracy
  current_accuracy = Calculate_Accuracy()
  classes_improve = 0
  for c = 1..number_of_classes do
    if current_accuracy[c] > previous_accuracy[c] then
      classes_improve = classes_improve + 1
    else
      if current_accuracy[c]! = previous_accuracy[c] then
        for g = 1..number_of_Gaussians_per_class do
          Gaussians[c, g] = previous_Gaussians[c, g]
        end for
      end if
    end if
  end for
until classes_improve == 0
```

dure converges to a *maximum likelihood* estimate of the mixture parameters. Accordingly, to fit a mixture of Gaussians to each class, we iterate between the following two steps:

E-step Compute the contributions of the prototypes $\boldsymbol{x}^t$ in the class c_j, belonging to the set $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{N_j}\}$, where N_j is the cardinality of the class c_j. The conditional *pdf* for the m-th mode is:

$$P_m(\boldsymbol{x}^t|c_j) = \frac{\alpha_{m|j} \prod_{k=1}^{n} N(x_k^t; \mu_{m|kj}, \sigma_{m|kj})}{\sum_{l=1}^{M} \alpha_{l|j} \prod_{k=1}^{n} N(x_k^t; \mu_{l|kj}, \sigma_{l|kj})}, \tag{4}$$

and it is normalised such that $\sum_{m=1}^{M} \alpha_{m|j} = 1$. The multivariate Gaussians are represented as a product of univariate normal distributions, with the means $\mu_{m|kj}$ and the standard deviations $\sigma_{m|kj}$.

M-step Compute the parameters of the m-th mode for each value $\mathbf{x^t}$ that exists in the class c_j:

$$\alpha_{m|j} = \frac{1}{N_j} \sum_{t=1}^{N_j} P_m(\boldsymbol{x}^t|c_j), \qquad \mu_{m|kj} = \frac{\sum_{t=1}^{N_j} P_m(\boldsymbol{x}^t|c_j) x_k^t}{\sum_{t=1}^{N_j} P_m(\boldsymbol{x}^t|c_j)}, \tag{5}$$

$$\sigma_{m|kj} = \frac{\sum_{t=1}^{N_j} P_m(\boldsymbol{x}^t|c_j)(x_k^t - \mu_{m|kj})^2}{\sum_{t=1}^{N_j} P_m(\boldsymbol{x}^t|c_j)}. \quad (6)$$

The EM iterative optimisation can cause an overlap between the Gaussian components from different classes, which will deteriorate the final classification accuracy. Therefore, this process should stop when no class yields an increase in performance with respect to the previous iteration. To address this problem, the consistency criterion is applied. After each iteration, the consistency criterion is estimated by calculating the 1-NN error over the TS by using the estimated means of the Gaussians as the current condensed set. The movement of the Gaussians is carried out while any class improves the classification rate with respect to the previous step. The complete technique is summarised in Algorithm 3.

3 Prototype-based classification methods to compare

The usefulness of condensed sets, optimised by the approaches discussed above, will be evaluated in a classification task. Traditionally, the 1-NN rule, assigning an unknown object to the class of its nearest neighbour in the condensed set, is used for this purpose. The 1-NN rule is employed, as it is often used in condensing schemes for the design of the condensed set. If the classes are represented as compact Gaussian-like clouds of similar spreads, then the 1-NN is expected to generalise well for a small condensed set. Otherwise, a large condensed set might be needed to represent the variability in the data. Alternatively, classifiers in the so-called dissimilarity spaces can be considered.

3.1 Dissimilarity spaces

Remember that a TS X consist of N objects in a feature vector space. Let $R = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_r\}$ be a condensed set, called also representation set, of r optimised prototypes. Given a dissimilarity measure [2] d, one may consider a new representation based on the proximities to the set R. Every object $\boldsymbol{x} \in X$ is then described by a vector of dissimilarities computed between $\boldsymbol{x}$ and the prototypes from R, i.e. as $D(\boldsymbol{x}, R) = [d(\boldsymbol{x}, \boldsymbol{p}_1), d(\boldsymbol{x}, \boldsymbol{p}_2), \ldots, d(\boldsymbol{x}, \boldsymbol{p}_r)]$. Hence, for a set X, it extends to an $N \times r$ dissimilarity matrix $D(X, R)$.

[2] In general, a nonnegative dissimilarity measure should express a degree of commonality between pairs of objects. It should be zero for two identical objects, take small values for similar objects and large values for objects that differ. It does not need to be metric.

