PERGAMON

Pattern Recognition 35 (2002) 2771–2782

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

An efficient prototype merging strategy for the condensed 1-NN rule through class-conditional hierarchical clustering ☆

R.A. Mollineda[a], F.J. Ferri[b,*], E. Vidal[a]

[a] *Univ. Politècnica de València, Instituto Tecnològico de Informàtica, Camino de Vera s/n, 46071 València, Spain*
[b] *Dept. d'Informàtica, Universitat de València, Dr Moliner 50, 46100 Burjassot, Spain*

Received 3 November 2000; accepted 19 October 2001

Abstract

A generalized prototype-based classification scheme founded on hierarchical clustering is proposed. The basic idea is to obtain a condensed 1-NN classification rule by merging the two same-class nearest clusters, provided that the set of cluster representatives correctly classifies all the original points. Apart from the quality of the obtained sets and its flexibility which comes from the fact that different intercluster measures and criteria can be used, the proposed scheme includes a very efficient four-stage procedure which conveniently exploits geometric cluster properties to decide about each possible merge. Empirical results demonstrate the merits of the proposed algorithm taking into account the size of the condensed sets of prototypes, the accuracy of the corresponding condensed 1-NN classification rule and the computing time. © 2002 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Nearest neighbor; Prototype merging; Class-conditional hierarchical clustering; Cluster-based condensing

1. Introduction

Given a training set of previously labeled samples and an unknown sample x, the k-nearest neighbor (k-NN) rule assigns the most frequently represented class-label among the k closest prototypes to x. Over the last 40 years, this simple classification rule has been intensively used in a broad range of pattern recognition applications. In contrast to its conceptual simplicity, the rule has a good behavior when applied to non-trivial problems. In fact, the k-NN rule is asymptotically optimal in the Bayes sense [1]. In other words, the k-NN rule performs as well as any other possible classifier, provided there is an arbitrarily large number of (representative) prototypes available and the volume of the k-neighborhood of x is arbitrarily close to zero for all x.

From the point of view of its implementation, the k-NN rule consists of a search of prototypes given a particular distance definition. A trivial consequence of the large size of the sets of prototypes (to guarantee representativity and approach optimality) is the computational burden this searching problem implies. This constitutes one of the main drawbacks of the NN rules. Another very important drawback comes from the fact that the prototypes which are available may contain erroneously labeled or noisy prototypes which may lead to arbitrarily large deviations from the asymptotically optimal results that could be expected.

Among many other approaches for solving either the computational problem and/or the performance problem, *Prototype Selection* aims at modifying an initially given set of prototypes in order to reduce its size as well as to improve classification performance.

From the point of view of their goal, prototype selection methods can be divided into two different kinds of techniques which are usually referred to as *editing* and

☆ This work has been partially supported by Spanish projects TIC98-677-C02-02, 1FD97-279, TIC2000-1703-C03-03 and a grant from the Agencia Española de Cooperación Internacional.

* Corresponding author. Tel.: +34-96-3160-414; fax: +34-96-3160-418.

E-mail addresses: rmollin@iti.upv.es (R.A. Mollineda), ferri@uv.es (F.J. Ferri).

PII: S0031-3203(01)00208-4

condensing [2]. The first one aims at removing outliers and prototypes which are placed at the overlap among classes. Editing does not generally entail substantial reductions in size, but it usually produces well-clustered groups of homogeneous prototypes that lead to optimal 1-NN classification results. On the other hand, condensing algorithms try to find a significantly reduced set of prototypes whose 1-NN classification results are as close as possible to those obtained using all original prototypes.

According to the way in which prototypes are obtained and represented, there is a separation between *selection* techniques, in which the resulting prototypes are taken from the original set, and *replacement* techniques in which resulting prototypes are built and may be different from any prototype in the original set. Prototypes obtained by either of these methods can be referred to as *S*- and *R*-prototypes [3], respectively.

One of the first prototype selection algorithms was presented in Ref. [4]. This is in fact a selection condensing algorithm in which *S*-prototypes are picked from the original set in order to ensure that the selected prototypes manage to *correctly classify the entire original set*. This property which can be applied to both *R*- and *S*-prototypes is usually referred to as *consistency* and will be formally defined in Section 3.

The consistency property is a key issue in many condensing algorithms proposed to date. This is nothing but an easy way of making the 1-NN decision boundaries induced by the selected prototypes as close as possible to the ones induced by the original set of prototypes with the same rule. In this way, the quality of the condensed sets depends on the quality of the original set, and reducing the size of the selected set is the main goal of recently proposed condensing algorithms of both types. In fact, obtaining the minimal consistent set of a given set is considered as a challenging problem specially in the case of *S*-prototypes, where it becomes a hard combinatorial problem [5,6]. There has been a considerable interest in similar techniques to exemplar or instance-based methods in learning [7,8] and several interesting algorithms based on different concepts have recently been proposed.

Apart from consistency, there are other criteria for prototype generation. The LVQ [9] and DSM [10] algorithms are based on competitive learning update equations to modify the prototype set. The winning prototype is punished or rewarded, depending on the 1-*nearest prototype* classification result.

A novel approach for obtaining a consistent set of *R*-prototypes from an initial set is presented in this work. A way of considering the representativity of each prototype is proposed. Each *R*-prototype and the original samples it correctly classifies form a (labeled) cluster. Clusters from the same class can be merged to obtain a new prototype/cluster following a *Hierarchical Clustering* scheme. A preliminary version of these ideas has already been published [11]. In the present work, the geometric properties of the clusters are used to construct a consistent set of prototypes efficiently. This scheme constitutes a generalization of previous approaches presented by Chang [12] and Bezdek [13], but its flexibility and its ability to obtain good sets of prototypes goes far beyond them.

Section 2 presents a short description of the most important known algorithms, which constitute the groundwork for the new one presented here. Section 3 proposes the new geometric prototype-based learning scheme, and analyzes its computational advantages. In Section 4, some experiments are described which evaluate the performance of the new method along with a discussion of the results. Section 5 presents the conclusions and future developments.

