International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 19, No. 6 (2005) 787–806

World Scientific
www.worldscientific.com

DECISION BOUNDARY PRESERVING PROTOTYPE SELECTION FOR NEAREST NEIGHBOR CLASSIFICATION

RICARDO BARANDELA

Instituto Tecnológico de Toluca, Av. Tecnológico s/n
52140 Metepec, México

FRANCESC J. FERRI

Department d'Informàtica, U. Valencia
46100 Burjassot (Valencia), Spain

J. SALVADOR SÁNCHEZ*

Department Llenguatges i Sistemes Informàtics, U. Jaume I
12071 Castelló, Spain

The excessive computational resources required by the Nearest Neighbor rule are a major concern for a number of specialists and practitioners in the Pattern Recognition community. Many proposals for decreasing this computational burden, through reduction of the training sample size, have been published. This paper introduces an algorithm to reduce the training sample size while preserving the original decision boundaries as much as possible. Consequently, the algorithm tends to obtain classification accuracy close to that of the whole training sample. Several experimental results demonstrate the effectiveness of this method when compared to other reduction algorithms based on similar ideas.

Keywords: Nearest Neighbor rule; size reduction; classification accuracy; consistent subset; decision boundaries.

1. Introduction

The Nearest Neighbor (NN) rule is one of the oldest and better-known algorithms for performing supervised nonparametric classification. The entire training set (TS) is stored in the computer memory. To classify a new pattern, its distance to each one of the stored training patterns is computed. The new pattern is then assigned to the class represented by its nearest training pattern. From this definition, it is obvious that this classifier suffers from a main drawback: large memory requirement to store the whole TS and the also large response time needed. This disadvantage is more critical in contexts like Data Mining where huge databases are commonly dealt with

*Author for correspondence.

in Refs. 4 and 22. Nevertheless, the NN rule is very popular mainly because: (a) conceptual simplicity; (b) easy implementation; (c) known error rate bounds; and (d) potentiality to compete favorably in accuracy with other classification methods in practice.

The above mentioned drawback has been considerably cut down by the development of suitable data structures and associated search algorithms and also by proposals to reduce the TS size. Hart's idea[13] of a consistent subset has become a milestone in the latter research line, stimulating a sequel of new algorithms aimed at eliminating as many training patterns as possible without seriously affecting the predictive accuracy of the classifier. Most of the research done in this direction is reviewed in Refs. 27 and 29 from slightly different viewpoints.

The present work is concerned with reducing the TS size while trying to maintain (or even improve) the accuracy rate of the NN rule. A novel reduction technique based on some features of the Selective Subset,[23] the Modified Selective Subset (MSS), is presented. In comparison to the original Selective algorithm, the MSS yields a better approximation of the decision boundaries as induced by the plain NN rule when using the whole training sample. As a byproduct, an algorithm much simpler (in terms of storage and computational time requirements) is obtained. This algorithm may be of crucial interest in situations in which particularly good decision boundaries need to be accurately represented by a reduced set of prototypes.

In the following section, we briefly review the basic characteristics of both the Condensed[13] and Selective[23] subsets. Next, the Modified Selective Subset is introduced in Sec. 3 where a detailed derivation of the new algorithm from the (Minimum) Selective Subset is also given. Experimental results with synthetic and real datasets are presented in Sec. 4. The results allow to compare in both classification accuracy and reduction rate the proposed algorithm with regard to several similar methods. The final section presents some concluding comments.

2. Techniques for Reducing the TS Size

It is generally acknowledged that the NN rule is able to properly generalize in a variety of practical situations either as it is or as part of a composite classifier. However, since it must store all the available training patterns and search through all of them to identify a new pattern, it has large memory requirements and may become too slow in the classification phase. Practical importance of this subject is obvious in many domains, such as data mining, text categorization, remote sensing, and retrieval of multimedia databases, to name a few. As a consequence, a vast catalogue of methods to reduce the TS size (as one of the possibilities of overcoming this drawback) has been proposed in the literature.

TS size reduction approaches can be classified into two categories[3,29] according to which it is possible to distinguish between those schemes that select a subset of the original training patterns, and those that generate new ones. The former is the case of Hart's algorithm,[13] the Selective Subset[23] approach, the

Proximity Graph-based editing and condensing methods,[24] the Minimal Consistent Subset[8] condensing, Tomek's algorithms,[25] and the Reduced NN.[11] Among the proposals aimed at creating new prototypes, the following can be mentioned: merging schemes,[3,6] clustering-based methods,[20] the LVQ methods,[16] and the Bootstrap technique introduced in Ref. 12.

In this paper, as in Ref. 29, we concentrate on those techniques that reduce the TS size by retaining only a subset of the original training patterns. This option is mandatory in many cases in which there is no vector space available to support prototype generation or when the newly generated prototypes make no sense in the particular application domain. These selection algorithms primarily operate by removing irrelevant and redundant patterns while retaining only critical cases using different philosophies.[29] Within this group, to take note is the set of approaches based on the concept of *consistency*. A consistent subset of a TS is any subset that correctly classifies every pattern in TS using the 1-NN. Even though consistency implicitly assumes that the original TS has no noise and is good enough at classification,[29] selecting a proper consistent subset has been an active area of research that recently has also received a theoretical justification in the context of Structural Risk Minimization theory.[15] Almost all the methods named above are based on consistency and, in particular, Hart's[13] and Ritter's[23] approaches which are particularly important for the present work. In most consistency-based approaches, the goal has been intimately related to obtaining a minimum cardinality subset. In the present work, however, the goal will be related with obtaining a sufficiently reduced subset (not necessarily minimal) that most accurately preserves the original classification boundaries.

2.1. *The condensed subset*

Hart's algorithm is the earliest attempt at minimizing the number of stored patterns by retaining only a consistent subset of the original TS. This algorithm finds a condensed subset (CS) from the training set, TS. This scheme begins by randomly selecting one pattern belonging to each class from the TS and putting them into CS. Each remaining pattern in TS is then classified using the prototypes in the current CS. If a pattern in the TS is misclassified, it is added to CS. This process is repeated until there are no patterns in the TS that are misclassified by CS. An algorithmic description of this procedure is shown in Fig. 1.