The dissimilarity matrix $D(X, R)$ is interpreted as a data-dependent mapping $D(\cdot, R) : X \to \mathbb{R}^r$ from the representation X to a *dissimilarity space*, defined by the set R. This is a vector space, in which each dimension corresponds to a dissimilarity to a prototype from R, $D(\cdot, \boldsymbol{p}_i)$. The advantage of this representation is that any traditional classifier operating in vector spaces can now be used (24; 27; 28).

A vector $D(\cdot, \boldsymbol{p}_i)$ of dissimilarities to the prototype $\boldsymbol{p}_i$ can be interpreted as a feature. This follows since the dissimilarities to $\boldsymbol{p}_i$ will differ depending on the class membership. If the measure is metric and the dissimilarity $d(\boldsymbol{p}_i, \boldsymbol{p}_j)$ is small, then $d(\boldsymbol{x}, \boldsymbol{p}_i) \approx d(\boldsymbol{x}, \boldsymbol{p}_j)$ for other object $\boldsymbol{x}$, as it is guaranteed by the backward triangle inequality (28). As a result, only one of them can be chosen as a prototype. However, a small number of prototypes might be insufficient to represent the data variability if the classes have different spreads (in terms of the dissimilarities). This occurs when, in a feature space one class is represented by a compact cloud of points, while the other class is elongated. It can be, therefore, more beneficial to inspect the vectors of dissimilarities to the entire set R, instead of looking at the nearest neighbour only. If the objects $\boldsymbol{x}$ and $\boldsymbol{y}$ are similar, then their dissimilarity vectors $D(\boldsymbol{x}, R)$ and $D(\boldsymbol{y}, R)$ are expected to be correlated. Hence they should lie close in a dissimilarity space (as measured e.g. by an Euclidean distance in this space). A linear or quadratic classifier in this dissimilarity space might be of interest.

3.2 Normal-density based classifiers in dissimilarity spaces

In case of small condensed sets or non-representative TS, a better generalisation can be achieved by a classifier built in a dissimilarity space than by the 1-NN rule. Many traditional classifiers can be applied there (22; 24; 25; 28). Linear and quadratic functions are weighted linear (quadratic) combinations of the dissimilarities $d(\boldsymbol{x}, \boldsymbol{p}_i)$ between a given object x and the prototypes $\boldsymbol{p}_i$. Although the classifiers are trained on $D(\cdot, R)$, the weights are still optimised on the complete TS.

Bayesian classifiers, i.e. linear and quadratic normal density based classifiers, tend to perform well in dissimilarity spaces (22; 23; 24; 25; 26). This is especially true, for a summation-based dissimilarity measure, summing over a number of components with similar variances. The reason is that such dissimilarities will be approximately normally distributed thanks to the central limit theorem (if one or few variances are dominant, then they will approximate the χ^2 distribution) (28). In our case, the vectorial data will be normalised to unit variances, hence the resulting Euclidean distances tend to be approximately normally distributed. One may, therefore, expect a good performance of such decision rules.

For a two-class problem, a linear normal density based classifier (NLC), defined by the condensed set R, is given by:

$$f\left(D(\boldsymbol{x},R)\right) = [D(\boldsymbol{x},R) - \frac{1}{2}(\boldsymbol{m}_{(1)} + \boldsymbol{m}_{(2)})]^T\, S^{-1}\left(\boldsymbol{m}_{(1)} - \boldsymbol{m}_{(2)}\right) + \log\frac{p_{(1)}}{p_{(2)}}, \quad (7)$$

and the quadratic function (NQC) becomes:

$$\begin{aligned} f(D(\boldsymbol{x},R)) = \sum_{i=1}^{2} \frac{(-1)^i}{2}\left(D(\boldsymbol{x},R) - \boldsymbol{m}_{(i)}\right)^T S_{(i)}^{-1}\left(D(\boldsymbol{x},R) - \boldsymbol{m}_{(i)}\right) \\ + \log\frac{p_{(1)}}{p_{(2)}} + \frac{1}{2}\log\frac{|S_{(1)}|}{|S_{(2)}|}, \end{aligned} \quad (8)$$

where $\boldsymbol{m}_{(1)}$ and $\boldsymbol{m}_{(2)}$ are the mean vectors, S is the sample covariance matrix (average of the class covariance matrices) and $S_{(1)}$ and $S_{(2)}$ are the estimated class covariance matrices, all computed in the dissimilarity space $D(X,R)$. $p_{(1)}$ and $p_{(2)}$ are the class prior probabilities. When the covariance matrices become singular, they can be regularised. In the implementation of these classifiers for multi-class problems, the normal-density functions are estimated per each class and the final decision is based on the maximum a posteriori probability (11).