2. Prototype replacement algorithms based on merging

Following the basic idea of replacing a group of samples by a representative, iterative solutions have been proposed to obtain consistent sets of *R*-prototypes via pairwise merging. Chang [12] and Bezdek et al. [13] presented the main condensing methods based on this strategy. In fact, the latter constitutes a slight improvement over the Chang algorithm, named as *modified chang algorithm* (MCA).

A general prototype-merging algorithm begins with a training set T, considering all the samples in T as initial prototypes. It then successively merges any two closest prototypes (p, q) of the same class (by replacing p and q with a new prototype p^*) if the merging does not degrade the classification of patterns in T (that is, if the resulting current set of prototypes is *consistent*). This process is stopped when no new merge is possible in any class. MCA introduces an algorithmic change in the way in which pairs of prototypes are merged, leading to smaller sets of prototypes than the Chang algorithm. A computational improvement is also achieved based on storing cross distances among prototypes in the same class only. Nevertheless, the strategy behind the original idea remains unchanged.

Both algorithms have two kinds of drawbacks. First, they have a restricted strategy for building prototypes based on pairwise merging and, consequently, they may provide condensing results which are far from the optimal ones, both from the point of view of their size and their representativity. And second, they employ a considerable amount of computation to check consistency exhaustively for any possible merging.

3. A new generalized prototype merging strategy

A further generalization of the idea of merging prototypes while maintaining consistency is proposed in this paper. MCA [13] can be viewed as a particular case of the algorithm template presented in Fig. 1. In fact, if prototypes are merged using the simple arithmetic mean, if the Euclidean distance is considered and if consistency is

Input: an initial set T, a way of merging prototypes, a distance between prototypes, a consistency-checking procedure

Output: the final set of prototypes P

Initialization: Let $P = T$ (the current consistent set)
Compute L as the set of all candidate pairs from P of the same class.

Method:

repeat
Let (p, q) be the pair from L whose distance is minimum and remove it from L.
Let p^* be the result of merging p and q and let $P^* = (P \cup \{p^*\}) - \{p, q\}$.
if P^* is consistent **then**
Let $P = P^*$ and recompute the set of candidate pairs, L, from P
until a complete pass through L has produced no merge

Fig. 1. An algorithmic description for prototype condensing based on merging.

exhaustively checked, the previous scheme is equivalent to MCA. Also, something very similar (in spirit) to the original Chang algorithm would be obtained if prototypes were merged through the weighted average and there was no recomputation of the list of candidate pairs, L, after updating the current set of prototypes, P, in the last line inside the *repeat* loop of the previous algorithm.

3.1. Using clusters as a more meaningful representation

Apart from trivial generalizations, the algorithmic scheme just presented allows us to extend the concept of prototype by attaching a subset of samples from T which are close enough to the prototype. The main idea consists of considering clusters of original samples and their cluster representatives as some sort of extended prototypes. In this way, the merging of prototypes becomes a union of two clusters (and the recomputation of the new representative), while the distance between prototypes becomes a cluster distance. Within this framework, the whole process can be seen as a particular case of a class-conditional agglomerative hierarchical clustering.

Looking at the process from a clustering viewpoint has several advantages. First, almost every single result from well-known clustering algorithms [14] is directly applicable. Second, merging prototypes on the basis of distances between the whole clusters that they represent may lead to a more meaningful placement of the final prototypes. It also opens up the possibility of having a final set of "informed" prototypes which may guarantee higher representativity apart from consistency. And last but not least, the use of clusters leads to some computational shortcuts when checking the consistency of prototypes.

3.2. From prototype consistency to cluster consistency

The underlying idea in the above generalized scheme consists of considering each prototype as a representative of the samples in its cluster. The consistency of the set of representatives with regard to the original sample set will be achieved by the fact that each representative is responsible for the correct classification of its cluster. This means that, for each sample in the cluster, the representative must be closer than any other representative from a different class. [1] This leads to the concept of cluster consistency as a sufficient condition with respect to the (plain) consistency (also referred to as *prototype consistency* in this work).

Definition 1 (Prototype-consistency). A given set of labeled prototypes, P, is said to be *prototype-consistent* with respect to a set of labeled points, T, if every point in T is correctly classified by the 1-NN rule using P as reference set.

Definition 2 (Cluster consistency). A given partition of a set of labeled prototypes, T, into clusters (with the corresponding set of representatives, P) is said to be *cluster-consistent* if every point in T is closer to its representative than to any other prototype in P with a different class label.

It trivially follows from these definitions that cluster consistency implies prototype consistency. The converse is not true in general but there is a close relationship between these two concepts that is formalized in the following proposition.

Proposition 1. *Any prototype-consistent set, P, with regard to a set T, induces a partition of T into clusters which is cluster-consistent.*

Proof. This partition can be obtained by grouping points in T according to their nearest neighbor in P which will have the same class label if P is prototype-consistent. □

[1] The representative of a sample x is not required to be its closest prototype.

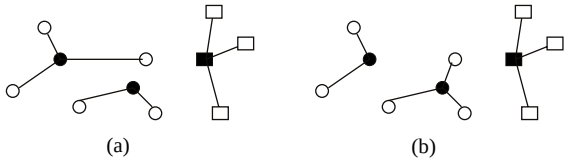

Fig. 2. An illustrative example with eight samples, two classes (circles and squares) and three clusters (the representatives are shown in black). Both partitions lead to the same consistent set of prototypes, but the partition in (b) is cluster-consistent and the one in (a) is not.

Cluster consistency can also be incrementally recovered provided there is a prototype-consistent set of prototypes. We only need to move points that produce inconsistencies from their actual cluster to the cluster of their closest representative in P. This is illustrated in Fig. 2.

3.3. An efficient consistency verification procedure

Exhaustively checking the consistency of p prototypes with regard to n original samples requires $O(np)$ time, which leads to agglomerative algorithms running in $O(n^3)$ time [12,13]. Moreover, the empirically observed hidden constants in these asymptotic results appear to be quite large.