The process ensures that all prototypes in TS are classified correctly. That is, the resulting CS is a consistent subset. But the algorithm does not guarantee minimality. Moreover, both the quality and the size of the condensed subset depend on the order in which the training patterns are presented to the algorithm. Patterns that are examined early in the process tend to be added to CS. Consequently, not all the training patterns close to the decision boundaries are maintained in the reduced subset and some redundant (internal) prototypes are unnecessarily kept. Notwithstanding, Hart's idea has prompted a series of subsequent studies, aiming

Hart's Algorithm

Let R be initially set to TS
Let CS be initially set to an arbitrary prototype per class

Repeat
  For all $x \in R$ do
    if x is misclassified using the 1-NN rule with CS then
      Let $R = R - \{x\}$
        Let $CS = CS \cup \{x\}$
    endif
  endfor
Until there are no changes or R is exhausted

Fig. 1. Algorithmic description of the method proposed by Hart.[13]

at solving these problems without changing the main idea (e.g. Refs. 2, 7, 8, 11 and 25).

As a particular example, the Gowda–Krishna Algorithm (GKA)[7] consists of a straightforward extension in which prototypes are taken into account according to their closeness to the decision boundaries. The position of a prototype in the ordered list of neighbors of its Nearest Neighbor from an Opposite class (NNO) is used as a way to measure closeness to boundaries.[7] Another particularly interesting approach is Tomek's Second Algorithm (TSA)[25] in which Gabriel graphs[26,27] are implicitly used to identify pairs of prototypes that are close to the decision boundaries. The way in which pairs of prototypes are selected makes the algorithm very good at preserving the original decision boundaries.

2.2. *The selective subset*

Ritter *et al.*[23] extended Hart's condensing idea by introducing a condition stronger than consistency in order to be able to search for prototypes in an order independent and more convenient way. They define a subset, SS, as *selective* if

1. it is consistent;
2. all prototypes in the original TS are nearer to a selective neighbor (a member of SS) of the same class than to any prototype from a different class in TS.

This second condition (which in fact makes the first unnecessary) is the main difference between the condensed and the selective subsets. The consistency definition of Hart (condition 1) could be formulated exactly as condition 2 in the following way:

> All prototypes in the original TS must be nearer to a condensed neighbor (a member of CS) of the same class than to any prototype from a different class in CS.

We will refer later to this second condition as the *selective property*. Ritter *et al.* included minimality as a part of the previous definition but we will make here the distinction between any selective set and minimal selective subsets.

An interesting fact which represents another remarkable difference with regard to Hart's consistency, is that prototypes for the selective subset can be selected (or not) independently in each class. This is because the selective property refers only to the *nearest enemy* which need not be another member of the selective subset. Consequently, the search of the nearest enemy is independent of selective prototypes from different classes. The nearest enemy of x_i is the training pattern y that has been found as the nearest neighbor of x_i when considering only those training patterns from all the classes other than that of x_i. This concept has been extensively used in the literature with different names as, for example, Nearest Unlike Neighbor (NUN)[8] or Nearest Neighbor from the Opposite class (NNO).[25]

Ritter *et al.* introduced the following definitions:

(a) a prototype x_j is a *related neighbor* of another prototype x_i, both from the same class, if x_j is closer to x_i than the nearest enemy of x_i;
(b) the set of all related neighbors of x_i is represented by Y_i, the *relative neighborhood* of x_i;
(c) the (minimum) selective subset will be the smallest subset of TS which contains at least one member of Y_i for each prototype x_i in TS.

Although Ritter *et al.*, in their paper stated the importance of selecting "patterns near the decision boundaries", the best approximation to these boundaries is not guaranteed in their procedure because of this third definition which gives precedence to minimality.

To get this minimal selective subset, Ritter *et al.* proposed a backtracking algorithm which can also be seen as a branch and bound procedure that systematically searches for possible solutions in a directed way. This is an algorithm very extensive and complex (in memory and execution time). The original formulation of the algorithm describes the details in terms of a binary $n \times n$ matrix and elementary bit operations involving rows and columns. In the present work, we present it in a (more descriptive) form suitable to later formulate our alternative proposal using notation that involves prototypes and sets of prototypes and hiding the implementation details.

Let $\mathrm{TS} = \{x_i\}_{i=1}^{n}$ be the original training set of prototypes and SS be the resulting selective subset initially set as $\mathrm{SS} = \emptyset$. Both subsets are referred to a particular class, since the algorithm processes each class separately from the others. Additionally, we will also use as auxiliary sets:

C: a set of candidates initially set to TS, from which prototypes are taken to construct SS.

S: a set of prototypes that still have to fulfill the selective property. Initially, as SS $= \emptyset$, the set S is also set to TS. (Recall that the goal is that all prototypes in TS must fulfill the selective property at the end with the resulting SS).

The procedure proposed by Ritter *et al.* proceeds then by moving elements from C (candidates) to SS (results) in such a way that those prototypes in S that (as the result of these movements) fulfill the selective property are removed until S is exhausted. As intermediate steps, the algorithm eliminates elements from S and C as much as possible in order to save computation time.

As in the original paper, the (current) relative neighborhood of each prototype x_i will be represented as $Y_i \subseteq C$. Additionally, in this work, the set of prototypes with respect to which x_j is a relative neighbor will be denoted as $S_j \subseteq S$. That is, $S_j = \{x_k \in S | x_j \in Y_k\}$. This set will be later referred to as the *inverse* relative neighborhood of x_j.

The idea of the auxiliary sets Y_i and S_j (which correspond to columns and rows of the auxiliary matrix in the original description of the algorithm) is illustrated in Fig. 2, in which both sets are represented.

With the previous definitions, the procedure to obtain the minimum selective subset can be informally described as follows (a slightly more formal algorithmic description of the procedure is given in Fig. 4):

1. Identify candidate prototypes (that is, from C), x_j, which are the only (remaining) relative neighbors of some x_i in S. Add x_j to SS (and take it out from C) because this is the only possibility for x_i to fulfill the selective property. Also, as x_j is now in SS, remove from S all prototypes in S_j (among which there is x_i) that will now satisfy the selective property due to x_j.
2. Identify candidate prototypes, x_j in C, whose inverse relative neighborhood, S_j, is included in some other S_k. If this is the case, x_k will fulfill the property (at

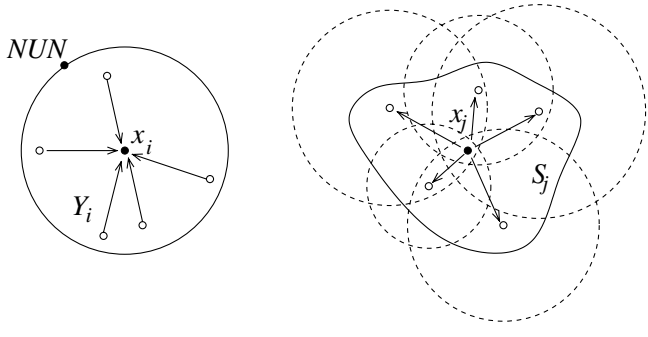


Fig. 2. Graphical representation of sets Y_i and S_j. The relative neighborhoods Y_i are represented by circles centered at the corresponding prototype and whose radii are defined by the corresponding nearest enemies (NUN). The inverse relative neighborhood S_j is shown as a region that contains those centers whose circles intersect the corresponding prototype.