Given equal class priors, the NLC is equivalent to the Fisher linear discriminant (FLD) obtained by maximising the *Fisher criterion*, i.e. $\max_{\boldsymbol{w}} \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}}$, where S_B is the between-class scatter and $S_W = S$ is the within-class scatter; see (10) for details. For a representation $D(T,R)$, the FLD is found as

$$f(D(\boldsymbol{x},R)) = (\boldsymbol{m}_1 - \boldsymbol{m}_2)^T S_W^{-1}\, D(\boldsymbol{x},R) - \frac{1}{2}\left(\boldsymbol{m}_1 + \boldsymbol{m}_2\right)^T S_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2) \quad (9)$$

A multi-class FLD is derived in the one-against-all strategy (11).

4 Experimental Results and Discussion

We will compare some methods of determining a condensed set R in combination with three prototype-based classification strategies. These is done on Euclidean distance representations $D(X,R)$ derived in feature spaces. The four condensing algorithms are the ones described before: *MaxNCN*, *Reconsistent*, *LVQ* and *MixtGauss*. The first classification method is the 1-NN rule based on the set R. The other two methods are the FLD and NQC trained in dissimilarity spaces $D(X,R)$.

4.1 Data Description and Experimental Setup

Eleven real data sets (see Table 1) are taken from the UCI Repository (20) to assess the behaviour of the algorithms presented here. All data sets are first normalised by a unit variance, and then the Euclidean distances are computed. The results are compared in terms of both accuracy and reduction of the TS size.

Table 1
Data sets used in the experiments.

Data set	No. classes	No. features	TS size	Test set size
Cancer	2	9	546	137
Diabetes	2	8	614	154
Glass	6	9	171	43
Heart	2	13	216	54
Liver	2	6	276	69
Vehicle	4	18	677	169
Vowel	11	10	422	106
Wine	3	13	142	36
Phoneme	2	5	4323	1080
Satimage	6	36	5148	1287
Texture	11	40	4400	1100

The *MaxNCN, Reconsistent* and *MixtGauss* algorithms need to be applied to overlap-free (no overlap between different class regions) data sets. Therefore, as a general rule, and according to the previously published results (2; 33), the Wilson's editing is considered to properly remove a possible overlap between the classes. In our experiments, the parameter k involved (of the k-NN rule) is obtained by performing a five-fold cross-validation experiment using only the TS and computing the average classification accuracies for different values of k. The best edited set (including the non-edited TS) is thus selected as the input for the different condensing schemes.

4.2 Quantitative Results

Tables 2, 3 and 4 report the classification accuracy of the *1-NN*, the *FLD* and the *NQC*, respectively, obtained by using different condensed sets. For each condensing method, the average performance values, computed over the

Table 2
Classification accuracy of the 1-NN (in %). The corresponding reduction rates of the TS size (in %) and the equivalent number of prototypes are given in brackets.

	Ori.	MaxNCN	Reconsistent	LVQ	MixtGauss
Canc.	95.2	92.0 [95.2≡26]	93.4 [93.8≡34]	96.9 [99.3≡4]	96.5 [99.3≡4]
Diab.	70.1	67.7 [87.0≡80]	71.8 [81.8≡112]	76.6 [98.1≡12]	75.0 [91.9≡50]
Glass	69.7	64.1 [59.1≡70]	69.4 [46.8≡91]	72.6 [71.9≡48]	64.5 [75.4≡42]
Heart	76.3	69.3 [83.8≡35]	74.8 [77.8≡48]	84.1 [89.8≡22]	81.9 [93.5≡14]
Liver	62.6	58.9 [76.8≡64]	61.0 [68.5≡87]	65.6 [93.5≡18]	63.7 [97.1≡8]
Vehi.	69.4	64.4 [60.3≡269]	66.0 [49.2≡344]	72.2 [61.6≡260]	69.6 [70.5≡200]
Vowel	98.1	90.8 [74.4≡108]	95.5 [71.8≡119]	80.8 [76.5≡99]	90.7 [76.5≡99]
Wine	94.4	91.6 [93.0≡10]	92.1 [90.9≡13]	95.5 [95.8≡6]	96.0 [97.9≡3]
Phon.	71.0	69.1 [91.6≡365]	70.2 [88.9≡480]	77.3 [93.1≡300]	73.3 [93.1≡300]
Sati.	83.0	79.2 [86.5≡693]	80.3 [82.7≡890]	82.2 [88.3≡600]	81.3 [94.8≡270]
Text.	98.9	95.3 [89.3≡470]	97.3 [87.8≡535]	85.9 [99.5≡22]	97.2 [90.0≡440]
Avr	80.8	76.6 [81.5]	79.2 [76.4]	80.9 [87.9]	80.9 [89.1]

eleven data sets are also included. There are no results of the FLD and the NQC for the *Satimage* and *Texture* sets (for the original TS), because of their high memory requirements during training (11). The average performances excluding these data sets are also presented. Since the adaptive techniques, the *MixtGauss* and the *LVQ*, allow one to determine a condensed set of any size, only a few results are shown in the tables. These correspond to the best accuracy values, found in the sets between one prototype per class and the size determined by the MaxNCN algorithm (as the *MaxNCN* condensed set is always smaller than the *Reconsistent* condensed set).

