

## COPYRIGHT NOTICE

© 2007 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

This is the author's version of the work. The definitive version was published in *IEEE Transactions on Knowledge and Data Engineering* (TKDE), 19(11):1450-1464, November 2007.

DOI: <http://dx.doi.org/10.1109/TKDE.2007.190645>.

# Fast Nearest Neighbor Condensation for Large Data Sets Classification

Fabrizio Angiulli

**Abstract**—This work has two main objectives, namely, to introduce a novel algorithm, called the Fast Condensed Nearest Neighbor (FCNN) rule, for computing a training set consistent subset for the nearest neighbor decision rule, and to show that condensation algorithms for the nearest neighbor rule can be applied to huge collections of data. The FCNN rule has some interesting properties: it is order independent, its worst case time complexity is quadratic but often with a small constant pre-factor, and it is likely to select points very close to the decision boundary. Furthermore, its structure allows for the triangle inequality to be effectively exploited to reduce the computational effort. The FCNN rule outperformed even here enhanced variants of existing competence preservation methods both in terms of learning speed and learning scaling behavior, and often in terms of the size of the model, while it guaranteed the same prediction accuracy. Furthermore, it was three order of magnitude faster than hybrid instance-based learning algorithms on the MNIST and MIT Face databases and computed a model of accuracy comparable to that of methods incorporating a noise filtering pass.

**Index Terms**—Classification, large and high-dimensional data, nearest neighbor rule, prototype selection algorithms, training-set consistent subset.



## 1 INTRODUCTION

The nearest neighbor decision rule [10] (NN rule for short) assigns to an unclassified sample point the classification of the nearest of a set of previously classified points. For this decision rule, no explicit knowledge of the underlying distributions of the data is needed. A strong point of the nearest neighbor rule is that, for all distributions, its probability of error is bounded above by twice the Bayes probability of error [10], [27], [16]. That is, it may be said that half the classification information in an infinite size sample set is contained in the nearest neighbor.

Naive implementation of the NN rule requires storage of all the previously classified data, and then comparison of each sample point to be classified to each stored point. In order to reduce both space and time requirements, several techniques to reduce the size of the stored data for the NN rule have been proposed (see [32] and [28] for a survey) referred to as training set reduction, training set condensation, reference set thinning, and prototype selection algorithms. In particular, among these techniques, *training set consistent* ones, aim at selecting a subset of the training set that classifies the remaining data correctly through the NN rule.

Using a training set consistent subset, instead of the entire training set, to implement the NN rule, has the additional advantage that it may guarantee better classification accuracy. Indeed, [22] showed that the VC dimension of an NN classifier is given by the number of reference points in the training set. Thus, in order to achieve a classification rule with controlled generalization, it is better to replace the training set with a small

consistent subset. Unfortunately, computing a minimum cardinality training set consistent subset for the NN rule turns out to be intractable [30].

Several training set consistent condensation algorithms have been introduced in the literature. We point out that, among the criteria characterizing condensation methods, learning speed is usually neglected. However, in order to manage huge amounts of data, methods exhibiting good scaling behavior are definitively needed. This work introduces a novel order independent algorithm for finding a training set consistent subset for the NN rule, called the FCNN (Fast Condensed Nearest Neighbor) rule, shows that it is scalable on large multi-dimensional training sets, and compares it with existing condensation methods.

The rest of the paper is organized as follows. In Section 2, previous approaches are described and compared to the approach here proposed and the contribution of this work is illustrated. In Section 3, the FCNN rule is described and its main properties are stated. In Section 4, experimental results are presented together with a thorough comparison with existing methods. Finally, in Section 5, strengths and weaknesses of the different condensation methods are discussed and conclusions are drawn.

## 2 RELATED WORKS AND CONTRIBUTION

Starting from the seminal work of [21], several training set condensation algorithms have been introduced in the literature, also known as instance-based [2], lazy [1], memory-based [26], and case-based learners [29]. These methods can be grouped into three main categories depending on the objectives that they want to achieve [8], i.e. *competence preservation*, *competence enhancement*, and *hybrid approaches*.

---

F. Angiulli is with DEIS, University of Calabria, Italy, email: f.angiulli@deis.unical.it

The goal of competence preservation methods is to compute a training set consistent subset removing superfluous instances that will not affect the classification accuracy of the training set. Competence enhancement methods aim at removing noisy instances in order to increase classifier accuracy. Finally, hybrid methods search for a small subset of the training set that, simultaneously, achieves both noisy and superfluous instances elimination.

Competence enhancement and preservation methods can be combined in order to achieve the same objectives as hybrid methods. However, while goals of enhancement and preservation methods are clearly separated, i.e. smoothing the decision boundary in the former case and computing a possibly small training set consistent subset in the latter, often it is not clear how subsets computed by hybrid methods can be related to the sets computed by the two other methods. Thus, searching for a small training set consistent subset is a relevant task *per se*, improving both classification speed and classifier accuracy [22], and computationally hard [30].

Next, we first describe training set consistent subset methods for the NN rule and some other related work, and then point out the contributions of this work. A detailed survey of condensation methods can be found in [32] and [28].

### 2.0.1 Condensed Nearest Neighbor rule

The concept of a training set consistent subset was introduced by P.E. Hart [21] together with an algorithm, called the CNN rule (for Condensed NN rule), to determine a consistent subset of the original sample set.

The algorithm uses two bins, called  $S$  and  $T$ . Initially, the first sample of the training set is placed in  $S$ , while the remaining samples of the training set are placed in  $T$ . Then, one pass through  $T$  is performed. During the scan, whenever a point of  $T$  is misclassified using the content of  $S$ , it is transferred from  $T$  to  $S$ . The algorithm terminates when no point is transferred during a complete pass of  $T$ .

The motivation for this heuristic is that misclassified data lies close to the decision boundary. Nevertheless, the CNN rule may select points far from the decision boundary. Furthermore, the CNN is order dependent, that is it has the undesirable property that the consistent subset depends on the order in which the data is processed. Thus, multiple runs of the method over randomly permuted versions of the original training set should be executed in order to settle the quality of its output [3].

### 2.0.2 Modified Condensed Nearest Neighbor rule

The MCNN rule (for Modified CNN rule) [14] computes a training set consistent subset in an incremental manner.

The consistent subset  $S$  is initially set to the empty set. During each main iteration of the algorithm, first the points  $S_t$  of the training set  $T$  misclassified by using

$S$  are selected. Then, the set of centroids<sup>1</sup>  $C$  of the points in  $S_t$  are computed and used to classify  $S_t$ .  $S_t$  is thus partitioned into two sets,  $S_r$  and  $S_s$  of points that are correctly classified and misclassified, respectively, by using the centroids  $C$ . If set  $S_s$  is empty, then the set  $S$  is augmented with the set  $C$ , otherwise  $S_t$  is set to  $S_r$  and the process is repeated until  $S_s$  becomes empty. The algorithm terminates when there are no misclassified points of  $T$  by using  $S$ .

Differently from the CNN rule, the MCNN rule is order independent, that is, it always returns the same consistent subset independently of the order in which the data is processed. As claimed by the authors of the algorithm, the MCNN rule may work well for data with a gaussian distribution or that can be split up into regions with a gaussian distribution, but, in general, it is unlikely to select points close to the decision boundary. Also, during each iteration of the MCNN rule, at most  $m$  points can be added to the subset  $S$ , where  $m$  is the number of classes in  $T$ , thus the method might require a lot of iterations to converge.

### 2.0.3 Structural Risk Minimization Nearest Neighbor rule

In order to compute a small consistent subset  $S$  of the training set  $T$ , [22] proposed the following algorithm (NNSRM for Structural Risk Minimization using the NN rule).

Assume that the training set  $T$  contains only two class labels.<sup>2</sup> First, all pairwise distances among points having different class labels are computed. Let  $\{x_i, y_i\}$  denote the pair having the  $i$ th smallest distance,  $i \geq 1$ . The set  $S$  is initialized to  $\{x_1, y_1\}$ , i.e. to the closest points  $x_1$  and  $y_1$  from the two classes, and a counter  $k$  is initialized to 2. Next, until set  $S$  misclassifies at least a point in  $T$ , the following steps are performed: if  $\{x_k, y_k\}$  is not contained in  $S$ , then  $S$  is augmented with both  $x_k$  and  $y_k$ ; in any case,  $k$  is set to  $k + 1$ . That is, the algorithm performs one pass on the pairs of points from the two classes ordered in increasing distance and until the initially empty set  $S$  misclassifies the training set  $T$ , whenever a pair is not contained in  $S$  it is added to  $S$ .

The intuition underlying the method is that since the nearest pairs of points from the opposite classes lie in the regions where the domains of the two classes are closest, i.e. where most classification errors occur, they must be considered first. However, the method is costly. Indeed, the complexity of the NNSRM algorithm is  $O(|T|^3)$ . Furthermore, we note that the size of the condensed set computed is sensitive to the pairs having the greatest distances. Indeed, assume that two points from opposite

1. Given a set  $S$  of points having the same class label, the *centroid*  $c$  of  $S$  is the point  $c$  of  $S$  which is closest to the geometrical center of  $S$ . The set of centroids  $C$  of a set of points  $S = S_1 \cup \dots \cup S_n$ , where each  $S_i$  ( $1 \leq i \leq n$ ) is a maximal subset of points of  $S$  having the same class label, is the set  $C = \{c_1, \dots, c_n\}$ , where  $c_i$  is the centroid of  $S_i$  ( $1 \leq i \leq n$ ).