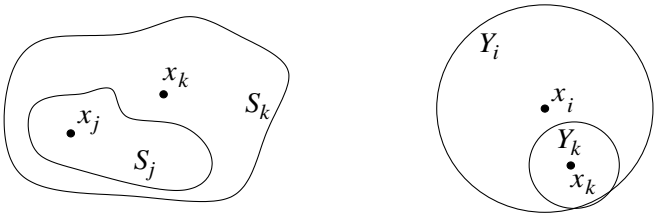


Fig. 3. (a) (Step 2) Inverse relative neighborhood S_j is contained into another inverse relative neighborhood S_k: x_j need not be considered as a candidate any more. x_k will do so. (b) (Step 3) Relative neighborhood Y_i includes another relative neighborhood Y_k: x_i need not fulfill the selective property. It will happen when x_k fulfills it.

SELECTIVE (S, C, SS: Sets) returns subset of C

1. For all $x_j \in C$ such that $\exists\, i : Y_i = \{ x_j \}$ do
   $SS \leftarrow SS \cup \{x_j\}$; $C \leftarrow C - \{x_j\}$; $S \leftarrow S - S_j$;
2. For all $x_j \in C$ such that $\exists\, k : S_j \subseteq S_k$ do
   $C \leftarrow C - \{x_j\}$
3. For all $x_i \in S$ such that $\exists\, k : Y_k \subseteq Y_i$ do
   $S \leftarrow S - \{x_i\}$

4. if $S = \varnothing$ then return SS
   if there have been changes in S and C during steps 1-3, go to 1

5a. Compute a lower bound M_j on the number of prototypes needed to complete the solution if each x_j in C is put in SS. Let M_j be the size of the minimum subset Z from C such that $\sum_{k:x_k \in Z} |S_k - S_j| = |S - S_j|$. Let $M = \min_j (M_j)$.

Repeat

5b. For all $x_j \in C$ such that $M_j = M$ do
$SS' \leftarrow$ **SELECTIVE** ($S - S_j$, $C - \{x_j\}$, $SS \cup \{x_j\}$)
if $|SS'| = |SS| + M$ then return SS'
else if $|SS'| = |SS| + M + 1$ *then* save SS' as SS''

5c. If there is a saved solution then return SS''
else Let $M \leftarrow M + 1$

until $M > |S|$

Fig. 4. Algorithmic description of the method proposed by Ritter *et al.*[23]

least) of the same prototypes in S than x_j so that we can safely remove the latter from the candidate set, C. This situation is illustrated in Fig. 3(a).

3. Identify prototypes x_i that still do not have a relative in SS (that is, from S) whose relative neighborhood, Y_i, contains another Y_k. If this is the case, when

selecting a relative neighbor of x_k, it will be also a relative neighbor of x_i and it will be implicitly taken into account by x_k. (That is, x_i will fulfill the property when the current SS ensures x_k fulfills it.) Consequently, remove x_i from S. This situation is also graphically illustrated in Fig. 3(b).

4. Stop with the current SS as a result if S is exhausted. Otherwise repeat steps 1–3 until there are no changes in S and C.

(a) Compute M_j for each x_j as a lower bound on the minimum number of candidate prototypes still needed to globally fulfill the selective property if x_j is included in SS. This bound is computed by selecting the minimum number of prototypes whose resulting inverse neighborhood sizes, $|S_k - S_j|$, sum up to at least the resulting number of prototypes in S, that is, $|S - S_j|$. Let M be the minimum of the M_j.

(b) For each x_j such that $M_j = M$, tentatively add it to SS (and update C and S accordingly) and backtrack (go to 1). Return if a solution is obtained with exactly M more prototypes.

(c) If no solution with M has been found in (b) but there is one with $M+1$, return it. Otherwise, increase M, and go to (b).

The $n \times n$ matrix proposed by Ritter *et al.* efficiently represents the sets S, C and all Y_i and S_j at the same time. It is worth noting that Y_i consists of elements from C. That is, the set of possible candidates to SS. Consequently, S_j contains elements from S, the set of prototypes waiting to fulfill the selective property. As C and S change in the algorithm, all Y_i and S_j need to be recomputed or efficiently represented correspondingly.

The step numbers in both descriptions of the algorithm correspond to the same numbers in the original formulation of the algorithm. The fact that it is a backtracking algorithm is put forward by writing it as a recursive procedure whose parameters are assumed to be passed by value. This description makes explicit the complexity of the algorithm in the worst case which is exponential. The spatial complexity of the algorithm cannot be assessed from our formulation because that is much more dependent on the implementation. Nevertheless, it must be noted that, according to the implementation proposed by Ritter *et al.*, a large (binary) matrix that contains information about all related neighbors to any prototype must be defined and processed.

3. The Modified Selective Subset

Although there is not any proof, it is quite likely that the algorithm of Ritter *et al.* really gives a minimal selective subset as a solution because it really performs an implicit exhaustive search. In fact, it has been proved that the problem of selecting the minimum selective subset is NP-complete.[28] The aim of this algorithm is the same as in several approaches which try to obtain minimal consistent subsets,[5,8,17] but using instead the selective property.

Although challenging, minimality is not necessarily a good property for real problems in practice. A slightly larger subset of (selective or not) prototypes can represent more accurately the original (optimal) decision boundaries. Accordingly, an adequate size reduction method must consider both storage requirement decrease and classification accuracy with the obtained reduced subset. The convenience of using minimality (with regard to consistency) has also been recently discussed.[9,30]

In the reduction technique here proposed, the MSS rests upon a modification of the definition of the Selective Subset in the preceding section. We keep definitions (a) and (b) as in the work of Ritter *et al.*, but definition (c) is changed in the following way:

> (c′) the modified selective subset (MSS) is defined as that subset of TS which contains, for every x_i in TS, that element of its Y_i that is the nearest to a class other than that of x_i (that is, the closest to its nearest enemy).