As the consistency checking procedure is used to accept possible merges, the whole algorithm could benefit from early consistency confirmation by using the concept of cluster consistency. This, in turn allows us to use the auxiliary information and the geometric properties of the clusters involved to obtain some shortcuts in checking consistency.

All the concepts introduced so far, and consistency in particular, are independent of the particular metric used. Nevertheless, to simplify the Consistency Verification procedure and the corresponding formulae, *Euclidean* distance will be assumed unless otherwise stated. Comments about how the proposed methods extend to other distances will be appropriately included.

Let us assume that we have a cluster-consistent partition of a set of labeled prototypes, T, and that a new cluster-prototype, p^* from class k^* has been created. In this situation, checking cluster consistency would require that for every prototype, s, of a different class, no sample from the cluster of s is closer to p^* and, conversely, no sample from the new cluster is closer to s. In a more formal way:

Proposition 2. *Given a cluster-consistent partition of T, and the associated set of representatives P, the partition resulting from combining two clusters into a new one which is represented by p^* from class k^*, will also be cluster-consistent* if and only if

$$\forall s \in P \text{ whose class is different from } k^* \quad \begin{cases} d(s,x) > d(p^*,x), \forall x \text{ in the cluster of } p^* \\ d(s,y) < d(p^*,y), \forall y \text{ in the cluster of } s. \end{cases} \tag{1}$$

The proof of this proposition follows trivially from Definition 2. It is worth noting that checking cluster consistency using Condition (1) does not alleviate the computational problem mentioned above. Nevertheless, it is possible to obtain a more efficient procedure by using geometric information about clusters to perform an early assessment of possible consistency.

Let r_p be the radius associated to the cluster represented by prototype p, in such a way that any sample in this cluster is at a distance from p which is not greater than r_p. If the radii of two clusters are smaller than half the distance between their representatives, there is no need to go through all the samples in both clusters to check consistency as stated in Condition (1). In other words,

Proposition 3. *Given a cluster-consistent partition of T, and the associated set of representatives P, the partition resulting from combining two clusters into a new one which is represented by p^* from class k^*, will satisfy Condition* (1) if

$$\forall s \in P\text{: } class(s) \neq k^*, \quad d(p^*,s) > 2 \cdot \max(r_{p^*}, r_s). \tag{2}$$

A similar equation can be obtained for distances other than the Euclidean one. Details about this and the proof of the proposition are given in the appendix.

Corollary 1. *Condition* (2) *is sufficient for cluster consistency.*

Fig. 3 shows a simple example where Condition (2) is used to assess cluster consistency.

Condition (2) can be further generalized by considering only the closest representatives for each class and the maximum radius in the corresponding class.

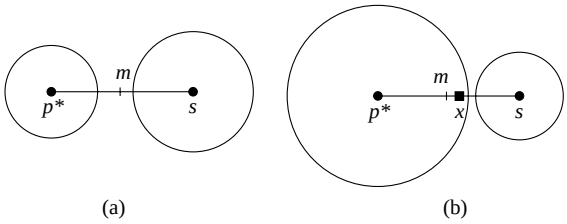

Fig. 3. A geometric representation of Condition (2) which is sufficient to test cluster consistency: (a) the condition is satisfied; (b) the condition is not satisfied (could exist a sample x which is represented by p^* but which is closer to s).

Proposition 4. *Given a cluster-consistent partition of T, and the associated set of representatives P, the partition resulting from combining two clusters into a new one which is represented by p^* from class k^*, will satisfy Condition* (2) if

$$\forall k \neq k^*, \quad d(p^*, s^k) > 2 \cdot \max(r_{p^*}, r^k), \tag{3}$$

where s^k and r^k denote the prototype of class k that is closest to p^* and the radius of the maximum-radius cluster in class k, respectively.

Note that the rest of the cluster representatives of each class $k \neq k^*$ are located farther from p^* than s^k and their associated radii are smaller than r^k. A detailed proof is given in the appendix.

Corollary 2. *Condition* (3) *is sufficient for cluster consistency.*

Fig. 4(a) illustrates a situation in which cluster consistency can be assessed using Condition (3). A different one is shown in Fig. 4(b) in which Condition (3) fails and cluster consistency can be confirmed by using Condition (2).

Consequently, Condition (3) can be used as the *first* stage, Condition (2) as the *second* stage, and Condition (1) as the *third* stage, of an efficient scheme to check cluster consistency, and therefore, prototype consistency (the final goal). In this way, when prototype consistency cannot be assessed by using the conditions listed in the given order, it needs to be directly checked and transformed in cluster consistency, by moving points that produce cluster inconsistencies from their current clusters to the clusters of their closest representatives. This process will be referred to as *fourth* stage. The consistency checking strategy is schematically shown in Fig. 5.

The fourth stage is needed when Condition (3) fails for some class k, Condition (2) fails for some cluster s member of class k, and Condition (1) is not satisfied by a sample x member of the cluster represented by p^* or a sample y member of s. In this context, checking prototype consistency requires looking for the *same-class nearest prototype*[2] to the sample, x or y, which is producing the cluster inconsistency. The distance between them is compared with the distance between the sample and the other-class prototype involved. If the sample is correctly classified, it must be moved from its current cluster to the cluster represented by its same-class nearest prototype, transforming the current prototype consistency into cluster consistency (see Proposition 1). A detailed example is shown in Fig. 6.

When compared to exhaustively checking consistency, the computational burden can be reduced even when the merge is not accepted, because the more intensive fourth stage is only applied on isolated samples. A more detailed algorithmic description of the proposed consistency checking procedure is presented in Fig. 7.

With regard to the application of the fourth stage, it is worth noting that as the new set of prototypes is consistent, each sample that has produced cluster inconsistency has necessarily been *moved* from its original cluster to the cluster of its same-class nearest prototype. From this moment, the behavior of the algorithm separates from a truly agglomerative hierarchical clustering scheme.