2. They provided also a similar algorithm working on data with more than two class labels.

classes in the training set are far from the other training set points, and such that their distance is greater than the distance from any other pair of the remaining points from the opposite classes. Then, a training set consistent subset must contain these two points and, hence, all the training set points.

#### 2.0.4 Reduced Nearest Neighbor rule

The RNN rule [20], for Reduced NN rule, is a post-processing step that can be applied to any other competence preservation method. It works as follows. First, subset  $S$  is set equal to training set  $T$  or to a training set consistent subset of  $T$ . Then, points of  $S$  whose deletion from  $S$  does not leave any point of  $T$  misclassified are eliminated from  $S$ . Experiments have shown that this rule yields a slightly smaller subset than the CNN rule, but it is costly.

#### 2.0.5 Minimum cardinality methods

Methods previously discussed compute a training set consistent subset in an incremental or decremental manner and have polynomial execution time requirements. Next, competence preservation methods whose aim is to compute a minimum cardinality training set consistent subset (an NP-hard task, see [30]) are briefly described.

The SNN rule [25], for Selective NN rule, computes the smallest training set consistent subset  $S$  of  $T$  having the following additional property: each point of  $T$  is closer to a point of  $S$  of the same class than to any point of  $T$  of a different class. This set is also called *selective subset*. This algorithm runs in exponential time (the problem of computing the minimum cardinality selective subset has been proved to be NP-hard [30]) and, hence, it is not suitable on large training sets.

The MCS rule [12], for Minimal Consistent Subset, aims at computing a minimum cardinality training set consistent subset. The algorithm, based on the computation of the so-called nearest unlike neighbors [13], is quite complex. Furthermore, counterexamples have been found to the conjecture that it computes a minimum cardinality subset.

Approximate optimization methods, such as tabu search, gradient descent, evolutionary learning, and others, have been used to compute subsets close to the minimum cardinality one. These algorithms can be applied in a reasonable amount of time only to a small- or medium-sized data set. [24] provides a comparison of a number of these techniques.

#### 2.0.6 Other approaches

Finally, we mention the literature less related to the competence preservation task, but involving complementary or alternative approaches, concerning *decision boundary consistent subset* methods [7], which compute a subset of the original training set preserving the decision boundary, condensing and editing techniques through the use of *proximity graphs* [28], *nearest neighbor search*

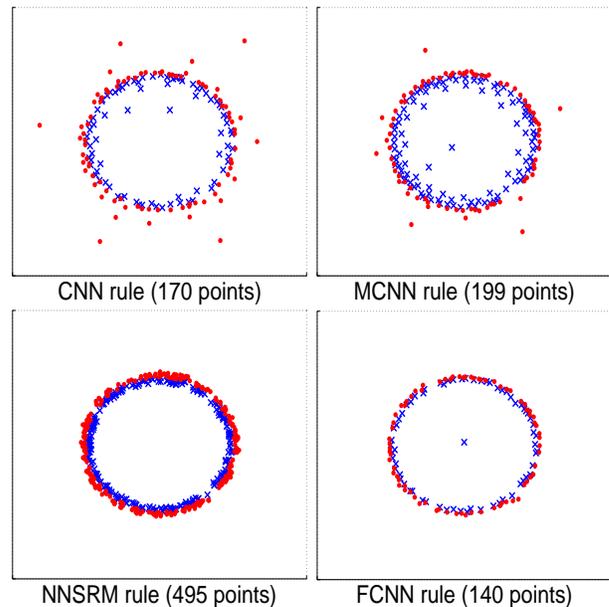


Fig. 1. Example of training set consistent subsets computed by the CNN, MCNN, NNSRM, and FCNN rules.

techniques [11], [19], which can alleviate the cost of searching for the nearest neighbor of a query point at classification time, and *competence enhancement* and *hybrid* methods [32], introduced early in this section.

If the training set is clean, then using a consistent method is very appropriate. In the presence of noise, if the accuracy of the consistent methods is not adequate, they can be combined with competence enhancement methods (e.g., Wilson editing or multi-edit [31], [15]). Alternatively, accuracy may be improved by using the  $k$  nearest neighbor rule [18], [17], the generalization of the nearest neighbor rule in which a new object is assigned to the class with the most members present among its  $k$  nearest neighbors in the training set. Indeed, the probability of error of the  $k$  nearest neighbor rule asymptotically approaches the Bayes error. Accuracy of condensed nearest neighbor classifiers can be also enhanced by combining *multiple classifiers*, as done in [3], where it is proposed to train multiple condensed nearest neighbor classifiers on smaller training sets and to take a vote over them, or in [5], [6], where the MFS algorithm is described, combining multiple nearest neighbor classifiers, each using only a random subset of the features.

## 2.1 Contribution

This work introduces a novel algorithm for the computation of a training set consistent subset for the nearest neighbor rule<sup>3</sup>. The algorithm, called Fast CNN rule (FCNN for short), works as follows.

First, the consistent subset  $S$  is initialized to the centroids of the classes contained in the training set  $T$ . Then,

3. A preliminary version of this work appeared in [4].

during each iteration of the algorithm, for each point  $p$  in  $S$ , a point  $q$  of  $T$  belonging to the Voronoi cell of  $p$ ,<sup>4</sup> but having a different class label, is selected and added to  $S$ . The algorithm stops when no further point can be added to  $S$ , i.e. when  $T$  is correctly classified using  $S$ .

Despite being quite simple, the FCNN rule has some desirable properties. Indeed, it is order independent, its worst case time complexity scales as the product of the cardinality of the training set and of the condensed set, requires few iterations to converge, and it is likely to select points very close to the decision boundary.

For example, Figure 1 compares the consistent subsets computed by the CNN, MCNN, NNSRM, and FCNN rules on a training set composed of 9,000 points uniformly distributed into the unit square and partitioned into two classes by a circle of diameter 0.5.

As already noted elsewhere [32], training set reduction algorithms can be characterized by their storage reduction, classification speed increase, generalization accuracy, noise tolerance, and learning speed. Among these criteria, the learning speed one is usually neglected. However, in order to be practicable on large training sets or in knowledge discovery applications requiring a learning step in their cycle, the method should exhibit good learning behavior.

The contribution of this work can be summarized as follows:

- A novel order independent algorithm, called FCNN rule, for the computation of a training set consistent subset for the nearest neighbor rule is presented.
- It is proved that the worst case time complexity of the FCNN is quadratic with an often small constant pre-factor that corresponds to the reduction ratio. Furthermore, an implementation cleverly exploiting the triangle inequality and sensibly reducing this pre-factor is described.
- It is shown that the FCNN method scales well on large-sized multidimensional data sets.
- The FCNN rule is compared with both standard and here enhanced implementations of existing competence preservation algorithms, showing that it outperforms the other methods in terms of learning speed and learning scaling behavior, and often in terms of size of the model.
- The FCNN rule is compared with hybrid instance-based learning algorithms on the MNIST and MIT Face databases showing that it is at least three order of magnitude faster while guarantees a comparable accuracy.
- An extension of the basic rule taking into account the  $k$  nearest neighbors is presented, and it is shown that the computed subset has the same accuracy of hybrid methods incorporating a noise filtering step.

In its entirety, this is the first work providing nearest neighbor condensation algorithms which have been

4. The Voronoi cell of point  $p \in S$  is the set of all points that are closer to  $p$  than to any other point in  $S$ .

**Algorithm** FCNN rule  
**Input:** A training set  $T$ ;  
**Output:** A training set consistent subset  $S$  of  $T$ ;  
**Method:**  
 $S = \emptyset$ ;  
 $\Delta S = Centroids(T)$ ;  
**while** ( $\Delta S \neq \emptyset$ ) {  
 $S = S \cup \Delta S$ ;  
 $\Delta S = \emptyset$ ;  
**for each** ( $p \in S$ )  
 $\Delta S = \Delta S \cup \{rep(p, Voren(p, S, T))\}$ ;  
**}**  
**return**( $S$ );

Fig. 2. The FCNN rule.

shown experimentally to be scalable on large multidimensional data sets.

### 3 THE FCNN RULE

We start by giving some preliminary definitions.

In the following we denote by  $T$  a labelled training set from a metric space with distance metrics  $d$ .

Let  $p$  be an element of  $T$ . We denote by  $nn(p, T)$  the nearest neighbor of  $p$  in  $T$  according to the distance  $d$ . If  $p \in T$ , then  $p$  itself is its first nearest neighbor. We denote by  $l(p)$  the label associated with  $p$ .

Given a point  $q$ , the NN rule  $NN(q, T)$  assigns to  $q$  the label of the nearest neighbor of  $q$  in  $T$ , i.e.  $NN(q, T) = l(nn(q, T))$ .

A subset  $S$  of  $T$  is said to be a *training set consistent subset* of  $T$  if, for each  $p \in (T - S)$ ,  $l(p) = NN(p, S)$ .