The main purpose of this modification is to strengthen the condition to be fulfilled by the reduced subset in order to attain a better approximation of the decision boundaries. Also, when stated in this way, it is possible to introduce a greedy algorithm which tries to obtain selective prototypes in such a way that preference is given to training patterns which lie close to the original (as defined by the whole TS) NN decision boundaries. This algorithm constitutes an efficient alternative to the Selective algorithm of Ritter *et al.* and is usually able to select better (closer to the boundary) prototypes as will be empirically shown later. Similar ideas have already been used with different flavors to obtain consistent subsets.[7,25] The criterion that will be used here to measure the closeness to the boundary is the distance to its nearest enemy. Using this measure, it is possible to define the *best* selective subset as the one that contains the *best* related neighbor for each prototype in the TS. In this context, best means lower distance to its nearest enemy.

The proposed algorithm (shown in Fig. 5), selects prototypes from the original TS according to this measure, and updates the sets S and C correspondingly. S and C are here employed with the same meaning as in the preceding section. SS is replaced now by MSS. We use D_j to refer to the distance from x_j to its nearest enemy. An informal description of the algorithm is also included in the same figure using comments.

The conceptual simplicity of the proposed algorithm contrasts with the one in Fig. 4. In fact, the implementation is also much more straightforward. There is no need to compute related neighborhoods or to maintain any matrix in memory. A possible efficient implementation of this algorithm makes use of a sorting algorithm followed by two nested for loops as shown in Fig. 6. Note that in this implementation, the sets C, Y_i and S_j are no longer needed. Recall also that both algorithms, Selective Subset and Modified Selective Subset, are applied to each class separately. Consequently, the value n in the algorithm in Fig. 6 refers to the number of prototypes in a particular class.

```
MSELECTIVE (S, C, MSS: Sets) returns subset of S

while C ≠ ∅ do                          // while candidates remain
Let x_j = argmin (D_k)                  // take the next better
        x_k ∈ C                         // according to D_k
C ← C − {x_j}

if S_j ∩ S ≠ ∅ then                     // if it makes some (new) prototypes
                                        // fulfill the selective property
MSS ← MSS ∪ {x_j}                       // accept it
S ← S − S_j                             // and update S accordingly
endif
endwhile
return MSS
```

Fig. 5. Algorithmic description of the proposed modified selective subset.

```
Sort the prototypes {x_j}_{j=1}^n according to increasing values of D_j.

  For i = 1, …, n do                    //  x_i is the next best

    add ← FALSE
    For j = i, …, n do                  //  check others including x_i
      if x_j ∈ S ∧ d(x_i , x_j) < D_j then // x_j fulfills now the property with x_i
          S ← S − {x_j}
          add ← TRUE
    endfor

    if add then MSS ← MSS ∪ {x_i}       //  add if S has changed

    if S = ∅ then return MSS            //  every x has a relative
  endfor
```

Fig. 6. Efficient implementation of the proposed modified selective algorithm.

Once the prototypes have been sorted, a unique pass through the set suffices to select the above defined best selective subset. At the end of iteration i of the first loop, at least the prototypes 1 to i are guaranteed to fulfill the property. To do this, an inner loop starting at prototype i is used to decide whether or not to include x_i in the solution. This algorithm requires a quadratic number of basic operations like checking set membership, distance calculation or retrieval, and set updates. Consequently, with an appropriate implementation, the proposed method constitutes a worst-case quadratic algorithm to obtain a selective subset. The obtained subset will not be necessarily minimal, but will usually give better classification accuracy, as will be shown in the experiments.

4. Experimental Results

In this section, several experiments with both synthetic and real databases are presented. Firstly, an artificial dataset will be employed to assess the quality of the

decision boundaries produced by different prototype selection algorithms. On the other hand, experiments over different real datasets will be carried out in order to compare the MSS scheme with other condensing methods, both in classification accuracy and reduction rate.

Apart from the proposed MSS algorithm and Ritter's Selective Subset (SS) approach from which MSS has been developed, other prototype selection algorithms have been considered in this experimental section for different reasons. First, Hart's Condensed Set (CS) and GKA because they can be considered as the counterpart to the SS–MSS pair but dealing with consistency instead of selectiveness. Also, TSA because it clearly aims at boundary preserving as our MSS. Finally, Aha's IB2 algorithm[1] which is very popular in the Machine Learning community and DROP3 which has been recently proposed by Wilson and Martinez[29] have also been considered as reference. These two algorithms can be seen as representatives of a straightforward and elaborate approach to prototype selection, respectively. While in most cases, IB2 results are basically indistinguishable from Hart's CS, DROP3 has been shown to give very good and reasonably convenient results in a variety of different situations.[29]

4.1. *Illustrative experiments with a synthetic database*

In order to illustrate the behavior of both selective algorithms with regard to other prototype selection methods, a variation of the XOR problem has been considered. Random bivariate patterns have been generated following a uniform distribution in a square of length two, centered at zero (apart from a strip of width 0.1 along both axes). The patterns have been labeled at each quadrant to reproduce the well-known XOR problem. More specifically, the label of each point (x, y) is computed as $\text{sign}(x) \cdot \text{sign}(y)$. Different training sets of increasing sizes (400; 800; 1,200; and 1,600) have been considered.

The results of applying several different algorithms to TS of size 800 are depicted in Figs. 7(a)–7(f). In this particular illustrative experiment, IB2 results are not shown because they were very similar to Harts's CS.