Even though the worst-case complexity of the consistency checking procedure presented here is certainly in $O(np)$, the empirically observed behavior of the final algorithm is

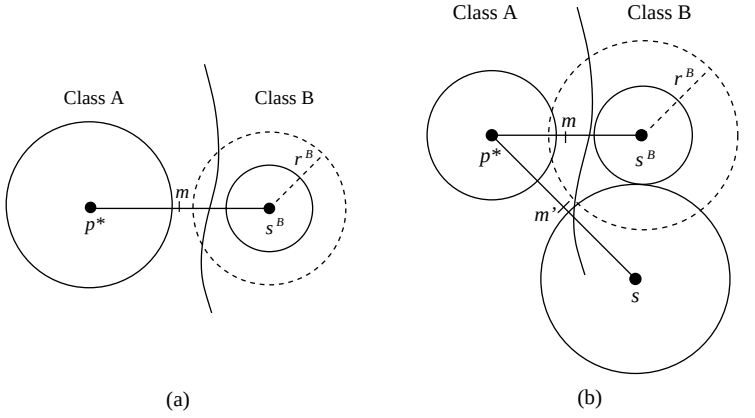

Fig. 4. Illustrative examples. Let m and m' be the midpoints between prototypes. In (a) consistency can be guaranteed by Condition (3). No sample can possibly create an inconsistency because other class B representatives must be farther than s^B and with smaller or equal radius than r^B. In (b) Condition (2) is needed to guarantee consistency because Condition (3) is not satisfied. All clusters in class B (the ones represented by s^B and s) need to be visited.

[2] This prototype does not need to be the sample representative.

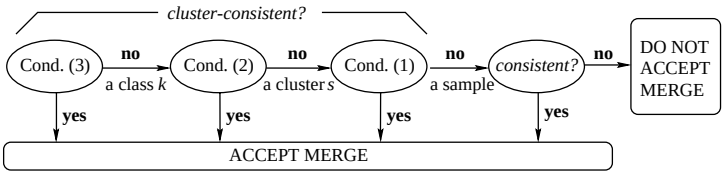

Fig. 5. Schematic description of the consistency checking procedure.

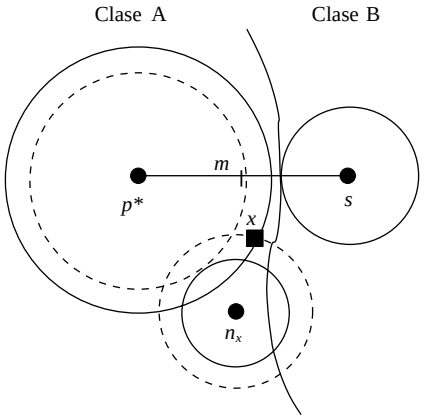

Fig. 6. Geometric situation in which the three first stages to check cluster-consistency cannot be applied, therefore the fourth stage is needed to test prototype-consistency and to transform it in a cluster-consistent partition. The class A is where the new merge took place; p^* is the new prototype, s is a prototype in class B which is causing a cluster inconsistency regarding its distance to some sample x member of the cluster represented by p^*. The fourth stage finds the prototype n_x which is the nearest prototype to x in class A, checks the maintenance of the prototype-consistency and moves the sample x from the cluster of p^* to the cluster of n_x. In this way the prototype-consistency is transformed into cluster-consistency. The dashed circles represent possible variations on radii of clusters involved.

significantly better than our MCA implementation. The computational cost of the four stages of the previous algorithm are $O(p)$, $O(p)$, $O(np)$ and $O(np)$, respectively. Nevertheless, it has been observed that the third and fourth stages have a reduced influence in the work done to check consistency, specially for moderate-size data sets (see Section 4).

The whole proposed method could still benefit from the implementation of efficient neighbor search to compute s^k after each merge. Nevertheless, the bottleneck of the method is the recomputation of candidate pairs and the condition checking itself and not neighbor finding. Use of alternative and/or approximate distances is another possibility that is also currently under investigation.

3.4. The generalized-modified Chang algorithm (GMCA)

MCA can be generalized from the algorithm in Fig. 1. To do this, the algorithm must be extended to clusters and any intercluster dissimilarity measure must be considered as a merging criterion. In addition, the consistency checking procedure shown in Fig. 7 could also be used. The resulting method will be named GMCA. In the particular case of the intercluster distance Median [14], (which implies the simple mean as the way to agglomerate prototypes), GMCA results in an improved version of the MCA yielding identical results, but reducing the computing time by more than half in most cases. When other intercluster measures were used, smaller and better sets of prototypes (in the sense of classification power) were built, reducing the condensing time even more.

4. Experimental results

A number of experiments were conducted to compare GMCA (with different intercluster measures) and MCA with respect to the number of prototypes built, the error rate of the corresponding condensed 1-NN classification rule and the computation time. Experiments were also used to show how GMCA works, in accordance with the percentage of consistency work carried out by each stage.

Four basic intercluster distances [14] were used: average link (AV), complete link (CO), median (ME), and the Ward minimum variance method (WA). Also, the radius of the next agglomerated cluster was used as an additional dissimilarity measure (RA). Following the suggestions found in Ref. [13], the Euclidean distance and the simple mean as the way to compute new prototypes were always used, except for Ward's method where the weighted mean and the squared Euclidean distance [3] were considered because of its original formulation [14]. In this way, a proper comparison between GMCA and MCA results can be accomplished.

First, a simple example is presented to illustrate the condensing capacity of the GMCA merging schemes with respect to previous approaches (Chang and MCA), in the sense of building the least number of prototypes to represent a set and classify it without errors with the 1-NN rule. For such purposes, a modified version (suggested in Reference [12]) of the synthetic data set originally proposed by

[3] For squared distance, Conditions (2) and (3) need to be changed by substituting the coefficient 2 by 4. See the appendix for more details.

Consistency Checking Procedure

Input: a new prototype p^*, its class k^*, and the corresponding radius r_{p^*}.

Output: a boolean variable $MERGE$.