The first observation is that the *LVQ* and the *MixtGauss* increase the average accuracy in comparison to the original TS accuracy. The increase is larger by using the *FLD* and the *NQC* than by using the *1-NN* rule (where the increase is quite small). Anyway, it is significant that in all these cases, the performance is increased despite the high reduction of the TS (see Tables 2 – 4). Remember that although the *FLD* and the *NQC* are based on the distances to the prototypes, they still make use of the entire TS in training. This is the reason why the classification accuracy increases so much when using any condensed set, in comparison to the results by using the original TS.

By using the *FLD*, the classification accuracy obtained for the *MaxNCN* and *Reconsistent* procedures is very similar. The *MixtGauss* leads to a higher ac-

Table 3
Classification accuracy of the FLD in dissimilarity spaces and the corresponding reduction rates of the TS size in brackets. Both values are expressed in %. Averages over all data sets (1-11) and data sets excluding Satimage and Texture, (1-9), are also given.

	Original	MaxNCN	Reconsistent	LVQ	MixtGauss
1 Cancer	85.50	96.48 [95.2]	96.48 [93.8]	97.22 [98.2]	96.92 [96.3]
2 Diabetes	74.23	76.05 [87.0]	77.09 [81.8]	77.08 [98.7]	76.70 [87.0]
3 Glass	53.88	66.53 [59.1]	67.49 [46.8]	77.17 [61.4]	69.77 [93.0]
4 Heart	60.75	80.72 [83.8]	78.49 [77.8]	84.79 [98.1]	82.60 [98.2]
5 Liver	68.35	68.96 [76.8]	67.83 [68.5]	70.70 [92.8]	68.67 [93.5]
6 Vehicle	77.19	79.31 [60.3]	79.79 [49.2]	80.96 [67.5]	80.02 [76.4]
7 Vowel	41.03	95.76 [74.4]	96.67 [71.8]	91.42 [76.5]	94.28 [76.5]
8 Wine	33.15	98.28 [93.0]	98.30 [90.9]	98.87 [95.8]	98.90 [95.8]
9 Phoneme	70.62	69.75 [91.6]	70.52 [88.9]	77.11 [98.2]	73.94 [99.8]
10 Satimage	****	24.49 [86.5]	23.90 [82.7]	82.11 [95.3]	82.20 [95.9]
11 Texture	****	37.33 [89.3]	27.56 [87.8]	99.00 [92.5]	99.13 [93.8]
Avr (1-11)	****	72.2 [81.5]	71.3 [76.4]	85.1 [88.6]	83.9 [91.5]
Avr (1-9)	62.7	81.3	81.4	83.9	82.4

curacy than both of them. The best performance of the FLD relies on the *LVQ*. A similar pattern can be observed for the *NQC*. This differs, however, for the *1-NN* rule, especially in the case of the *Reconsistent* algorithm. The smallest classification accuracy is reached for the *MaxNCN*. The *Reconsistent* gives better results, and the best values are obtained for the *LVQ* and the *MixtGauss*.

The training reduction rates, being the percentage by which the original training size is reduced, are provided in square brackets in Tables 2 – 4. Since the *MaxNCN* and the *Reconsistent* techniques determine the optimal number of prototypes directly, their reduction rates are identical for the three classification methods used. Additionally, for each condensing method, the average reduction values computed over the eleven data sets are also included.

In relation to the training reduction rates, the average is higher for the *MaxNCN* than for the *Reconsistent*. This is to be expected as in the second algorithm, some instances are added to the MaxNCN condensed set to improve the classification accuracy. It can also be observed that the average TS reduction rates are higher for the adaptive *LVQ* and *MixtGauss* techniques than for the selective *MaxNCN* and *Reconsistent* procedures. The highest reduction rate is

Table 4
Classification accuracy of the NQC in dissimilarity spaces and the corresponding reduction rates of the TS size in brackets. Both values are expressed in %. Averages over all data sets (1-11) and data sets excluding Satimage and Texture, (1-9), are also given.