Figure 7(a) illustrates how consistency only (Hart's procedure) gives rise to an induced boundary which lies relatively far from the optimal one. The main difference between the two selective sets shown in Fig. 7(e) (SS) and Fig. 7(f) (MSS), lies in the fact that MSS manages to obtain more accurate boundaries in this example, at the price of retaining slightly more prototypes. The two approaches that involve boundary closeness measures (TSA and GKA, in Figs. 7(c) and 7(d), respectively) also improve the decision boundaries obtained by Hart's procedure. Particularly interesting is the result given by TSA which gives a very close approximation to the original decision boundary. Nevertheless, this algorithm retains a very large number of prototypes (twice or more than any other techniques). The number of prototypes retained by each one of the algorithms in this particular run was 31, 38, 73, 28, 21 and 34, respectively. Note that SS obtains in fact the best result in

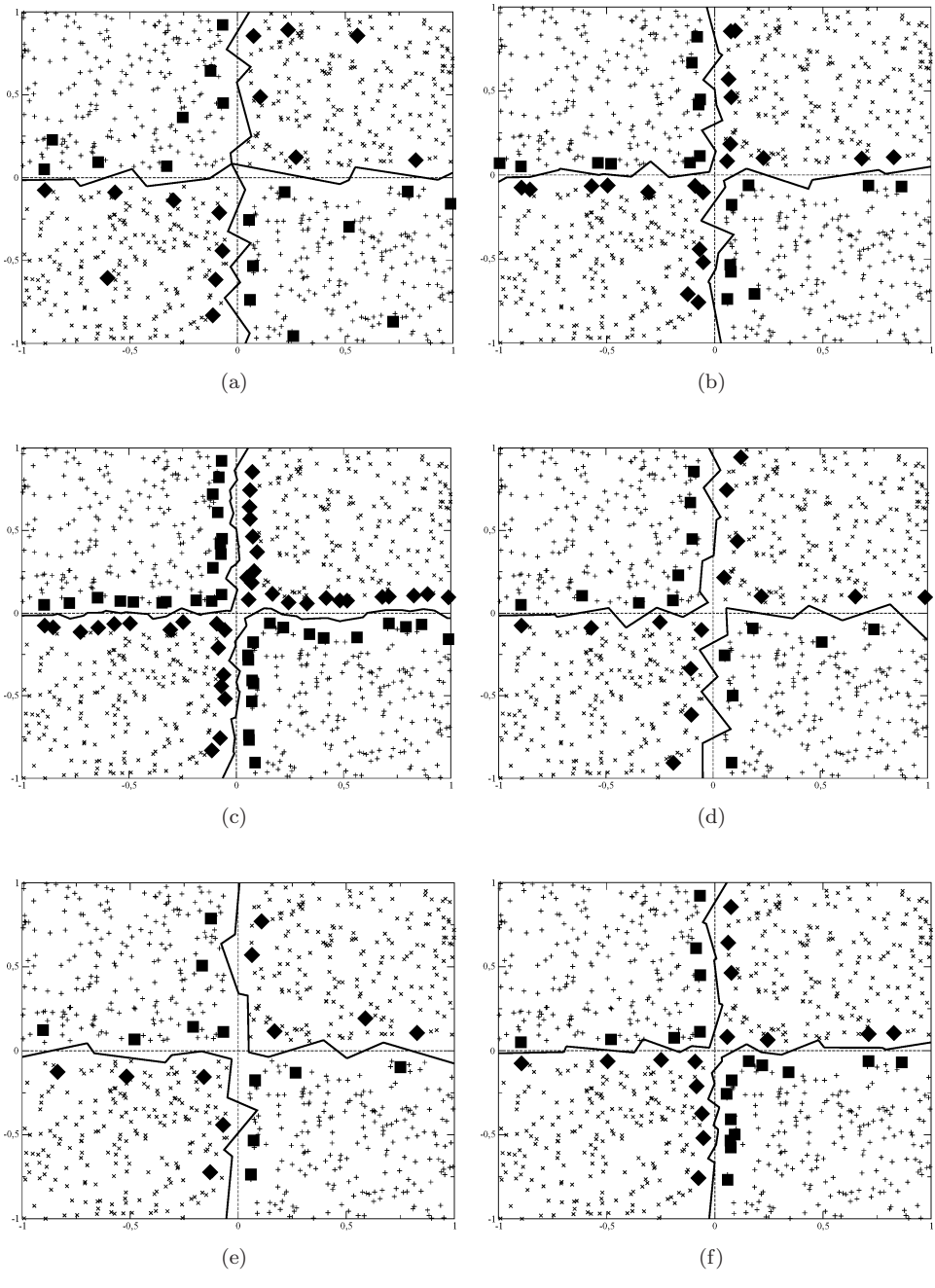
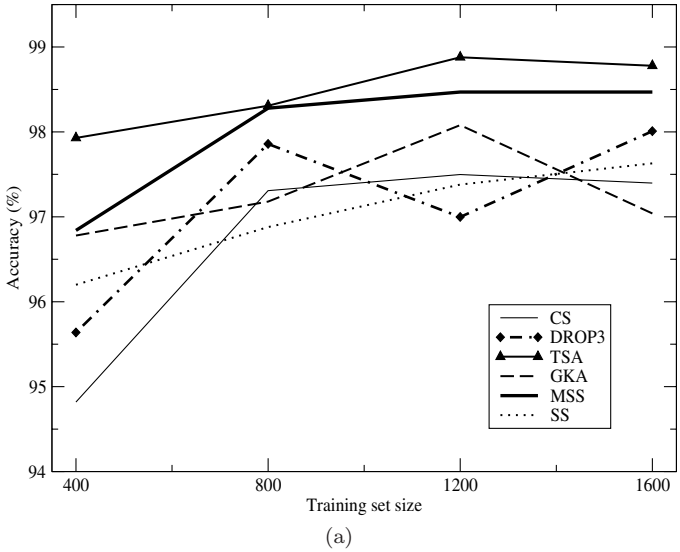


Fig. 7. Subsets of prototypes and induced decision boundaries (thick solid lines) obtained by six different algorithms: (a) CS; (b) DROP3; (c) TSA; (d) GKA; (e) SS; and (f) MSS. The optimal boundaries are also shown as thin dotted lines.

terms of size reduction (as expected). It is also worth noting that the (relatively) high number of prototypes retained by DROP3 in this case is not enough to obtain sufficiently good boundaries. This is partially motivated by the fact that DROP3 is a composite approach aiming both at reducing the training set size and also to clean it.