Initialization: Let s^k be the closest prototype to p^* in class k
Let P^k be the subset of prototypes of class k
Let $MERGE = TRUE$

Method:

forall class k, $k \neq k^*$, such that $d(p^*, s^k) \leq 2 \cdot \max(r_{p^*}, \max_{s \in P^k}(r_s))$ **do** // Cond. (3)

  forall prototype $s \in P^k$, such that $d(p^*, s) \leq 2 \cdot \max(r_{p^*}, r_s)$ **do** // Cond. (2)

    forall sample x member of the cluster of p^* such that $d(p^*, x) \geq d(s, x)$ **do** // Cond. (1)
      Let $n_x = \arg\min_{t \in P^{k^*}}\{d(t, x)\}$
      if $d(n_x, x) < d(s, x)$ **then**
        Move x from the cluster of p^* to the cluster of n_x
      else $MERGE = FALSE$; **exit**
    end forall

    forall sample y member of the cluster of s such that $d(p^*, y) \leq d(s, y)$ **do** // Cond. (1)
      Let $n_y = \arg\min_{t \in P^k}\{d(t, y)\}$
      if $d(n_y, y) < d(p^*, y)$ **then**
        Move y from the cluster of s to the cluster of n_y
      else $MERGE = FALSE$; **exit**
    end_forall

  end_forall

end_forall

Fig. 7. Algorithmic description of the proposed consistency checking procedure.

Hart [14] was considered. This problem has two categories of two-dimensional non-overlapping regions which are adjacent to each other. Chang reported 17 prototypes built by his method, which were needed to classify this particular set without errors.

Both MCA and its equivalent GMCA+ME obtained the 13 prototypes shown in Fig. 8(a) (the radii shown were computed by the GMCA+ME method). All other GMCA schemes lead to 10 (CO and RA), 11 (AV) and 12 (WA). Fig. 8(b) shows the decision boundary induced by the 10 final prototypes built by GMCA+CO and their associated radii. Note that the optimal number of prototypes to exactly define the piecewise-linear decision boundary in this problem is 10.

Three real data sets were used to compare the merging schemes. These sets are the well-known Anderson Iris data [15], the DNA data set [16] and the Landsat Satellite Image data [16], which are publicly available at UCI Repository Machine Learning [17].

Apart from measuring the final number of prototypes, and the computational burden (in time and computed distances), the percentage of work carried out by each stage in the consistency checking procedure was obtained as a percentage of cluster consistencies which were solved at each stage.

Both schemes (MCA and GMCA) were implemented in C programming language using the same data structures and code to make them as similar as possible. The major functional difference was in the checking-consistency procedure. The experiments were performed on a 400 MHz Intel Pentium III with 256 Mb of RAM.

Apart from the computational burden of each algorithm considered, the Percentage of CPU Time spent by MCA (PTMCA) and the Percentage of computed Distances by MCA (PDMCA) are included as relative measures. The percentage of consistencies solved at each one of the four stages of the Consistency Verification Procedure is also shown.

4.1. Experiments on the Iris data set

This set has three classes that represent three varieties of iris flowers: Iris Setosa, Iris Versicolor and Iris Virginica. Fifty samples were obtained from each of the three classes, thus a total of 150 samples is available. Every sample is described by four measurements. Using the 150 prototypes available, Chang [12] and Bezdek et al. [13] reported two consistent sets of 14 and 11 prototypes built by their schemes, respectively. The results obtained by GMCA are shown in Table 1. The best result obtained with GMCA+WA, consisting of only 9 consistent prototypes are shown in Table 2.

The merging schemes GMCA+WA, GMCA+CO and GMCA+RA built consistent sets of prototypes which were smaller than the ones obtained by the MCA and by the Chang approach. It is worth noting that all GMCA schemes were about twice as fast as MCA at obtaining solutions which were equal or better.

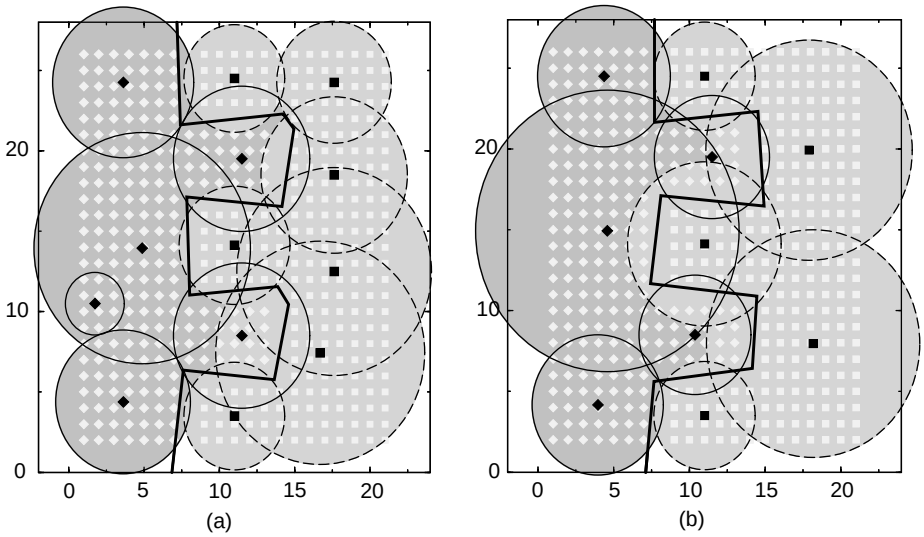

Fig. 8. Illustrative synthetic data condensing results. (a) prototypes and Voronoi diagram obtained by MCA and GMCA+ME (which also gave the displayed radii). (b) prototypes, associated radii and Voronoi diagram obtained by GMCA+CO.