	Original	MaxNCN	Reconsistent	LVQ	MixtGauss
1 Cancer	80.23	96.19 [95.2]	96.48 [93.8]	97.22 [96.3]	96.34 [96.3]
2 Diabetes	65.62	75.39 [87.0]	74.74 [81.8]	77.09 [97.1]	76.70 [98.7]
3 Glass	40.15	53.81 [59.1]	54.40 [46.8]	65.10 [89.5]	64.11 [82.5]
4 Heart	55.56	82.97 [83.8]	80.00 [77.8]	82.95 [97.2]	82.21 [88.9]
5 Liver	46.67	62.63 [76.8]	60.00 [68.5]	67.79 [85.5]	65.20 [89.1]
6 Vehicle	27.78	60.27 [60.3]	54.37 [49.2]	74.81 [76.4]	74.58 [91.1]
7 Vowel	10.20	31.96 [74.4]	29.41 [71.8]	94.59 [92.2]	94.10 [92.2]
8 Wine	33.15	97.73 [93.0]	96.05 [90.9]	97.76 [97.9]	99.43 [95.8]
9 Phoneme	73.26	73.08 [91.6]	73.22 [88.9]	76.11 [95.4]	73.33 [95.4]
10 Satimage	****	79.40 [86.5]	79.17 [82.7]	81.93 [98.3]	81.62 [97.1]
11 Texture	****	96.31 [89.3]	96.51 [87.8]	97.58 [95.0]	97.53 [96.3]
Avr (1-11)	****	73.6 [81.5]	72.2 [76.4]	83.0 [92.8]	82.3 [93.0]
Avr (1-9)	48.1	70.5	68.7	81.5	80.7

the one of *MixtGauss*. This holds for the three prototype-based classification strategies used here.

In relation to the classification methods considered, in general, we can observe that the *FLD* and the *NQC* perform better than the *1-NN* rule, when they rely on the condensed sets determined by the adaptive schemes. Comparing both algorithms, the one with the best performance corresponds to the *FLD* On the other hand, the *MaxNCN* and the *Reconsistent* procedures have their best classification accuracy with the *1-NN* rule. As designed, the *Reconsistent* leads to a higher nearest neighbour performance than the *MaxNCN*.

These results can be better analysed by studying the graphs for each data set (see an example for four data sets in Figure 2). The best trade-off results (between the accuracy and the reduced TS size) are the ones for the *FLD*. The results for the *NQC* are similar, and the ones for the *1-NN* rule are significantly worse. In relation to the condensing algorithms, the *LVQ* and the *MixtGauss* lead to very similar results. In general, the points corresponding to the *MaxNCN-FLD* and the *Reconsistent-FLD* results are near the functions representing the *MixtGauss-FLD* and the *LVQ-FLD* results, similarly as the points representing the *MaxNCN-NQC* and the *Reconsistent-NQC* results are

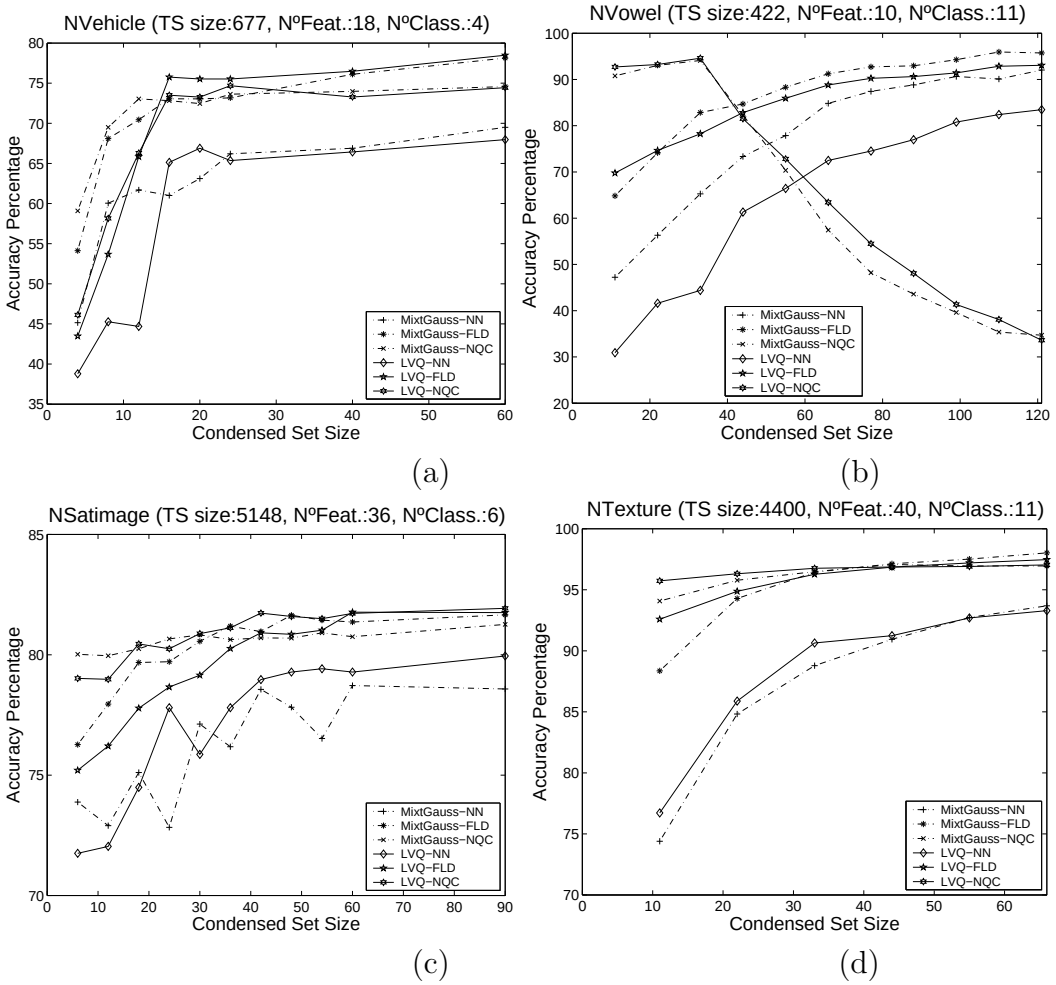

Figure 2. Trade-off between the resulting condensed set sizes and classification accuracy for different algorithms. The following data sets are shown: (a) *Vehicle*, (b) *Vowel*, (c) *Satimage*, (d) and *Texture*.