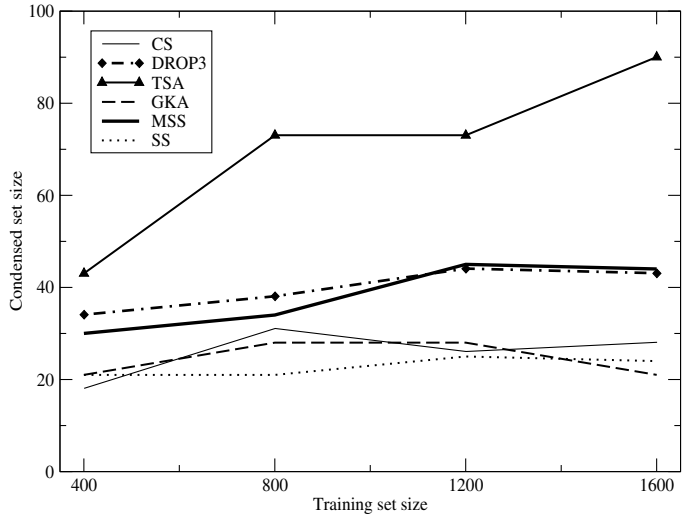
In order to approximately measure the quality of each reduced subset, an independent test set with 40,000 prototypes has been generated. The corresponding accuracies for different TS sizes are depicted in Fig. 8(a) in which higher classification accuracy corresponds to better approximation to the decision boundary. Both, TSA and MSS give consistently the best results for all TS sizes. Figure 8(b) shows the number of retained prototypes in which both MSS and DROP3 exhibit a slightly increasing behavior with regard to CS, GKA and SS approaches. On the other hand, TSA not only retains more prototypes but also its number increases in a quicker way.

Apart from boundary preserving and reduction rate shown in Fig. 8, another aspect that is important to assess the relative merits of prototype selection algorithms is the computing time they need. Figure 9 shows the computing times needed by each prototype selection algorithm with increasing TS sizes. It can be observed that all methods gave similar results for the lowest size, but there is a clear separation in their behavior as the TS size increases. Even though these results cannot be representative of all situations mainly because some of the algorithms are strongly dependent on input data, very different behaviors can be identified. Apart from CS (and IB2) which are basically linear algorithms, the most efficient algorithms (in this and further experiments) were GKA and MSS approaches which have a



(a)

Fig. 8. (a) Classification accuracy and (b) reduced subset size with varying training set sizes.



(b)

Fig. 8. (*Continued*)

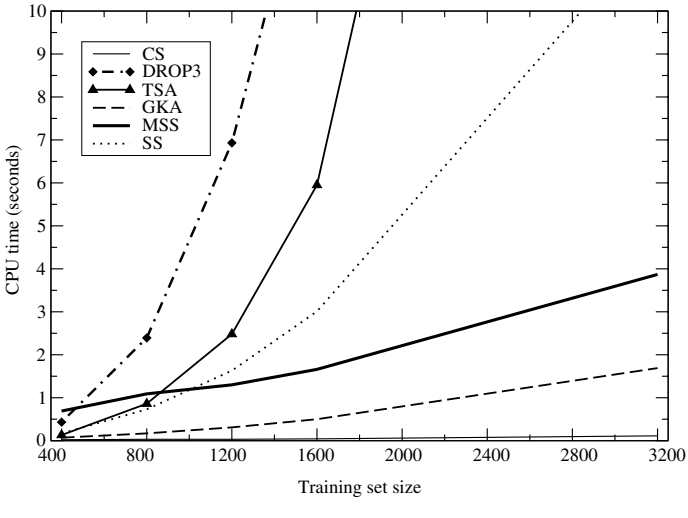


Fig. 9. Computing times for different training set sizes.

worst-case time complexity of $O(n^2)$. The (constant) difference between them can be partially attributed to differences in implementation (GKA was implemented using static memory in C while MSS manages memory dynamically in Java). We used our own implementation of an algorithm for TSA which has an empirical behavior close to $O(n^2)$ as in Refs. 24, 26 and 27. Finally, for SS, IB2, CS and DROP3, we used the implementations referenced in Ref. 29.

4.2. *Experimental evaluations with real datasets*

Experiments with several real datasets taken from the UCI Repository[19] were done to compare some prototype selection algorithms, mainly in classification accuracy. The last dataset (i.e. the Image database) has been previously employed in a real application to locate oranges in outdoor scenes under daylight conditions.[21] These datasets have been selected to cover a wide range of different practical situations. Five-fold cross validation was employed for the experiments in each dataset. Characteristics of the datasets are summarized in Table 1 including the effective training and test sizes used in the experiments.

Table 2 shows the different accuracies obtained by the NN rule using the different subsets of prototypes obtained by each method. The NN rule working with the whole TS is also reported as the baseline classifier.

Experimental results in Table 2 show MSS as the reduction technique with the higher classification accuracy and as the closest to the performance of the NN rule, in most datasets and also in average. More importantly, figures in Table 3 corroborate that MSS is significantly better than CS, IB2 and SS in our experiments. In particular, MSS is significantly better in 8, 8 and 9 datasets, respectively. With regard to GKA, MSS is better in 5 datasets and is never worse. The more elaborate (with an internal cleaning procedure) DROP3 algorithm is better than MSS only in 2 datasets while MSS is significantly better in 4. Finally, MSS is better than TSA in 3 datasets and vice versa which in fact reflects the previously observed behavior. That is, both algorithms manage to accurately preserve the original (1-NN) decision boundaries.

Table 1. Characteristics of the employed real datasets.

	Iris	Glass	Liver	Pima	Vowel	Wine	Vehicle	Texture	Phoneme	Image
TS size	120	170	268	614	422	142	542	4,400	4,324	6,283
Test set size	30	43	67	154	106	36	136	1,100	1,081	1,571
No. classes	3	6	2	2	11	3	4	11	2	3
No. features	4	9	6	8	10	13	18	40	5	2

Table 2. Accuracy rates of the experiments with the reduction algorithms and eight real datasets. Basic NN rule is reported as the baseline classifier.

	Iris	Glass	Liver	Pima	Vowel	Wine	Vehicle	Texture	Phoneme	Image	Average
NN	95.3	67.4	63.8	70.2	96.9	94.7	67.6	99.0	70.3	98.2	82.3
CS	90.6	66.0	60.5	65.4	91.7	94.7	64.0	69.6	66.2	97.2	76.6
SS	90.6	64.3	61.0	66.2	92.9	92.9	65.2	69.5	63.2	97.8	76.4
IB2	90.6	66.5	60.5	65.4	91.2	94.1	64.0	69.5	66.4	97.2	76.5
DROP3	94.7	63.1	64.3	74.3	78.7	93.5	66.7	70.0	72.7	98.1	77.6
TSA	96.0	71.4	65.8	67.2	97.9	73.1	64.4	99.0	69.8	98.4	80.3
GKA	96.0	65.8	63.2	66.0	94.7	69.6	63.3	97.3	68.7	98.1	78.3
MSS	95.3	67.0	63.2	69.5	97.0	93.8	69.1	98.9	67.7	98.4	82.0

Table 3. Confidence levels for the statistical significance (one-tailed t tests) of the differences in accuracy of MSS with the other techniques. Numbers in parentheses indicate lower accuracy of MSS.