Table 1
Condensing results on the Iris data set

Merging scheme	No. of prot.	CPU time (ms)	PTMCA (%)	PDMCA (%)	% use stage 1	% use stage 2	% use stage 3	% use stage 4
MCA	11	90	100	100	—	—	—	—
GMCA+ME	11	40	44.44	28.53	89.918	6.431	3.210	0.441
GMCA+CO	10	60	66.67	55.43	87.693	8.249	3.716	0.342
GMCA+WA	9	60	66.67	36.34	82.043	10.443	6.863	0.651
GMCA+RA	10	60	66.67	45.56	85.769	10.107	3.829	0.295
GMCA+AV	11	60	66.67	61.92	85.910	7.970	5.451	0.669

Table 2
The 9-prototype consistent set built by GMCA+WA from the Iris data

Iris setosa	Iris versicolor	Iris virginica
5.01 3.43 1.46 0.25	6.72 3.00 4.68 1.46	6.89 3.10 5.81 2.12
	6.15 3.27 4.62 1.62	5.95 2.77 4.97 1.92
	5.70 2.66 4.07 1.25	5.67 2.43 5.03 1.53
	6.15 2.60 5.00 1.55	6.35 2.75 5.20 1.70

4.2. Experiments on the DNA data set

This data set corresponds to primate gene sequences. The problem is to recognize boundaries between different parts of the DNA. There are two sets, one for training composed by 2000 samples and one for test with 1186 samples, which are partitioned into three classes. Each sample is described by 180 binary attributes (Statlog version).

Error rates were estimated on the test set by using the 1-NN rule with the condensed sets of prototypes which were built from the training set. Table 3 lists the condensing results. Classification error rates on test set by using the 1-NN and the 30-NN (best k-NN classifier) rules with the original training set were 23.44% and 13.07%, respectively. All these results are graphically shown in Fig. 9.

GMCA merging schemes yielded smaller sets of prototypes than MCA, which were surprisingly much better at classifying the test set. Both 1-NN and best k-NN error rates were reduced. At the same time, GMCA schemes achieved a very important reduction in the resources (condensing time

Table 3
Condensing results on the DNA data set

Merging scheme	No. of prot.	Error rate	CPU time (min)	PTMCA (%)	PDMCA (%)	% use stage 1	% use stage 2	% use stage 3	% use stage 4
MCA	193	21.33	4500	100	100	—	—	—	—
GMCA+ME	193	21.33	1337	29.71	29.70	0.24	0.15	93.69	5.92
GMCA+CO	68	12.48	23.4	0.52	0.16	18.04	26.09	54.96	0.91
GMCA+WA	59	10.12	54.9	1.22	0.10	20.48	29.80	49.04	0.68
GMCA+RA	65	11.38	133.7	2.97	0.09	22.47	32.78	44.00	0.75
GMCA+AV	63	11.89	63.0	1.40	0.49	10.81	14.92	72.49	1.78

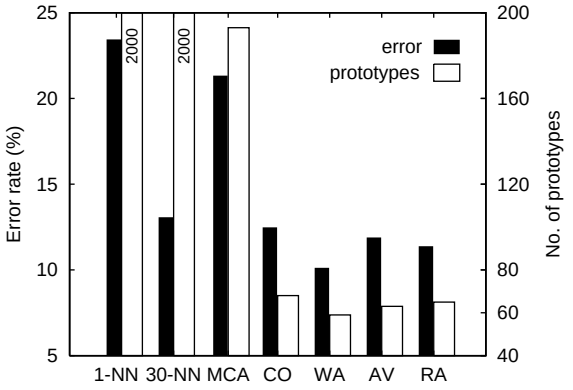

Fig. 9. Error rates (%) (on the test set) and number of prototypes for the DNA experiment, considering 1-NN, 30-NN and the condensed 1-NN classification rule with the MCA prototypes and GMCA (CO, WA, AV, RA), respectively.

and computed distances) required by MCA. Although in this case CPU times were quite high, it is worth noting that all GMCA variants apart from GMCA+ME (which is equivalent to MCA) spent CPU times below 3% of that of MCA.

4.3. Experiments on the Landsat Satellite Image data

The purpose of this experiment was to illustrate the capabilities of the GMCA merging schemes with respect to previous approaches by using a real, well-behaved and "reasonably" large database. This third experiment was performed on the Landsat Satellite Image data. This database consists of the multi-spectral values of pixels in 3×3 neighborhoods in a satellite image, and the classification associated with the central pixel in each neighborhood. The aim is to identify regions with different soils and crops. There are two sets, one for training composed by 4435 samples and one for test with 2000 samples, which are partitioned into six classes of 36-dimensional data.

As consistency does not make sense in the case of overlapping among classes, the Wilson editing scheme [18] was applied on the training set. A five-fold cross validation experiment was performed on the training set to estimate a value for k to use in the experiments. The value $k = 8$ was selected because of the lowest average classification error on test partitions, when the 1-NN rule was used with their edited training partitions.

The edited original training set (4018 samples) was condensed with the different merging schemes. The error rates were estimated on the test set by using the 1-NN rule with the condensed sets of prototypes. Table 4 lists the condensing results on the edited training set including the estimated error rates. As a reference, the classification results on the test set by using the 1-NN and the 4-NN (best k-NN classifier) rules with the original training set were 10.55% and 9.25%, respectively.

The GMCA strategy generated condensed set of prototypes notably faster than MCA, proving its better computational efficiency. Note the trend of achieving higher time reductions while increasing the percentage of use of stages 1 and 2. The GMCA sets of prototypes were generally better than MCA ones, although the differences in their sizes and the corresponding condensed 1-NN classification results were not significant. Some GMCA methods built more prototypes than MCA, which generally led to better accuracies. On the other hand, GMCA+CO built a smaller set than the MCA one, achieving also a better classification result. This fact shows the flexibility of GMCA schemes to better fit the structure of data with respect to MCA. From the point of view of performance, the best scheme was GMCA+RA.

5. Conclusions and further work

A generalized condensing scheme based on class-conditional hierarchical clustering (GMCA) is proposed. The basic idea is to replace a group of prototypes by a representative while keeping the consistency property. The algorithm improves and generalizes previous work by explicitly introducing the concept of cluster and cluster consistency. The use of geometric cluster properties produces a very efficient merging scheme based on local consistency verification, guaranteeing the entire system consistency while minimizing the computation needed.