Let  $S$  be a subset of  $T$ , and let  $p$  be an element of  $S$ . We denote by  $Vor(p, S, T)$  the set  $\{q \in T \mid p = nn(q, S)\}$ , that is the set of the elements of  $T$  that are closer to  $p$  than to any other element  $p'$  of  $S$ . Furthermore, we denote by  $Voren(p, S, T)$  the set of the *Voronoi enemies* of  $p$  in  $T$  w.r.t.  $S$ , defined as  $\{q \in Vor(p, S, T) \mid l(q) \neq l(p)\}$ .

We denote by  $Centroids(T)$  the set containing the centroids of each class label in  $T$ .

The following Theorem states the property exploited by the FCNN rule in order to compute a training set consistent subset.

*Theorem 3.1:*  $S$  is a training set consistent subset of  $T$  for the NN rule iff for each element  $p$  of  $S$ ,  $Voren(p, S, T)$  is empty.

*Proof:* ( $\Rightarrow$ ) By contradiction, assume that there is an element  $p$  of  $S$  such that  $Voren(p, S, T)$  is not empty. Then at least an element  $q$  of  $Voren(p, S, T)$ , and hence of  $T$ , is such that  $NN(q, S) = l(nn(q, S)) = l(p) \neq l(q)$ , and  $S$  is not training set consistent.

( $\Leftarrow$ ) First of all, note that  $(\cup_{p \in S} Vor(p, S, T)) = T$ . Thus, for each  $q \in T$ , there is  $p \in S$  such that  $q \in Vor(p, S, T)$ , and, being  $Voren(p, S, T)$  empty, it holds that  $l(nn(q, S)) = l(p) = l(q)$ .  $\square$

#### 3.1 Algorithm

The algorithm FCNN rule is shown in Figure 2. It initializes the consistent subset  $S$  with a seed element

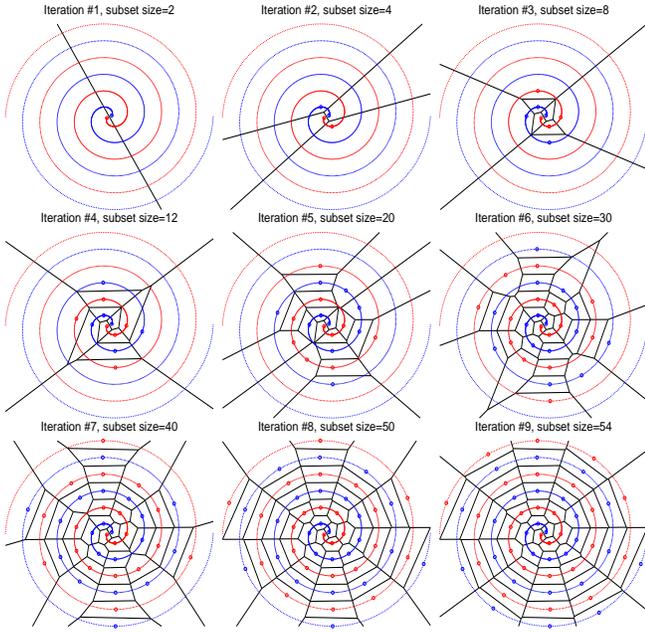


Fig. 3. Example of execution of the FCNN1 rule.

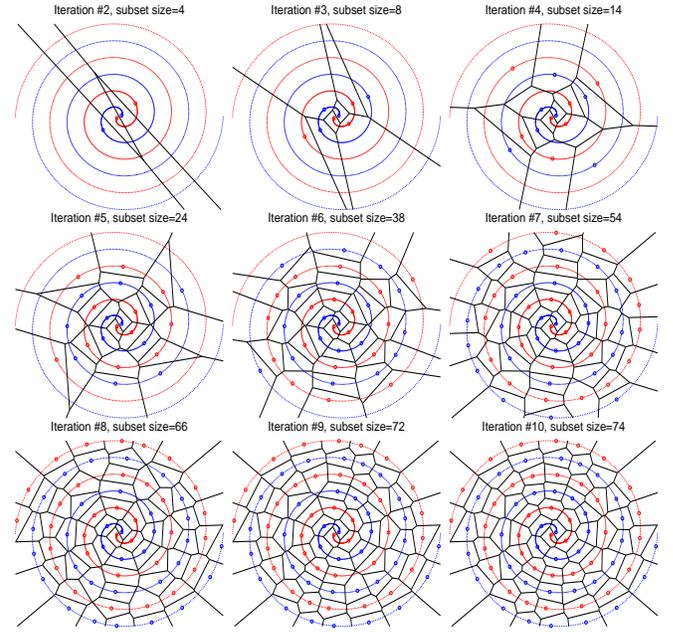


Fig. 4. Example of execution of the FCNN2 rule.

from each class label of the training set  $T$ . In particular, the seeds employed are the centroids of the classes in  $T$ .

The algorithm works in an incremental manner: during each iteration the set  $S$  is augmented until the stop condition, given by Theorem 3.1, is reached. For each element of  $S$ , a representative element of  $\text{Voren}(p, S, T)$  w.r.t.  $p$ , denoted as  $\text{rep}(p, \text{Voren}(p, S, T))$  in Figure 2, is selected and inserted into  $S$ .

Given  $p \in T$  and a subset  $X \subseteq T$ , the representative  $\text{rep}(p, X)$  of  $X$  w.r.t.  $p$  can be defined in different ways. We investigate the behavior of two different definitions for  $\text{rep}(p, X)$ . The first definition, we call FCNN1 the variant of the FCNN rule using this definition, selects the nearest neighbor of  $p$  in  $X$ , that is  $\text{rep}(p, X) = \text{nn}(p, X)$ . The second definition, we call FCNN2 the variant of the FCNN rule using it, selects the class centroid in  $X$  closest to  $p$ , that is  $\text{rep}(p, X) = \text{nn}(p, \text{Centroids}(X))$ .

If, during an iteration, no new element can be added to  $S$ , then, by Theorem 3.1,  $S$  is a training set consistent subset of  $T$ , and the algorithm terminates returning the set  $S$ .

Figures 3 and 4 report an example of execution of the FCNN1 and FCNN2 rules on the same data set. The data set considered is composed of 2,000 points on the plane. Half of the points belong to one of two concentric spirals representing two distinct classes. In this case, the FCNN1 rule performs nine iterations and returns a subset composed of 54 points, while the FCNN2 rule performs ten iterations and computes a subset composed of 74 points. It is clear from these figures that the FCNN rule requires few iterations to converge to a solution. Indeed, the number of points contained in the subset could double at the end of each iteration. This is due to the fact that, for each point  $p$  such that  $\text{Voren}(p, S, T)$  is not empty, a new point is added to the set  $S$ .

As discussed above, the FCNN1 and FCNN2 rules build the set  $\Delta S$  by selecting a representative of the Voronoi enemies of all the points in the current subset  $S$ . It is thus of interest to analyze the behavior of the FCNN rule in the special case in which the set  $\Delta S$  is constrained to be composed of one single point per iteration, i.e. in the special case in which  $\Delta S$  is built by selecting the representative of the Voronoi enemies of only one of the elements of  $S$ .

Such an element can be defined in different ways, but a natural choice is to select the point  $p^*$  of  $S$  such that  $|\text{Voren}(p^*, S, T)|$  is maximum, that is, the point having most Voronoi enemies. In the following, we will call the FCNN3 rule the variant of the FCNN rule whose set  $\Delta S$  is built by selecting only point  $\text{nn}(p^*, \text{Voren}(p^*, S, T))$ , and FCNN4 the variant whose set  $\Delta S$  is built by selecting only point  $\text{nn}(p^*, \text{Centroids}(\text{Voren}(p^*, S, T)))$ . Differently from the FCNN1 and FCNN2, both the FCNN3 and FCNN4 initialize the set  $S$  by using only the centroid of the most populated class.

The following theorem states the main properties of the algorithm.

*Theorem 3.2:* The algorithm FCNN rule (i) terminates in a finite time, (ii) computes a training set consistent subset, and (iii) is order independent.

The proof of Theorem 3.2 is reported in the Appendix. Figure 5 shows the implementation of the FCNN1 rule. For the sake of clarity, the treatment of the ties described in Theorem 3.2 is not reported in the figure.

The algorithm maintains in the array *nearest*, for each training set point  $q$ , the closest point  $\text{nearest}[q]$  of  $q$  in the set  $S$ , and in the array *rep*, for each point  $p$  in  $S$ , its current representative  $\text{rep}[p]$  of the misclassified points lying in the Voronoi cell of  $p$ .

During each iteration, the array *nearest* and *rep* must

```

Algorithm FCNN1 rule
Input: A training set  $T$ ;
Output: A training set consistent subset  $S$  of  $T$ ;
Method:
  for each ( $p \in T$ )
     $nearest[p] = \text{undefined}$ ;
   $S = \emptyset$ ;
   $\Delta S = \text{Centroids}(T)$ ;
  while ( $\Delta S \neq \emptyset$ ) {
     $S = S \cup \Delta S$ ;
    for each ( $p \in S$ )
       $rep[p] = \text{undefined}$ ;
    for each ( $q \in (T - S)$ ) {
      for each ( $p \in \Delta S$ )
        if ( $d(nearest[q], q) > d(p, q)$ )
           $nearest[q] = p$ ;
        if ( $(l(q) \neq l(nearest[q])) \ \&\&$ 
          ( $d(nearest[q], q) < d(nearest[q], rep[nearest[q]])$ ))
           $rep[nearest[q]] = q$ ;
      }
     $\Delta S = \emptyset$ ;
    for each ( $p \in S$ )
      if ( $rep[p]$  is defined)  $\Delta S = \Delta S \cup \{rep[p]\}$ ;
  }
  return( $S$ );

```

Fig. 5. The FCNN1 rule.

be updated. Let  $\Delta S$  be the set of points added to the set  $S$  at the beginning of the current iteration. To update the array  $nearest$ , it is sufficient to compare the training set points in  $(T - S)$  with the points in the set  $\Delta S$ , as the nearest neighbors in  $S - \Delta S$  of the points in  $(T - (S - \Delta S))$  where computed in the previous iterations, and are already stored in  $nearest$ .