	Iris	Glass	Liver	Pima	Vowel	Wine	Vehicle	Texture	Phoneme	Image
NN	no	no	no	no	no	no	no	(0.05)	(0.005)	no
CS	0.005	no	0.005	0.001	0.001	(0.10)	0.001	0.001	0.05	0.01
SS	0.005	0.01	0.005	0.001	0.001	0.10	0.005	0.001	0.001	0.01
IB2	0.001	no	0.001	0.001	0.001	no	0.001	0.001	0.01	0.01
DROP3	no	0.001	no	(0.001)	0.001	no	0.001	0.001	(0.001)	no
TSA	no	(0.005)	no	0.005	no	0.001	0.001	(0.05)	(0.005)	no
GKA	no	no	no	0.001	0.001	0.001	0.001	0.05	no	no

Table 4. Storage requirement (%) after each prototype selection technique.

	Iris	Glass	Liver	Pima	Vowel	Wine	Vehicle	Texture	Phoneme	Image	Average
CS	9.4	41.1	44.5	36.8	23.5	9.2	39.4	6.4	15.9	0.1	22.6
SS	12.1	42.9	52.7	43.2	21.4	14.3	44.3	5.7	16.8	0.3	25.4
IB2	9.1	39.5	44.5	36.8	23.6	9.1	39.5	6.4	15.6	0.1	22.4
DROP3	12.4	23.1	23.9	16.7	46.4	15.9	24.8	7.0	12.4	0.2	18.3
TSA	28.2	84.7	97.5	85.6	95.1	54.2	92.7	66.5	53.6	1.9	66.0
GKA	12.8	46.4	61.0	49.2	22.7	39.7	64.9	7.2	20.7	0.1	32.5
MSS	17.7	58.8	57.5	49.0	69.3	22.9	78.8	21.9	24.0	0.3	40.0

On the other hand, results concerning size reduction shown in Table 4, indicate that the two algorithms that are the best in preserving the decision boundaries (MSS and TSA) produce reduction rates not very impressive compared to the minimum sizes obtained. It is evident that the worst behavior corresponds to the TSA technique. This algorithm retains 66% of the initial prototypes in average. This behavior is similar to that shown in Fig. 8(b) corresponding to the artificial dataset.

5. Some Conclusions and Future Work

This paper introduces a method for effective reduction of the TS size. The proposed method (Modified Selective Subset) is primarily based on the concept of selective subset (that satisfies a condition stronger than consistency) and tries to keep the decision boundaries associated with the selected subset as close to the original ones as possible. Consequently, the most important aim of this method is to obtain an appropriate trade-off between preserving the classification accuracy of the NN rule and yielding a close to minimal training subset. As a very interesting subproduct, the greedy algorithm proposed for obtaining the reduced subset is quite simple in terms of storage and computational time requirements.

The experiments reported in this paper show the convenience of the Modified Selective Subset technique with regard to other reduction methods for several real and representative datasets. These experimental results corroborate that the proposed prototype selection method yields classification accuracy close to that obtained when employing the whole TS. Still more empirical evaluation is needed

to properly assess the relative merits of the proposed method in a variety of different situations in practice. Also, comparison with other reduction algorithms not considered here could be of interest. In particular, reduction algorithms such as those aimed at merging several training patterns[6] or at adaptively modifying the location of some instances,[14] and those based on the genetic algorithms approach.[18,31] Comparison and combination with other nonparametric models constitutes another line of further work that has already been started.[10]

Acknowledgments

This work has been partially supported by grants 44225 from the Mexican SEP-Conacyt, 744.99P from the Mexican Cosnet, TIC2003-08496 from the Spanish CICYT, SAB2003-0106 from the Spanish MECD and P1-1B2002-07 from the Fundació Caixa Castelló — Bancaixa. We would like to thank D.R. Wilson and T.R. Martinez for the possibility to employ source code of the IB2 and DROP3 algorithms, available at ftp://axon.cs.byu.edu/pub/randy/ml/drop/.

References

1. D. W. Aha, D. Kibler and M. K. Albert, Instance-based learning algorithms, *Mach. Learn.* **6** (1991) 37–66.
2. E. Alpaydin, Voting over multiple condensed nearest neighbors, *Artif. Intell. Rev.* **11** (1997) 115–132.
3. J. C. Bezdek, T. R. Reichherzer, G. E. Lim and Y. Attikiouzel, Multiple prototype classifier design, *IEEE Trans. Syst. Man Cybern.* **28**(1) (1998) 67–79.
4. H. Brighton and C. Mellish, Advances in instance selection for instance-based learning algorithms, *Data Min. Knowl. Discov.* **6** (2002) 153–172.
5. V. Cerverón and F. J. Ferri, Another move toward the minimum consistent subset: a tabu search approach to the condensed nearest neighbor rule, *IEEE Trans. Syst. Man Cybern.* Part B **31**(3) (2001) 408–413.
6. C. L. Chang, Finding prototypes for nearest neighbor classifiers, *IEEE Trans. Comput.* **23**(11) (1974) 1179–1184.
7. K. Chidananda-Gowda and G. Krishna, The condensed nearest neighbor rule using the concept of mutual nearest neighborhood, *IEEE Trans. Inform. Th.* **25**(4) (1979) 488–490.
8. B. V. Dasarathy, Minimal consistent set (MCS) identification for optimal nearest neighbor decision systems design, *IEEE Trans. Syst. Man Cybern.* **24**(3) (1994) 511–517.
9. V. S. Devi and M. N. Murty, An incremental prototype set building technique, *Patt. Recogn.* **35**(2) (2002) 505–513.
10. E. Gasca and R. Barandela, Influencia del preprocesamiento de la muestra de entrenamiento en el poder de generalización del perceptron multicapa, in *Proc. 6th Brazilian Symposium on Neural Networks*, Río de Janeiro (2000).
11. G. Gates, The reduced nearest neighbor rule, *IEEE Trans. Inform. Th.* **18**(3) (1972) 431–433.
12. Y. Hamamoto, S. Uchimira and S. Tomita, A bootstrap technique for nearest neighbor classifier design, *IEEE Trans. Patt. Anal. Mach. Intell.* **19**(1) (1997) 73–79.
13. P. E. Hart, The condensed nearest neighbor rule, *IEEE Trans. Inform. Th.* **6**(4) (1968) 515–516.