This procedure of consistency verification constitutes the kernel of the merging scheme. It is integrated by a family

Table 4
Condensing results on the Landsat Satellite Image data set

Merging scheme	No. of prot.	Error rate	CPU time (min)	PTMCA (%)	PDMCA (%)	% use stage 1	% use stage 2	% use stage 3	% use stage 4
MCA	119	11.45	82.8	100	100	—	—	—	—
GMCA+ME	119	11.45	34.8	42.03	34.32	16.866	67.687	14.637	0.810
GMCA+CO	118	10.70	14.0	16.91	10.15	67.746	20.743	10.497	1.014
GMCA+WA	121	11.55	24.2	29.23	6.80	65.905	20.342	12.454	1.300
GMCA+RA	124	10.35	22.1	26.69	9.18	68.182	19.212	11.172	1.434
GMCA+AV	126	10.60	24.5	29.59	21.37	47.008	38.089	13.712	1.191

of four complementary stages, with an increasing order of complexity. After any new merge the "least effort" is made to verify only the consistency of the new local structure introduced by the new prototype (and the new cluster), guaranteeing the consistency of the entire set of prototypes. It avoids operating on all points of the original training set as MCA and Chang's algorithm do.

MCA was experimentally compared with merging schemes induced by GMCA taking into account five different intercluster dissimilarity measures. In the particular case of the GMCA with the Median intercluster distance (GMCA+ME), which yields identical sets of prototypes as MCA, a remarkable reduction in the time and the computed distances required was achieved in all experiments.

When GMCA was considered with the rest of intercluster dissimilarity measures, the sets of prototypes were better, in most of cases, than those obtained by MCA. In general, GMCA was able to better figure out and fit the internal data structure than MCA, and consequently, to represent it in a more suitable way through a (usually more reduced) set of prototypes. The experiments showed the capacity of these schemes to construct more compact clusters than GMCA+ME, leading to higher percentages of use of the most efficient stages of the consistency checking procedure. Therefore, and in spite of the greater complexity required to compute these intercluster dissimilarity measures with respect to the Median distance, the amount of resources (time and computed distances) needed by the GMCA with these measures were, in most cases, smaller than those required by GMCA+ME, and, consequently, smaller than the MCA.

A number of technical improvements can still be applied. For example, intercluster distances could be computed in constant time by making proper use of the Lance–Williams combinatorial formula [14]. Nearest neighbors could be efficiently found using appropriate algorithms [19–21]. The concepts of consistency, distance definition or even cluster membership could also be relaxed in order to speed up some specific parts of the algorithm. All these aspects are currently under investigation.

According to the proofs in the appendix, the proposed method can be used with any metric satisfying triangle inequality and symmetry. Other distances as the squared Euclidean require a slight modification of the equations and, in general, any distance can be used by appropriately modifying conditions (2) and (3) by an additive term according to the degree of violation of triangle equality and symmetry. This term can be empirically obtained when no other information is available.

As the proposed condensing scheme does not involve explicit coordinate computation (apart from computing p^*) it can naturally be extended to metric (non-vector) spaces by appropriately selecting p^* from the available prototypes in the two merged clusters.

Another, particularly interesting future line of work could involve the information GMCA obtains from the sets in form of radii. As can be seen in Fig. 8, the set of prototypes and its associated radii give a good and quite natural description of the data set. This information is currently used by the GMCA only, but it could be fed into the corresponding classifier or used as an initialization stage for other learning and/or classification algorithms.

Appendix

Proof of Proposition 3. To show that Condition (2) is sufficient for Condition (1), the following statement must be proved for all clusters s (members) of a class which is different from k^*:

$$d(p^*,s) > 2 \cdot \max(r_{p^*}, r_s) \Rightarrow \begin{cases} \text{i. } d(s,x) > d(p^*,x), \ \forall x \text{ in the cluster of } p^* \\ \text{ii. } d(s,y) < d(p^*,y), \ \forall y \text{ in the cluster of } s \end{cases}$$

To prove part i., we have $\forall x$ in the cluster of p^*,

$$d(p^*,x) \leqslant r_{p^*} \leqslant \max(r_{p^*}, r_s) < \frac{d(p^*,s)}{2} \leqslant \frac{d(p^*,x) + d(x,s)}{2}$$

where the last step is due to the *triangle inequality*.

Finally, we arrive at $d(p^*,x) < d(s,x)$ if the distance satisfies the symmetry property.

Part ii. can be proved in the same way. □

Proof of Proposition 4. To show that Condition (3) is sufficient for Condition (2), the following statement must be proved for all classes k which are different from k^*:

$$d(p^*,s^k) > 2\cdot\max(r_{p^*},r^k) \Rightarrow d(p^*,s) > 2\cdot\max(r_{p^*},r_s), \quad \forall s \text{ in class } k$$

where s^k and r^k denote the prototype of class k that is closest to p^* and the radius of the maximum-radius cluster in class k, respectively.

For all s from a class k different from k^* we have

$$d(p^*,s) \geqslant d(p^*,s^k) > 2\cdot\max(r_{p^*},r^k) \geqslant 2\cdot\max(r_{p^*},r_s)$$

where we have applied the definition of s^k, the Condition (3) and the definition of r^k, respectively. □

Case of (Euclidean) squared distances

For any distance satisfying $d(a,c) \leqslant d(a,b) + d(b,c)$ (triangle inequality), we have

$$d^2(a,c) \leqslant (d(a,b)+d(b,c))^2 = d^2(a,b)+d^2(b,c)+2d(a,b)d(b,c)$$

from which the following inequality is obtained (given the fact that $2xy \leqslant x^2+y^2$ for any two reals):

$$\tfrac{1}{2}\cdot d^2(a,c) \leqslant d^2(a,b)+d^2(b,c).$$

This equation can be used instead of triangle inequality to prove the previous propositions. In this case the factor 2 appearing in Eqs. (2) and (3) need to be changed to 4.

References

[1] B.V. Dasarathy (Ed.), Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques, IEEE Computer Society Press, Los Alamitos, CA, 1991.