After having computed the closest point  $nearest[q]$  in  $\Delta S$ , and hence in  $S$ , of a point  $q$  in  $(T - S)$ , the array  $rep$  can be updated efficiently as follows. If the label of  $q$  is different from the label of  $nearest[q]$ , then  $q$  is misclassified. In this case, if the distance from  $nearest[q]$  to  $q$  is less than the distance from  $nearest[q]$  to its current associated representative  $rep[nearest[q]]$ , then  $rep[nearest[q]]$  is set to  $q$ .

The following theorem states an upper bound to the complexity of the FCNN rule.

**Theorem 3.3:** The FCNN1 rule requires at most  $|T| \cdot |S|$  distance computations using  $O(|T|)$  space.

*Proof:* The algorithm shown in Figure 5 has space complexity  $O(|T|)$ , as it employs the two arrays  $nearest$  and  $rep$ , having size  $|T|$  and  $|S|$  respectively, with  $|S| \leq |T|$ .

Let  $S_{i-1}$  and  $\Delta S_{i-1}$  denote, respectively, the values of sets  $S$  and  $\Delta S$  at the beginning of the  $i$ th iteration of the algorithm of Figure 5, and let  $S_i$  denote set  $S_{i-1} \cup \Delta S_{i-1}$ . In order to find the nearest neighbor among the points in  $\Delta S_{i-1}$  of each training set point in  $(T - S_i)$ , the algorithm computes  $(|T| - |S_{i-1}|) \Delta S_{i-1}$  distances.

Note that, in order to avoid repeated distance calculations, the distances  $\{d(q, nearest[q]) \mid q \in T\}$  and  $\{d(p, rep[p]) \mid p \in S\}$  can be maintained into two additional arrays of floating point numbers having size

$|T|$  and  $|S|$ , respectively, without additional asymptotic space requirements.

Thus, the overall number of distances computed is  $|T| \cdot \sum_i |\Delta S_i| - \sum_i |S_i| \cdot |\Delta S_{i-1}| = |T| \cdot |S| - \sum_{i < j} |\Delta S_i| \cdot |\Delta S_j|$ , and, hence,  $|T| \cdot |S|$  is an upper bound to the number of distance computations required.  $\square$

The FCNN2 has a very similar implementation. The only difference lies in the update of the array  $rep$ . Indeed, in order to determine the centroids of the misclassified points of each Voronoi cell induced by the points in  $S$ , the FCNN2 rule performs a supplemental training set scan at the end of each main iteration. Hence, as for the worst case time complexity of the FCNN2 rule, it requires at most  $\sum_i [(|T| - |S_i|) |\Delta S_{i-1}| + (|T| - |S_i|)] \leq |T| \cdot (|S| + t) \leq 2|T| \cdot |S|$  distance computations, where  $t \leq |S|$  is the number of iterations required to converge. As we will see in the experimental results section, the number  $t$  of iterations performed by the FCNN2 rule is always small (up to 20-30 iterations, even when  $S$  is composed of ten of thousands objects).

### 3.2 Exploiting the Triangle Inequality

In a metric space the FCNN rule can take advantage of the triangle inequality in order to avoid the comparison of each point in  $(T - S)$  with each point in  $\Delta S$ .

To this aim, the grouping in Voronoi cells of the training set points is exploited. During the generic  $i$ th main iteration, for each  $p \in S_{i-1}$ , the Voronoi cell  $\text{Vor}(p, S_{i-1}, T)$  is visited, and the points  $q \in (\text{Vor}(p, S_{i-1}, T) - \{p\})$  are compared with the points in  $\Delta S_{i-1}$ . Before starting the comparison, the distances among the point  $p$  and the points in  $\Delta S_{i-1}$  are computed, and the points in  $\Delta S_{i-1}$  are sorted in order of increasing distance from  $p$ . Then, instead of comparing each point  $q$  in  $\text{Vor}(p, S_{i-1}, T)$  with each point in  $\Delta S_{i-1}$ , each  $q$  is compared with the points in  $\Delta S_{i-1}$  having a distance from  $p$  less than twice the distance from  $q$  and  $p$ . Indeed, by the triangle inequality, they are all and the only points of  $\Delta S_{i-1}$  candidate to be closer to  $q$  than  $p$ .

We note that the partitioning of points in the Voronoi cells is induced by the array  $nearest$ . In order to retrieve the points lying in the same Voronoi cell efficiently, the array  $nearest$  does not directly store the identifiers of training set nearest neighbors. Rather, it is used to implement lists of points lying in the same Voronoi cell. To this aim, each entry  $nearest[q]$  stores the identifier of the successor of  $q$  in the list associated with the Voronoi cell containing  $q$ . The identifier of the first point of each list is then stored in an additional array having size  $|S_i|$ .

This implementation of the FCNN rule requires  $\sum_{i < j} |\Delta S_i| \cdot |\Delta S_j|$  additional distance computations and has no additional asymptotic space requirements. Thus, the upper bound of Theorem 3.3 remains unchanged. Furthermore,  $\sum_{i < j} |\Delta S_i| \cdot |\Delta S_j| \cdot \log |\Delta S_j|$  worst case comparisons of floating point numbers to sort distances are additionally required, but it must be noticed that the former operation is more expensive (its cost being related

to the dimensionality of the feature space) than the latter (which has instead a constant cost). Nevertheless, the last implementation guarantees great savings in terms of computed distances, and definitively outperforms the previous one, as will be confirmed by the experimental results.

### 3.3 Extension to $k$ nearest neighbors

The nearest neighbor decision rule can be generalized to the case in which the  $k$  nearest neighbors are taken into account. In such a case, a new object is assigned to the class with the most members present among the  $k$  nearest neighbors of the object in the training set. This rule has the additional property that it provides a good estimate of the Bayes error and that its probability of error asymptotically approaches the Bayes error [18], [17]. Let  $T$  be a training set and let  $p$  be an element of  $T$ . We denote by  $nn_k(p, T)$  the  $k$ th nearest neighbor of  $p$  in  $T$ , and by  $nns_k(p, T)$  the set  $\{nn_i(p, T) \mid 1 \leq i \leq k\}$ . Given a point  $q$ , the  $k$ -NN rule  $NN_k(q, T)$  assigns to  $q$  the label of the class with the most members present in  $nns_k(p, T)$ .

The FCNN algorithm can be directly adapted to deal with  $k$  nearest neighbors, by using the following notion of consistency: a subset  $S$  of  $T$  is said to be a  $k$ -training set consistent subset of  $T$  if, for each  $p \in (T - S)$ ,  $l(p) = NN_k(p, S)$ . Thus, this definition guarantees that the objects in  $T - S$  are correctly classified by  $S$  through the  $k$  nearest neighbor rule. For  $k = 1$ , the above definition coincides with the definition provided in Section 3.

The FCNN rule can directly compute a  $k$ -training set consistent subset  $S$  of  $T$  by using the following definition of  $Voren(p, S, T)$ . Let  $p$  be an element of  $S$ . We denote by  $Voren_k(p, S, T)$  the set  $\{q \in (Vor(p, S, T) - \{p\}) \mid l(q) \neq NN_k(q, S)\}$ . Thus, the set  $Voren_k(p, S, T)$  is composed of the points lying in the Voronoi cell of  $p$  ( $p$  excluded) which are misclassified by  $S$  through the  $k$  nearest neighbor rule. The implementation of the variant taking into account  $k$  nearest neighbors is the same as shown in Figure 5 with two differences: each entry  $nearest[q]$  of the array  $nearest$  stores the  $k$  objects of  $S$  closest to  $q$ , while  $S$  assigns  $q$  to the most frequent class label associated with the points in  $nearest[q]$ . Thus, Theorems 3.2 and 3.3 remain valid, but the space required is now  $O(k|T|)$ .

In Section 4.3, the behavior of this variant will be studied on some real world domains.

## 4 EXPERIMENTAL RESULTS

In this section we present experimental results obtained by using the FCNN rule.