close to the points describing the *MixtGauss-NQC* and the *LVQ-NQC* results. This observation does not hold for the *MaxNCN-NN* and the *Reconsistent-NN*, neither the *MixtGauss-NN* and the *LVQ-NN*, as there exists a big difference in favour of the *MixtGauss* and the *LVQ* for some data sets.

In Figure 2, the behaviour of the algorithms investigated here is shown for four data sets. Only the results for the smallest reduced sets for each data are plotted, in order to distinguish the plots. The first graph, Figure 2(a), refers to the *Vehicle* data set. It can be observed there that the best classification accuracy is reached by the *MixtGauss-NQC* for the smallest condensed sets. For sets with more than three instances per class, the best accuracy is reached by the *LVQ-FLD*. It is to note that the performance of the *MixtGauss-FLD* is always close to the best case.

In Figure 2(b) the results for the *Vowel* data set are shown. The three smallest

condensed sets represented (1, 2 and 3 instances per class) lead to the best performance for the combination of the *LVQ-NQC* and the *MixtGauss-NQC* algorithms. However, for larger sizes, the classification accuracy suddenly decreases. It is due to a large number of classes in relation to the number of prototypes, which effectively translates into the dimension of the dissimilarity space. As a result, the number of instances per class is too small, and as the size of the condensed set increases, class covariance matrices are badly estimated. Hence, the classification accuracy is diminished. The next best results belong to the *MixtGauss-FLD*, the *LVQ-FLD* and the *MixtGauss-NN*, in that order.

In Figure 2(c), the results for the *Satimage* data set are shown. The most reduced sets lead to a better performance by using the *MixtGauss-NQC*, the *LVQ-NQC*, the *MixtGauss-FLD* and the *LVQ-FLD* methods. The results for the data set *Texture* are shown in Figure 2(d). The best ones are obtained by the following combinations: the *LVQ-NQC*, the *MixtGauss-NQC*, the *LVQ-FLD* and the *MixtGauss-FLD*. As it is not plotted in this paper, in order to save space, it could be important to know the next point. The larger graphs for *Satimage* and *Texture* data sets show that the classification accuracy also decreases around the size of 300 instances for the *MixtGauss-FLD* and the *LVQ-FLD*. It is due to similar reasons as mentioned for the *Vowel* data.