14. Y. S. Huang, K. Liu and C. Y. Suen, A new method of optimizing prototypes for nearest neighbor classifiers using a multilayer perceptron, *Patt. Recogn. Lett.* **16** (1995) 77–82.
15. B. Karacali and H. Krim, Fast minimization of structural risk by nearest neighbor rule, *IEEE Trans. Neural Networks* **14**(1) (2002) 127–137.
16. T. Kohonen, *Self-Organizing Maps* (Springer, Berlin, 1995).
17. L. I. Kuncheva and J. C. Bezdek, Nearest prototype classification: clustering, genetic algorithms, or random search? *IEEE Trans. Syst. Man Cybern.* **28**(1) (1998) 160–164.
18. L. I. Kuncheva and L. C. Jain, Nearest neighbor classifier: simultaneous editing and feature selection, *Patt. Recogn. Lett.* **20** (1999) 1149–1156.
19. C. J. Merz and P. M. Murphy, *UCI Repository of Machine Learning Databases*, University of California Irvine (1996).
20. R. A. Mollineda, F. J. Ferri and E. Vidal, An efficient prototype merging strategy for the condensed 1-NN rule through class-conditional hierarchical clustering, *Patt. Recogn.* **35**(12) (2002) 2771–2782.
21. F. Pla, F. Juste, F. J. Ferri and M. Vicens, Colour segmentation based on a light reflection model to locate citrus fruits for robotic harvesting, *Comput. Electron. Agricul.* **9** (1993) 53–70.
22. T. Reinartz, A unifying view on instance selection, *Data Min. Knowl. Discov.* **6** (2002) 191–210.
23. G. L. Ritter, H. B. Woodruff, S. R. Lowry and T. L. Isenhour, An algorithm for selective nearest neighbor decision rule, *IEEE Trans. Inform. Th.* **21**(6) (1975) 665–669.
24. J. S. Sánchez, F. Pla and F. J. Ferri, Prototype selection for the nearest neighbor rule through proximity graphs, *Patt. Recogn. Lett.* **18** (1997) 507–513.
25. I. Tomek, Two modifications of CNN, *IEEE Trans. Syst. Man Cybern.* **7**(2) (1976) 769–772.
26. G. T. Toussaint, A counterexample to Tomek's consistency theorem for a condensed nearest neighbor rule, *Patt. Recogn. Lett.* **15** (1994) 797–801.
27. G. T. Toussaint, Proximity graphs for nearest neighbor decision rules: recent progress, in *Proc. 34th Symp. Computing and Statistics*, Montreal (2002).
28. G. Wilfong, Nearest neighbor problems, *Int. J. Comput. Geom. Appl.* **2**(4) (1992) 383–416.
29. D. R. Wilson and T. R. Martinez, Reduction techniques for instance-based learning algorithms, *Mach. Learn.* **38**(3) (2000) 257–286.
30. H. Zhang and G. Sun, Optimal reference subset selection for nearest neighbor classification by tabu search, *Patt. Recogn.* **35**(7) (2002) 1481–1490.
31. Q. Zhao and T. Higuchi, Minimization of nearest neighbor classifiers based on individual evolutionary algorithm, *Patt. Recogn. Lett.* **17** (1996) 125–131.

Ricardo Barandela (1938–2004) received a B.Sc. in mathematics from the Universidad de La Habana (Cuba) in 1975 and a Ph.D. in mathematics from the Institute for Cybernetics (Berlin, Germany) in 1987. He held a number of academic positions in the Universidad de La Habana and in the Cuban Department of Science and Technology. He was also a Senior Researcher in the Department of Computer Science at the Instituto Tecnológico de Toluca (Metepec, México), where he created the Pattern Recognition Lab in 1999. He authored a large number of scientific publications and was a member of IAPR and ACRP (Cuban Association for Pattern Recognition). He also served as a referee in well-reputed journals and conferences.

His research interests were primarily in the field of pattern recognition and data mining; specifically, the development of learning techniques for supervised methods, the study of combination of classifiers, and instance selection techniques.

Francesc J. Ferri is an Associate Professor in the Department of Computer Science at Universitat de València (Burjassot, Spain). He received a B.Sc. and a Ph.D. in physics from the Universitat de València in 1987 and 1993, respectively. He has been involved in a number of scientific and technical projects on computer vision and pattern recognition. He has authored a number of scientific papers in international conferences and well-established journals in his fields of interest. He has also helped in refereeing for several of these journals and major conferences. He has been a visiting scientist with the Vision, Speech and Signal Processing Research group at the University of Surrey (Guilford, UK), at the Institute for Information Theory and Automation (Prague, Czech Republic), at the Digital Imaging Research Center, Kingston University (UK), and at the Pattern Recognition and Image Processing Lab, Michigan State University (East Lansing, USA).

His current research interests include feature selection, nonparametric classification methods, prototype-based learning, computational geometry, and image analysis. He is a member of ACM, IAPR and AERFAI (Spanish Association of Pattern Recognition and Image Analysis).

J. Salvador Sánchez is an Associate Professor in the Department of Programming Languages and Information Systems at Universitat Jaume I (Castelló de la Plana, Spain) since 1992, and is currently the head of the Pattern Recognition Section in the Computer Vision Lab. He received a B.Sc. in computer science from the Universidad Politécnica de Valencia in 1990 and a Ph.D. in computer science engineering from Universitat Jaume I in 1998. He is a member of IEEE Signal Processing Society, IEEE Neural Networks Society, IEEE Information Theory Society, IAPR, AERFAI (Spanish Association of Pattern Recognition and Image Analysis), ECCAI, and AEPIA (Spanish Association for Artificial Intelligence). He is author or co-author of more than 70 scientific publications, co-editor of two books and guest editor of three special issues in international journals. He has been a visiting scientist at the Instituto Tecnológico de Toluca (Metepec, México) and at the University of Wales (Bangor, UK).

His current research interests lie in the areas of pattern recognition and machine learning, including nonparametric classification, feature and prototype selection, ensembles of classifiers, and decision tree induction.