Next, the competence preservation algorithms compared are commented upon. As far as the FCNN rule is concerned, the algorithm described in Section 3 was used. As for the other methods, first of all it must be noticed that the NNSRM rule is impracticable on large

training sets since its first step consists in computing all the pairwise distances among training set objects. Hence, we considered it only in experiments involving small-sized data sets. Furthermore, we note that the MCNN and CNN rules are slow if compared with the FCNN. Thus, in order to improve the efficiency of these methods, we enhanced them by adopting the two following strategies. (1) *Avoiding repeated distance computations* by maintaining the nearest neighbor of each training set point computed so far in an array of size  $|T|$  and then comparing each point only with the incremental part of the training set consistent subset (whose definition varies depending on the method considered) during a generic iteration. (2) *Exploiting the triangle inequality* technique described in Section 3.2 with the MCNN rule. Indeed, when set  $S_m$  becomes empty, the set  $C$  of the centroids computed by the MCNN method plays the same role of set  $\Delta S$  in the FCNN rule. Since these implementations represent a major modification of the basic methods, in the following the variant of the CNN rule augmented with strategy (1) is called fCNN, and the variant of the MCNN rule employing both strategies (1) and (2) is called fMCNN. It is worth pointing out that, with respect to the subset computed and the number of iterations performed, the fCNN rule is equivalent to the CNN rule and the fMCNN is equivalent to the MCNN.

The experiments are organized as follows.

Section 4.1 describes the experiments executed on three large training sets in order to study the scaling and learning behavior of the FCNN rule.

Next, Section 4.2 compares the competence preservation methods on small-sized real training sets. Both mutual condensation ratios and classification accuracies are measured and compared.

Finally, Section 4.3 experiments the FCNN method on two pattern recognition domains, studies the behavior of the variant of the basic rule taking into account  $k$  neighbors, and compares the FCNN rule with hybrid instance-based learning algorithms.

The results of the experiments presented below in this section are then discussed in the subsequent Section 5.

In all the experiments we used the Euclidean distance as a distance metrics. All the experiments were executed on a Pentium IV 2.66GHz based machine with 1GB of main memory under the Windows operating system.

### 4.1 Large data set

We tested the training set consistent subset methods on the three following large data sets.

The *Checkerboard* data set is a synthetically generated  $4 \times 4$  checkerboard data set partitioning points of the unit square into two classes composed of 1,000,000 points.

The *Forest Cover Type* data set<sup>5</sup> contains forest cover type data from the US Forest Service. It is composed of 581,012 tuples associated with  $30 \times 30$  meter cells. Each tuple has 54 attributes, of which 10 are quantitative (e.g.

5. <http://kdd.ics.uci.edu/databases/covertype/covertype.html>

	Method	Time	Subset size	Accuracy	Iterations	Complexity	Speed-up
<i>Checkerboard</i>	FCNN1	149	5,535 (0.55%)	99.77	149	1.44	11.2
	FCNN2	70	7,112 (0.71%)	99.75	25	1.45	44.4
	FCNN3	1,452	5,510 (0.55%)	99.77	5,510	1.37	10.4
	FCNN4	657	6,856 (0.69%)	99.76	6,856	1.37	85.6
	fMCNN	2,877	7,387 (0.74%)	99.74	3,975	1.60	–
	fCNN	2,748	7,866 (0.79%)	99.75	5	1.64	–
	TRAIN	–	–	99.80	–	–	–
<i>Forest Cover Type</i>	FCNN1	3,750	58,385 (10.05%)	85.64	59	1.61	17.0
	FCNN2	2,566	60,465 (10.41%)	84.99	27	1.59	21.8
	FCNN3	2,681	54,294 (9.35%)	84.78	54,294	1.59	21.1
	FCNN4	2,521	56,802 (9.78%)	84.60	56,802	1.59	40.8
	fMCNN	68,397	64,660 (11.13%)	85.39	29,234	1.84	–
	fCNN	35,019	64,741 (11.14%)	85.58	8	1.83	–
	TRAIN	–	–	87.51	–	–	–
<i>DARPA 1998</i>	FCNN1	33	3,365 (0.73%)	99.46	281	1.37	26.0
	FCNN2	23	4,064 (0.89%)	99.45	23	1.31	68.0
	FCNN3	308	3,205 (0.70%)	99.46	3,205	1.41	16.3
	FCNN4	232	3,946 (0.86%)	99.44	3,946	1.35	79.7
	fMCNN	858	4,730 (1.03%)	99.45	2,840	1.56	–
	fCNN	1,054	4,497 (0.98%)	99.38	13	1.65	–
	TRAIN	–	–	99.56	–	–	–

TABLE 1  
Experiments on large data sets.

elevation, aspect, slope, etc.), 4 are binary wilderness areas, and 40 are binary soil type variables. The data is partitioned into 7 classes.

The Defense Advanced Research Projects Agency 1998 intrusion detection evaluation data set<sup>6</sup> consists of network intrusions simulated in a military network environment. The TCP connections have been elaborated to construct a data set of 23 numerical features, one of which identifying the kind of attack: *DoS*, *R2L*, *U2R*, and *PROBING*. We used the TCP connections from 5 weeks of training data. The training set is composed of 458,301 objects partitioned into five classes (one representing normal data and the other associated with the different types of attack).

Table 1 summarizes the results of the experiments on the whole training sets. For each experiment are reported the execution time in seconds, the number of objects composing the consistent subset (between parentheses there is the relative size of the subset), the test accuracy measured using the hold out method, the number of iterations performed, the empirical complexity (defined next), and the speed-up achieved by exploiting the triangle inequality (see Section 3.2).

The *empirical complexity* is the ratio  $\log D / \log |T|$ , where  $D$  denotes the number of distances computed by the method, and provides an estimate of its computational complexity. Although it provides a short summary, it is not sufficient to characterize the effective effort of the method, since the execution time is influenced also by the number of iterations and by the number of training set passes per iteration.

The *speed-up* (defined only for the FCNN methods) is

defined as the ratio between the worst case number of distances computed by the method (see Theorem 3.3 and the related part in the text) and the number of distances actually computed by the method exploiting the triangle inequality. It was measured to verify effectiveness of the triangle inequality technique described in Section 3.2.

Scaling analysis of the execution time and of the size of the consistent subset computed is reported in Figure 6.

#### 4.1.1 Execution time

On the *Checkerboard* data set, the methods can be partitioned into three groups: the FCNN1 and FCNN2 are very fast, the FCNN3 and the FCNN4 have higher time requirements than the previous two rules, while the fMCNN and the fCNN are slow. It is worth noting that the fMCNN and fCNN rules are 40 times slower than the FCNN2 rule. We also measured the execution time of the MCNN and CNN rules, and, unfortunately, they turned out to be very slow. Indeed, the MCNN required 131,402 seconds to process 200,000 points, while the CNN required 4,032 seconds to process 400,000 points. Since these rules are too time demanding, we decided to disregard their analysis on larger samples. Moreover, they are no longer considered in subsequent experiments.

On the *Forest Cover Type* data set, the FCNN rules clearly outperformed both the fCNN and the fMCNN rules. In particular, on the whole training set, the FCNN2, FCNN3, and FCNN4 rules required about 2,500 seconds, the FCNN1 performed slightly worse, while the fCNN rule was more than 14 times slower than the FCNN rules and the fMCNN was about 27 times slower.

6. <http://www.ll.mit.edu/IST/ideval/index.html>

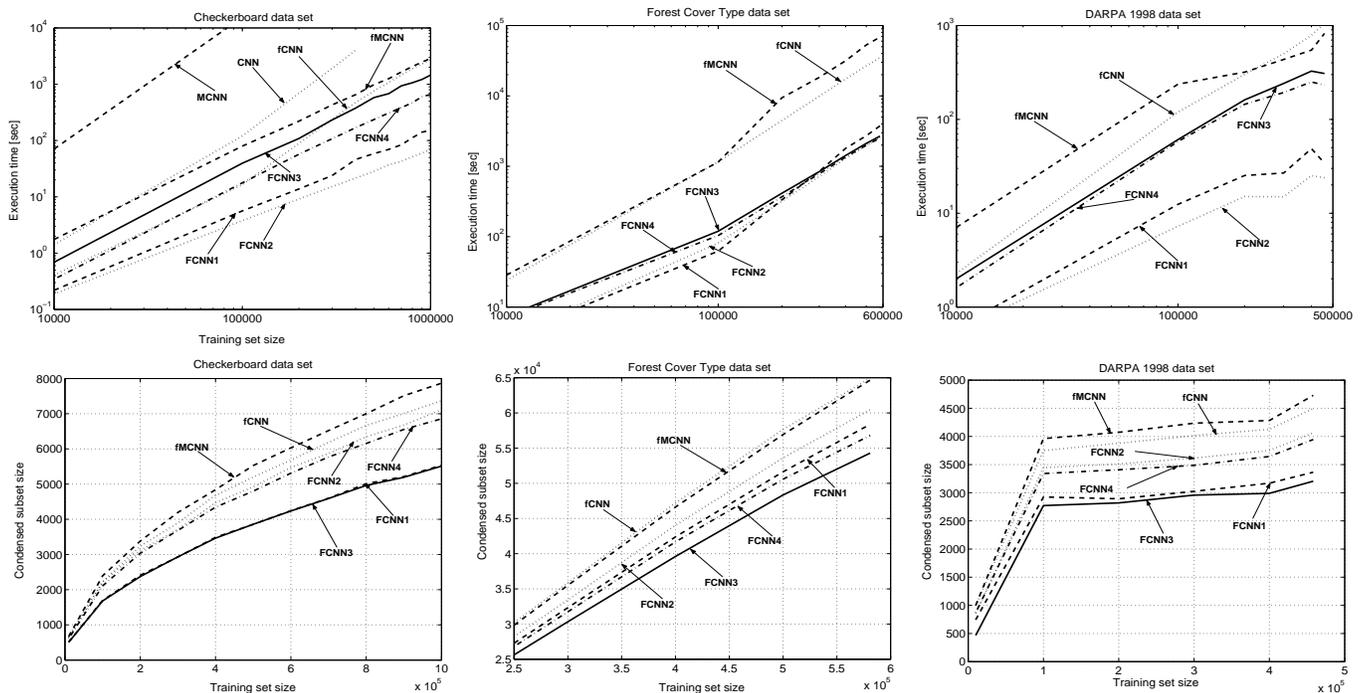


Fig. 6. Large data sets: scaling analysis.