5 Conclusions

In this paper, several reduction techniques, including both selecting and generating schemes, were compared in connection to the classifier based on the obtained set of prototypes. Three classifiers were used: the 1-NN rule, and linear and quadratic prototype-based classifiers defined in a dissimilarity space. The purpose of this experimental study was to discuss these prototype optimisation methods with respect to their ability of maintaining small prototype sets and high classification performance.

From our investigation it can be concluded that there is no significant difference between the *LVQ* and the *MixtGauss* prototype generation techniques. However, when comparing them to the *MaxNCN* and *Reconsistent* methods, the first two algorithms tend to give both larger reduction of the TS (i.e. smaller condensed sets) and lead to higher classification accuracy. This means that the *LVQ* and the *MixtGauss* are good as condensing algorithms since they yield a good performance and allow one to control the number of prototypes. Additionally, they represent a good trade-off between the classification accuracy and the reduction rates.

In relation to the prototype-based decision rules, in general, the classification

accuracy obtained by the *FLD* tends to be higher than that of the *NQC*, both higher than the accuracy of the *1-NN* rule, independently of the considered condensing algorithms. This is striking as the prototype sets are often optimised to guarantee that the 1-NN rule performs well. The reason is likely that the *FLD* and the *NQC*, although based on the condensed sets, still make use of the entire TS for determining the decision boundary. The *1-NN* rule, when applied as a nearest prototype rule, discards other training objects.

What is important, is that for the evaluation of new objects, the computational complexity of the *FLD* is $O(Jr)$, which is similar to that of the *1-NN* rule, $O(r)$, provided that J, the number of classes, is small. For a large number of classes, the computational cost increases. The FLD may, therefore, be an alternative to the 1-NN rule for the prototype-based classification. To our knowledge, it has not been used among the researchers studying the condensing techniques, yet.

The *NQC* is much more computationally heavy, so it can only be advantageous for small condensed sets and a small number of classes. In fact, the NQC was used here, to show that a more complex classifier than a linear one is not necessarily yielding a better performance in a dissimilarity space. As such our study opens a new possibility of using linear classifiers in dissimilarity spaces instead of the *1-NN* rule, both defined by the same optimised prototype sets. Although the FLD was studied here, other linear functions can be used, such as a logistic classifier or a hyperplane defined by a linear programming procedure. Their use is open for further research.

Our final conclusion is that the adaptive techniques, the *LVQ* and the *MixtGauss*, combined with the FLD, offer the best trade-off between the reduction rate, computational cost and the classification performance.

References

[1] D.W. Aha and D. Kibler and M.K.Albert, Instance-Based Learning Algorithms, *Machine Learning* **6**(1), (1991) 37–66.

[2] M.C. Ainslie and J.S. Sánchez, Space Partitioning for Instance Reduction in Lazy Learning Algorithms, *2nd Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning* (2002) 13–18.

[3] C.L. Chang, Finding Prototypes for Nearest Neighbor Classifiers, *IEEE Trans. on Computers*, **23** (1974) 1179–1184.

[4] B.B. Chaudhuri, A New Definition of Neighbourhood of a Point in Multi-Dimensional Space, *Pattern Recognition Letters*, **17** (1996) 11–17.

[5] C.H. Chen and A. Józwik, A Sample Set Condensation Algorithm for the

Class Sensitive Artificial Neural Network, *Pattern Recognition Letters*, **17** (1996) 819–823.

[6] B.V. Dasarathy, Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques, *IEEE Computer Society Press* (Los Alamitos, CA, 1990).

[7] B.V. Dasarathy, Minimal Consistent Subset (MCS) Identification for Optimal Nearest Neighbor Decision Systems Design, *IEEE Trans. on Systems, Man and Cybernetics* **24** (1994) 511–517.

[8] A.P. Dempster, N.M. Laird and D.B. Rubin, Maximum Likelihood from Incomplete Data Via the EM Algorithm, *Journal of the Royal Statistical Society (B).* **39** (1977) 1–38.

[9] P.A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, (Prentice Hall, Englewood Cliffs, N.J., 1982).

[10] R.O. Duda, P.E. Hart and D.G. Stork, *Pattern Classification, 2nd ed.* (John Wiley & Sons, Inc., 2001).

[11] R.P.W. Duin, P.Juszczak, D. de Ridder, P. Paclík, E. Pękalska and D.M.J. Tax, *PR-Tools, a Matlab Toolbox for Pattern Recognition* (2004). http://www.prtools.org.