[2] P. Devijver, J. Kittler, Pattern Recognition: A Statistical Approach, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[3] Liudmila I. Kuncheva, James C. Bezdek, Nearest prototype classification: clustering, genetic algorithms, or random search?, IEEE Trans. Systems Man Cybernet. 28 (1) (1998) 160–164.

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

[5] B.V. Dasarathy, Minimal consistent set (MCS) identification for optimal nearest neighbour decision systems design, IEEE Trans. Systems Man Cybernet. 24 (3) (1994) 511–517.

[6] V. Cerverón, F.J. Ferri, Another move towards the minimum consistent subset: a tabu search approach to the condensed nearest neighbor rule, IEEE Trans. Systems Man Cybernet. B 31 (3) (2001) 408–413.

[7] D.B. Skalak, Prototype and feature selection by sampling and random mutation hill climbing algorithms, Proceedings of the 11th International Conference on Machine Learning, Morgan Kaufmann, Los Altos, CA, 1994, pp. 293–301.

[8] D.R. Wilson, T.R. Martinez, Reduction techniques for instance-based learning algorithms, Mach. Learning 38 (3) (2000) 257.

[9] T. Kohonen, Self-Organizing Maps, Springer, Germany, 1995.

[10] S. Geva, J. Sitte, Adaptive nearest neighbor pattern classifier, IEEE Trans. Neural Networks 2 (2) (1991) 318–322.

[11] R.A. Mollineda, F.J. Ferri, E. Vidal, A cluster-based merging strategy for nearest prototype classifiers, Proceedings of the 15th ICPR. Vol. 2, Barcelona, Spain, 2000, pp. 759–762.

[12] Chin-Liang Chang, Finding prototypes for nearest neighbor classifiers, IEEE Trans. Comput. 23 (11) (1974) 1179–1184.

[13] James C. Bezdek, Thomas R. Reichherzer, Gek Sok Lim, Yianni Attikiouzel, Multiple-prototype classifier design, IEEE Trans. Systems Man Cybernet. 28 (1) (1998) 67–79.

[14] F. Murtagh, A survey of recent advances in hierarchical clustering algorithms, Comput. J. 26 (4) (1983) 354–359.

[15] E. Anderson, The IRISes of the gaspe peninsula, Proc. Bull. Amer. IRIS Soc. 59 (1935) 2–5.

[16] Ross D. King, Statlog databases, Technical Report, Department of Statistics and Modelling Science, University of Strathclyde, Glasgow G1 1XH, Scotland, U.K., October 1992.

[17] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[18] D.L. Wilson, Asymptotic properties of nearest neighbour rules using edited data, IEEE Trans. Systems Man Cybernet. 2 (1972) 408–421.

[19] E. Vidal, New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (AESA), Pattern Recognition Lett. 15 (1) (1994) 1–7.

[20] M.L. Mico, J. Oncina, E. Vidal, A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements, Pattern Recognition Lett. 15 (1) (1994) 9–17.

[21] V. Ramasubramanian, K. Paliwal, Fast nearest-neighbour search algorithms based on approximation-elimination search, Pattern Recognition 33 (2000) 1497–1510.

About the Author—RAMÓN A. MOLLINEDA received the BS and MS in Computer Science from Central University of Las Villas, Cuba, in 1995 and 1997, respectively. In 2001, he received the Ph.D. from Politechnical University of Valencia, Spain.

In 1998 he joined the Automatic Speech Recognition group in the Politechnical University of Valencia first with a grant from the Agencia Española de Cooperación Internacional (AECI) and later and until now as a research fellow at the Informatic Technology Institute at the same University where he has been involved in several research projects.

His current topics of interest include statistical pattern recognition, hierarchical clustering, non-parametric classifiers and feature selection.

Dr. Mollineda is a member of the Spanish Society for Pattern Recognition and Image Analysis (AERFAI) and the International Association for Pattern Recognition (IAPR).

About the Author—FRANCESC J. FERRI received the *Licenciado* degree in Physics (Electricity, Electronics and Computer Science) in 1987 and the Ph.D. in Pattern Recognition in 1993 both from the Universitat de València.

Dr. Ferri has been with the Computer Science and Electronics Department of the Universtitat de València since 1986; first as a research fellow and as a teacher of Computer Science and Pattern Recognition since 1988. He has been involved in a number of scientific and technical projects on Computer Vision and Pattern Recognition including a sabbatical with the Vision, Speech and Signal Processing group in the University of Surrey, UK.

He has authored or coauthored about 90 conference and journal papers on several aspects of Computer Vision and Pattern Recognition. His current research interests include Feature Selection, Statistical Pattern Recognition Methodology, Non-parametric Classification Methods, Neural Networks, Inductive Learning, Computational Geometry and Image Analysis.

Dr. Ferri is a member of the Spanish Society for Pattern Recognition and Image Analysis (AERFAI), the International Association for Pattern Recognition (IAPR) and the Association for Computing Machinery (ACM).

About the Author—ENRIQUE VIDAL received the *Licenciado* degree in Physics in 1978 and the *Doctor en Ciencias Físicas* (Ph.D. in Physics) degree in 1985, both from the Universitat de València.

From 1978 to 1986 he was with this University serving in computer system programming and teaching positions. In the same period he coordinated a research group in the fields of Pattern Recognition and Automatic Speech Recognition. In 1986 he Joined the Departamento de Sistemas Informáticos y Computación of the Universidad Politécnica de Valencia (UPV), where he is until now serving as a full professor of the Facultad de Informática. In 1995 he joined the Instituto Tecnológico de Informática, where he has been coordinating several projects on Pattern Recognition and Machine Translation. He is co-leader of the Pattern Recognition and Human Language Technology group of the UPV.

His current fields of interest include Statistical and Syntactic Pattern Recognition, and their applications to language, speech and image processing. In these fields, he has published more than one hundred papers in journals, conference proceedings and books.

Dr. Vidal is a member of the Spanish Society for Pattern Recognition and Image Analysis (AERFAI) and the International Association for Pattern Recognition (IAPR).