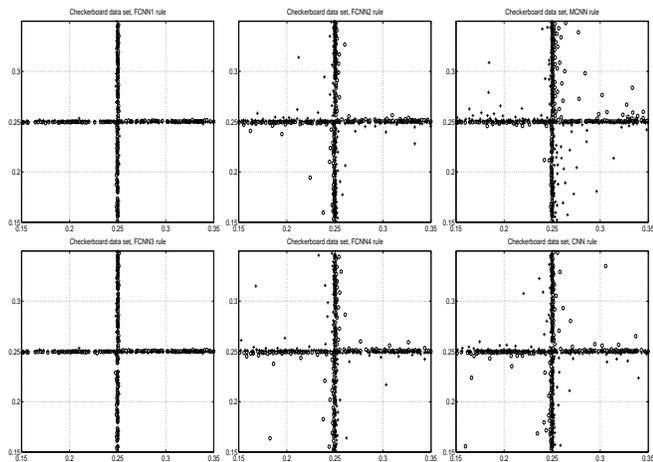


Fig. 7. Checkerboard data set: training set consistent subset computed.

On the *DARPA 1998* data set, the fastest rules were the FCNN2 and the FCNN1, followed by the FCNN4 and FCNN3, and eventually by the fMCNN and fCNN. The first method was about 45 times faster than the latter.

#### 4.1.2 Subset size

On the *Checkerboard* data set, the FCNN1 and FCNN3 computed the smallest subsets, while the FCNN2, FCNN4, CNN, and MCNN computed greater subsets, even though the MCNN computed a subset noticeably greater than those computed by any other method.

Figure 7 shows the subset computed by the various methods on the whole data set (for clarity, only the square  $[0.15, 0.35] \times [0.15, 0.35]$  is reported). Circles and

crosses are the consistent subset points of the two classes. It can be noticed that the FCNN1 and FCNN3 rules select points very close to the boundaries between the checkerboard cells, while the other methods also contain points inside the cells, though the MCNN and CNN appears to have high percentage of such points.

On *Forest Cover Type*, the FCNN3 computed the smallest subset, followed by the FCNN4, the FCNN1, the FCNN2, the MCNN, and the CNN. Notice that the subset computed by the FCNN3 rule contains 10,000 points less than the last two subsets.

As for *DARPA 1998*, three groups can be identified, i.e. the FCNN3 together with FCNN1, which reported the smallest subsets, the FCNN4 together with the FCNN2 and, finally, the CNN together with the MCNN. It is worth noting that the largest subset is about 1.5 times greater than those returned by the FCNN3 rule.

#### 4.1.3 Iterations

As far as the number of iterations is concerned (see Table 1), it can be noticed that the behavior of the methods was the same in all the experiments. Indeed, the FCNN3 and FCNN4 rules in general execute a lot of iterations. In fact, since these rules add one point per iteration, the number of performed iterations is identical to the size of the training set consistent subset. Also, the MCNN rule adds at most  $m$  points per iteration, where  $m$  is the number of classes in  $T$ , and thus the number of iterations it requires is greater than the number of subset points divided by the number of classes. The CNN rule always performs few iterations (less than ten) since it selects almost all the points of the subset during the first two iterations, and some additional iterations

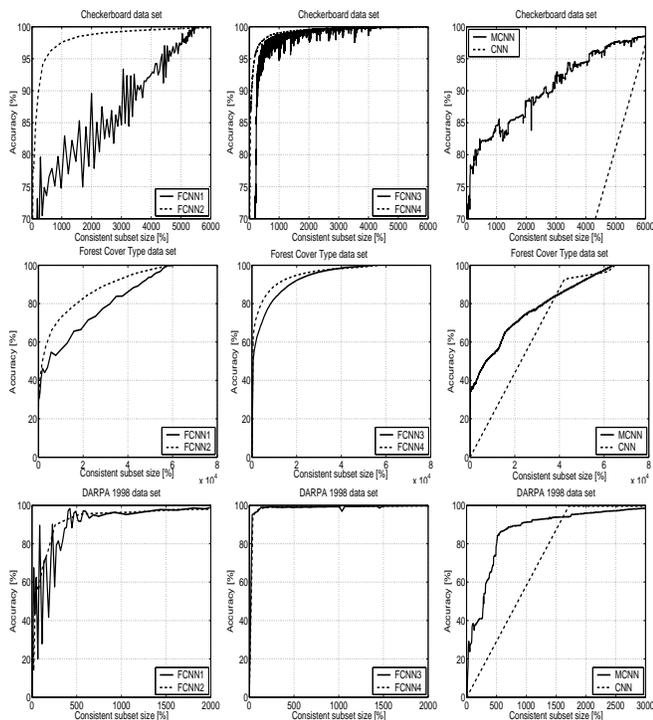


Fig. 8. Large data sets: training set classification accuracy versus subset size.

are then needed to assure consistency. However, this strategy has the drawback of producing a consistent subset noticeably larger than that returned by competitor methods. The FCNN2 rule always performs about a few tens of iterations, since it starts from the centroids of the classes and, due to the strategy adopted to select representatives of the misclassified points, rapidly covers the regions of the feature space far from the centroids. Finally, the FCNN1 rule is sensitive to the complexity of the decision boundary. The more involved this boundary is, the more iterations are required by the rule. Indeed, an involved decision boundary can make it difficult to cover the feature space quickly, due to the strategy of selecting nearest Voronoi enemies as representatives of misclassified points. Summarizing, in general the FCNN1 rule performs more iterations than the FCNN2 rule. Depending on the complexity of the decision boundary, it might execute from approximately the same number of iterations of the FCNN2 rule to some hundreds of iterations.

#### 4.1.4 Training-set accuracy

Figure 8 shows the accuracy of the subset  $S_i$ , computed during the  $i$ th iteration of the algorithms, in classifying the overall training set  $T$ .

As for *Checkerboard*, for clarity, only values of accuracy between 70% and 100% and subsets containing less than 6,000 points are shown. On this training set the curve of the FCNN1 rule oscillates sensibly and converges quite slowly. Differently, curves of the FCNN2, FCNN3, and FCNN4 methods rapidly converge to high values of

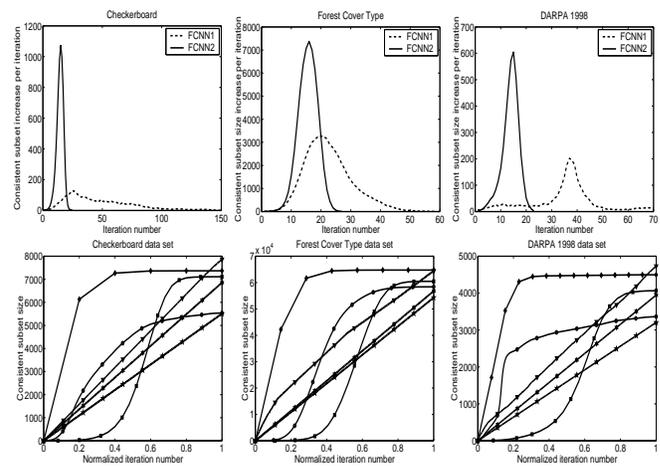


Fig. 9. Large data sets: points added per iteration versus iteration number (on the bottom: circle=FCNN1, square=FCNN2, pentagram=FCNN3, hexagram=FCNN4, triangle=MCNN, and diamond=CNN).

accuracy. The curve of the FCNN3 method also oscillates, but its fluctuation is more contained than that of the FCNN1 rule. It can be noticed that the area under the curve of the FCNN4 rule is the largest among the areas of all the methods (the first 178 points achieved an accuracy 95.09%, 968 points an accuracy 99.00%, and 5,164 points an accuracy 99.90%), while the FCNN2 achieved the second largest area (366 points achieved accuracy 94.38%, 2,377 points accuracy 99.04%, and 6,727 points accuracy 99.96%). The curve of the MCNN rule is similar to that of the FCNN1 rule, but it is less wavering. Finally, the curve of the CNN rule is the least accurate.

The above behavior is confirmed by the *Forest Cover Type* data set, where the curve of the FCNN4 method is the more accurate (the first 11,023 points achieved an accuracy of 90%, 20,738 points an accuracy of 95%, and 48,009 points an accuracy of 99%), and by the *DARPA 1998* data set, where the FCNN3 and FCNN4 showed the largest area (the curves of these two methods were smoothed for readability).

#### 4.1.5 Course of the subset size

Figure 9 on the top shows the number  $|\Delta S_i|$  of points added versus the iteration number  $i$  for the FCNN1 and FCNN2 methods. The curves show that during the initial and final iterations the incremental sets include a relatively small number of points, while during the central iterations the size of the incremental sets reaches a peak. Interestingly, the curve of the FCNN2 rule follows a gaussian-like distribution, while the curve of the FCNN1 is also unimodal but less symmetrical. This behavior is observed in all the experiments.