[12] M.A.T. Figueiredo and A.K. Jain, Unsupervised Learning of Finite Mixture Models, *IEEE Trans. on Pattern Analysis Machine Intelligence* **24** (2002) 381–396.

[13] P.E. Hart, The Condensed Nearest Neighbor Rule, *IEEE Trans. on Information Theory* **14** no. 5 (1968) 505–516.

[14] T. Kohonen, *Self-Organizing Maps*, (Springer-Verlag, 1995).

[15] T. Kohonen, J. Hynninen, J. Kangas, J. Laarksonen and K. Torkkola, *LVQ_PAK: The Learning Vector Quantization Program Package*, Helsinki University of Technology (1996) http://www.cis.hut.fi/research/som-research/nnrc-programs.shtml.

[16] M. Lozano, J.S., Sánchez, F. Pla, Using the geometrical distribution of prototypes for training set condensing, Current Topics in Artificial Intelligence, Lecture Notes in Artificial Intelligence, Vol. 3040, R. Conejo, M. Urretavizcaya, J.L. Pérez de la Cruz (Eds.), Springer-Verlag, (2004), 618-627.

[17] M. Lozano, J.M. Sotoca, J.S. Sánchez, F. Pla, An adaptive condensing algorithm based on mixtures of Gaussians, Set Congres Catal d'Intel.ligencia Artificial, Barcelona (Spain), Frontiers in Artificial Intelligence and Applications, Vol. 113, IOS Press, J. Vitrià J. et al. (Eds.), (2004), 225-232.

[18] G. McLachlan and K. Basford, *Mixture Models: Inference and Application to Clustering* (Marcel Dekker, New York, 1988).

[19] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions* (John Wiley & Sons, New York, 1997).

[20] C.J. Merz and P.M. Murphy, *UCI Repository of Machine Learning DB*, Dept. Information and Comp. Sci., U. California (1998). http://www.ics.uci.edu/˜mlearn.

[21] J. Novovičová and P. Pudil and J. Kittler, Divergence Based Feature Selection for Multimodal Class Densities, *IEEE Trans. on Pattern Analysis*

and Machine Intelligence. **18** (1996) 218–223.

[22] P. Paclík and R.P.W. Duin, Dissimilarity-Based Classification of Spectra: Computational Issues, *Real Time Imaging* **9** no. 4 (2003) 237–244.

[23] P. Paclík and R.P.W. Duin, Classifying Spectral Data using Relational Representation, *Spectral Imaging Workshop*, Graz, Austria (2003).

[24] E. Pękalska and R.P.W. Duin, Dissimilarity Representations Allow for building good Classifiers, *Pattern Recognition Letters* **23** no. 8 (2002) 943–956.

[25] E. Pękalska, P. Paclík and R.P.W. Duin, A Generalized Kernel Approach to Dissimilarity Based Classification, *Journal of Machine Learning Research*, **2** no. 2 (2002) 175–211.

[26] E. Pękalska, R.P.W. Duin, S. Günter and H. Bunke, On Not Making Dissimilarities Euclidean, *Structural Syntactic and Statistical Pattern Recognition, Lecture Notes in computer Science* **3138** (Springer Verlag, Berlin, 2004) 1145–1154.

[27] E. Pękalska, R.P.W. Duin and P. Paclík, Prototype Selection for Dissimilarity-based Classifiers, to appear in *Pattern Recognition* (2005).

[28] E. Pękalska and R.P.W. Duin, The Dissimilarity Representation for Pattern Recognition. Foundations and Applications, (World Scientific, Singapore, 2005).

[29] J.S. Sánchez, F. Pla and F.J. Ferri, On the Use of Neighbourhood-Based Non-Parametric Classifiers, *Pattern Recognition Letters* **18** (1997) 1179-1186.

[30] I. Tomek, *Two Modifications of CNN*, *IEEE Trans. on Systems, Man and Cybernetics* **6** (1976) 769-772.

[31] G.T. Toussaint and B.K. Bhattacharya and R.S. Poulsen, The Application of Voronoi Diagrams to Nonparametric Decision Rules, *Computer Science and Statistics: The Interface* (L. Billard, Elsevier Science, North-Holland, Amsterdam, 1985).

[32] D.L. Wilson Asymptotic Properties of Nearest Neighbor Rules Using Edited Data Sets, *IEEE Trans. on Systems, Man and Cybernetics* **2** (1972) 408–421.

[33] D.R. Wilson and T.R. Martinez, Reduction Techniques for Instance-Based Learning Algorithms, *Machine Learning*. **38** no. 3 (2000) 257–286.