On the bottom it is shown the size  $|S_i|$  of the subset  $S_i$  versus the normalized iteration number  $\frac{i}{i_{\max}}$ . The curves of the FCNN3, FCNN4, and MCNN rules are straight lines, since they add almost a constant number of points per iteration (one point in the case of the FCNN3 and

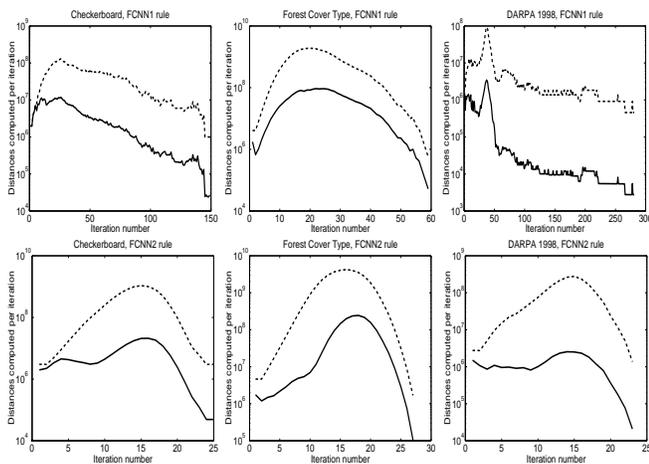


Fig. 10. **Large data sets:** distances computed versus iteration number (solid lines=distances computed by exploiting the triangle inequality, dashed lines=worst case number of distances to be calculated by the triangle inequality based method).

FCNN4). The curve of the CNN is a step, as almost all the points are selected during the first and the second iterations. Finally, the curves of the FCNN1 and FCNN2 rules resemble a sigmoid as expected from the previous diagrams.

#### 4.1.6 Distance computation savings

Figure 10 shows the distances computed by the FCNN1 and FCNN2 methods versus the iteration number. The solid lines represent the number of distances actually calculated by the methods exploiting the triangle inequality, while the dashed lines represent the worst case number of distances to be calculated by the same algorithms. It is clear from these figures that great savings are obtained, since the area between the solid and the dashed curve is at least one order of magnitude greater than the area under the solid curve. Similar curves were obtained also for the FCNN3 and FCNN4 rules. Table 1 summarizes the speed-up obtained by the FCNN methods through the use of the triangle inequality.

#### 4.1.7 Test accuracy

We discussed in Section 1 techniques for improving accuracy of nearest neighbor classifiers. Here we are mainly interested in comparing performances of different competence preservation methods on the same data, rather than in improving accuracy of the classifier. The test accuracy measured through the hold out method is summarized in Table 1. Accuracy of the methods was always almost the same and, in two cases, identical to the accuracy of the whole training set, while on the *Forest Cover Type* data set, all the methods reported a sensible loss in accuracy w.r.t. the training set.

## 4.2 Small data sets

In these experiments, a number of small training sets are considered in order to compare the CNN, MCNN, and

Data set name	Abbreviation	Size	Features	Classes
Bupa	BUP	345	6	2
Coil 2000	COI	4,992	85	2
Colon Tumor	COL	62	2,000	2
Echocardiogram	ECH	61	11	2
Ionosphere	ION	351	34	2
Iris	IRI	150	4	3
Image Segmentation	IMA	2,310	19	7
Pen Digits	PEN	7,494	16	10
Pima Indians Diabetes	PIM	768	8	2
Satellite Image	SAT	6,435	36	6
Spam Database	SPA	4,207	57	2
SPECT Heart Data	SPE	349	44	2
Vehicle	VEH	846	18	4
Wisconsin Breast Cancer	WBC	683	9	2
Wine	WIN	178	13	3
Wisc. Prognostic Breast Cancer	WPB	198	33	2

TABLE 2

UCI ML Repository data sets used in the experiments.

NNSRM rules with the FCNN rule.

The training sets employed are reported in Table 2. The data sets are from the UCI Machine Learning Repository<sup>7</sup>. Ten fold cross validation has been accomplished for each training set. Therefore, measures reported in the following concerning subset sizes, classification accuracies, and execution times, are average values over the ten executions. The NNSRM rule was tested only on training sets having two class labels, since the implementation we had available supports only this kind of data sets.

We start commenting on results shown in Table 3. This table reports the size of the training set consistent subsets (first line) together with the percentage of objects included in each subset (second line). At the bottom of the table, the achieved compression ratios are compared.

Since the size of the smallest training set consistent subset is an intrinsic property of the training set, to compare the various methods we measured the ratios  $\frac{size_{m_1} \%}{size_{m_2} \%}$  where  $m_1$  and  $m_2$  denote two methods, while  $size_{m_i} \%$  denotes the percentage of training set objects composing the subset computed by the method  $m_i$ . In particular, the entry of row  $m_r$  and column  $m_c$  at the bottom of the table represents the geometric mean of the ratios  $\frac{size_{m_r} \%}{size_{m_c} \%}$  achieved in the experiments reported at the top of the table.

As for the size of the subset, on these training sets the FCNN3 and FCNN4 rules performed better than any other method. As for the FCNN1 and the FCNN2 methods, they computed a subset which is on average, respectively, four and the five percent larger than that computed by the FCNN4 method. This difference can be explained by noticing that the FCNN3 and FCNN4 rules are more accurate than the FCNN1 and FCNN2 rules in the choice of the prototypes to add to the current subset. The MCNN method performed even well: indeed the increase in size w.r.t. the FCNN3 method is, on average, one percent. The increase in size w.r.t. the FCNN3 and FCNN4 of the CNN rule is noticeable, since it amounts to sixteen percent. Finally, it can be observed that the

7. <http://mllearn.ics.uci.edu/MLRepository.html>

	FCNN1		FCNN2		FCNN3		FCNN4		MCNN		CNN		NNSRM	
BUP	175	(56.45)	176	(56.77)	<b>168</b>	(54.19)	173	(55.81)	173	(55.81)	184	(59.26)	307	(99.03)
COI	904	(20.41)	907	(20.48)	<b>902</b>	(20.37)	913	(20.61)	909	(20.52)	1117	(24.86)	4372	(98.71)
COL	19	(34.55)	19	(34.55)	18	(32.73)	<b>17</b>	(30.91)	19	(34.55)	21	(37.63)	49	(89.09)
ECH	6	(11.11)	6	(11.11)	<b>5</b>	(9.26)	6	(11.11)	6	(11.11)	8	(14.57)	15	(27.78)
ION	60	(19.05)	62	(19.68)	59	(18.73)	58	(18.41)	<b>55</b>	(17.46)	73	(23.11)	309	(98.10)
IRI	15	(11.11)	14	(10.37)	16	(11.85)	14	(10.37)	<b>13</b>	(9.63)	16	(11.85)	—	—
IMA	274	(13.18)	272	(13.08)	<b>245</b>	(11.78)	252	(12.12)	256	(12.31)	272	(13.08)	—	—
PEN	300	(4.45)	284	(4.21)	286	(4.24)	<b>249</b>	(3.69)	253	(3.75)	311	(4.61)	—	—
PIM	329	(47.61)	333	(48.19)	<b>316</b>	(45.73)	332	(48.05)	334	(48.34)	357	(51.65)	691	(100.00)
SAT	1051	(18.15)	1055	(18.22)	1007	(17.39)	<b>983</b>	(16.97)	1012	(17.48)	1128	(19.48)	—	—
SPA	849	(22.42)	869	(22.95)	819	(21.63)	805	(21.26)	<b>800</b>	(21.13)	961	(25.38)	3774	(99.68)
SPE	98	(31.21)	104	(33.12)	<b>93</b>	(29.62)	96	(30.57)	94	(29.94)	107	(34.07)	257	(81.85)
VEH	401	(52.69)	398	(52.30)	385	(50.59)	<b>382</b>	(50.20)	390	(51.25)	405	(53.19)	—	—
WBC	55	(8.96)	55	(8.96)	<b>54</b>	(8.79)	54	(8.79)	<b>54</b>	(8.79)	64	(10.41)	511	(83.22)
WIN	63	(39.38)	64	(40.00)	<b>62</b>	(38.75)	63	(39.38)	64	(40.00)	65	(40.57)	—	—
WFB	84	(47.19)	85	(47.75)	<b>79</b>	(44.38)	83	(46.63)	84	(47.19)	90	(50.51)	175	(98.31)
FCNN1	—	—	1.00	—	1.04	—	1.04	—	1.04	—	0.90	—	0.32	—
FCNN2	1.00	—	—	—	1.05	—	1.05	—	1.04	—	0.91	—	0.33	—
FCNN3	0.96	0.95	—	—	—	—	1.00	—	0.99	—	0.87	—	0.31	—
FCNN4	0.96	0.96	—	—	1.00	—	—	—	0.99	—	0.87	—	0.32	—
MCNN	0.97	0.96	—	—	1.01	—	—	—	—	—	0.87	—	0.32	—
CNN	1.11	1.10	—	—	1.16	—	1.15	—	1.15	—	—	—	0.37	—
NNSRM	3.09	3.04	—	—	3.24	—	3.16	—	3.13	—	2.69	—	—	—

TABLE 3

**Small data sets:** training set consistent subset size and percentage of training set instances composing the training set consistent subset (between parentheses).

NNSRM rule contains a huge percentage of the training set objects. Thus, its increase in size w.r.t. the other methods is very high, namely, more than 200% in almost all the experiments.

Table 4 shows the classification accuracy measured using ten-fold cross-validation. The last column is the classification achieved when all the training set objects are used as a reference set during classification. At the bottom, the mean over all the experiments (mean 1) and the mean over the experiments concerning the NNSRM rule (mean 2) are reported. We can observe that all the methods present a loss of classification accuracy w.r.t. the use of the overall training set. This loss ranges from  $-1.5\%$  of the FCNN2 rule to  $-2.0\%$  of the CNN rule. Furthermore, we note that the NNSRM rule exhibits a negligible loss in accuracy. This can be explained by noticing that the subset computed by this rule contains a high percentage of training set objects, almost all the object in most cases, thus the small loss in accuracy is not repaid by a significative reduction of the reference set.

Finally, Table 5 summarizes the execution times. In almost all cases the times amount to a small fraction of one second or to a few seconds. Thus, we reported only

	FCNN1	FCNN2	FCNN3	FCNN4	MCNN	CNN	NNSRM
FCNN1	—	1.07	0.63	0.79	0.03	0.39	0.30
FCNN2	0.93	—	0.56	0.70	0.02	0.31	0.24
FCNN3	1.59	1.80	—	1.25	0.05	0.61	0.42
FCNN4	1.27	1.44	0.80	—	0.04	0.49	0.36
MCNN	33.17	44.78	20.90	26.19	—	12.83	8.16
CNN	2.59	3.24	1.63	2.04	0.08	—	0.62
NNSRM	3.28	4.22	2.37	2.76	0.12	1.62	—

TABLE 5

**Small data sets:** relative execution time.

the relative speeds: the entry of row  $m_r$  and column  $m_c$  of the table represents the geometric mean of the ratios  $\frac{time_{m_r}}{time_{m_c}}$ , where  $time_{m_i}$  denotes the execution time of the method  $m_i$ . It can be observed that, on these data sets, the FCNN2 is the fastest method, followed by the FCNN1 method, while the CNN, NNSRM, and MCNN are the slowest methods. In particular, the CNN is two and half times slower than the FCNN2, the NNSRM is three times slower, and the MCNN is thirty times slower. It should be noticed that the last result is due to the presence of some outlying data sets on which the MCNN rule performs badly (i.e. COI, SAT, and SPA).

### 4.3 Comparison with hybrid methods

In this section, some experiments on the *MNIST* and *MIT Face* databases are described. These two real-world databases are considered good test-beds for learning techniques and pattern recognition methods. Thus, the goal of this section is to test applicability of the FCNN rule on some typical pattern recognition domains, and to compare the FCNN methods with *hybrid* instance-based learning methods on some real-world applications.

The *MNIST* database<sup>8</sup> of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a  $28 \times 28$  image. Each example is composed of 784 features, each associated with a distinct pixel of the image (pixel values range from 0 to 255). Examples are partitioned in ten, about equally sized, classes. The *MIT Face* database<sup>9</sup> is a database of faces and non-faces, that has been used extensively at the Center for Biological and Computational Learning at MIT. This database has

8. <http://yann.lecun.com/exdb/mnist/>

9. <http://www.ai.mit.edu/projects/cbcl>

	FCNN1	FCNN2	FCNN3	FCNN4	MCNN	CNN	NNSRM	TRAIN
BUP	58.53 ± 0.88	59.71 ± 0.80	59.41 ± 0.79	58.23 ± 0.81	56.18 ± 0.84	56.47 ± 0.71	58.23 ± 0.57	57.94 ± 0.59
COI	85.73 ± 0.20	85.45 ± 0.20	85.55 ± 0.16	85.85 ± 0.15	85.95 ± 0.40	86.61 ± 0.16	89.47 ± 0.16	89.39 ± 0.16
COL	75.00 ± 1.71	78.33 ± 1.50	76.67 ± 2.00	78.33 ± 1.68	76.67 ± 1.34	80.00 ± 1.79	78.33 ± 0.76	80.00 ± 1.00
ECH	95.00 ± 0.76	93.33 ± 0.82	95.00 ± 0.76	93.33 ± 0.82	91.67 ± 1.05	90.00 ± 1.11	96.67 ± 0.67	95.00 ± 0.76
ION	85.43 ± 0.69	86.00 ± 0.61	86.00 ± 0.61	86.29 ± 0.55	86.29 ± 0.75	86.00 ± 0.56	86.86 ± 0.69	86.86 ± 0.69
IRI	93.33 ± 0.52	92.00 ± 0.50	93.33 ± 0.52	92.67 ± 0.55	92.67 ± 0.72	95.33 ± 0.52	—	95.33 ± 0.43
IMA	94.81 ± 0.18	95.71 ± 0.22	95.02 ± 0.22	94.63 ± 0.24	95.33 ± 0.44	94.63 ± 0.19	—	96.36 ± 0.16
PEN	98.69 ± 0.03	98.75 ± 0.03	98.60 ± 0.03	98.44 ± 0.04	98.64 ± 0.18	98.60 ± 0.03	—	99.44 ± 0.03
PIM	67.11 ± 0.29	67.63 ± 0.42	66.45 ± 0.22	66.71 ± 0.34	65.92 ± 0.61	65.39 ± 0.37	69.21 ± 0.47	69.21 ± 0.47
SAT	88.48 ± 0.15	88.44 ± 0.14	88.20 ± 0.12	88.68 ± 0.14	88.90 ± 0.40	88.69 ± 0.16	—	90.64 ± 0.13
SPA	86.05 ± 0.22	86.17 ± 0.20	85.88 ± 0.29	86.17 ± 0.23	85.57 ± 0.45	86.45 ± 0.20	89.79 ± 0.14	89.79 ± 0.14
SPE	83.53 ± 0.61	85.00 ± 0.55	84.41 ± 0.67	82.94 ± 0.72	82.35 ± 0.82	83.82 ± 0.67	86.47 ± 0.46	86.47 ± 0.46
VEH	62.38 ± 0.55	64.76 ± 0.57	63.33 ± 0.79	62.14 ± 0.61	62.02 ± 0.85	62.38 ± 0.72	—	65.36 ± 0.68
WBC	93.53 ± 0.21	93.68 ± 0.25	93.53 ± 0.26	93.68 ± 0.28	93.82 ± 0.36	93.82 ± 0.13	95.88 ± 0.18	96.03 ± 0.13
WIN	76.47 ± 0.79	75.88 ± 0.76	74.71 ± 0.91	77.06 ± 0.72	74.71 ± 0.94	74.71 ± 0.87	—	77.06 ± 0.85
WPB	68.95 ± 0.72	68.42 ± 1.15	68.42 ± 0.88	68.42 ± 1.13	67.37 ± 1.00	68.42 ± 1.00	70.53 ± 0.75	71.05 ± 0.79
mean 1	82.06	82.45	82.16	82.10	81.50	81.96	—	84.12
mean 2	79.89	80.37	80.13	80.00	79.18	79.70	82.14	82.17

TABLE 4  
Small data sets: classifier accuracy.

been partitioned in a training set of 25,317 face and 2,603 non-face examples, and in a test set of 2,804 face and 298 non-face examples. Each example (a  $19 \times 19$  image) consists of 361 features whose values range between 0 and 1.

In all the experiments the Euclidean distance was used.

Experimental results concerning the FCNN method are summarized in Table 6. As for the *MNIST* data set (see Table 6(a)), all the FCNN methods employed from fifteen to twenty minutes to compute a consistent subset including about 10% of the data set objects. The test accuracy of the FCNN rule is 94.5% (in [23] a  $k$ -nearest neighbor classifier using the whole training set, the parameter  $k$  set to 3, and the Euclidean distance, achieved 5.0% test error rate). On the *MIT Face* data set the FCNN1 and FCNN2 employed about 25 seconds, while the FCNN3 and FCNN4 about 95 seconds. The extracted subset is, in all cases, composed of 5.5% data set objects. The test accuracy is good, being close to 97%.

Figure 11 shows the results obtained by using the variant of the FCNN rule taking into account  $k$  nearest neighbors (see Section 3.3), for  $k$  ranging from 1 to 7. By using  $k = 3$ , the execution time (Figure 11 on the left) increased with respect to the case of  $k = 1$ . For greater values of  $k$ , the growth slowed down and the time remained substantially unchanged. As for the subset size (Figure 11 on the center), it remained almost the same for *MNIST*, while slightly decreased for *MIT Face*. In the latter case, the reduction achieved by the FCNN3 and FCNN4 methods is substantial. Finally, by increasing the number  $k$  of considered nearest neighbors, the test accuracy (Figure 11 on the right) improved. For example, the accuracy of the FCNN1 rule increased of 1.5% – 2.0% for  $k = 7$ .

The FCNN rule was compared with the DROP3 [32], ELGrow, and Explore [9] hybrid instance-based learning

algorithms<sup>10</sup>. The DROP3 was observed to have the best mix of storage reduction and generalization among the *Incremental Reduction Optimization Procedures* DROP1-DROP5, presented in [32]. The ELGrow and Explore methods are able to achieve high storage reduction, but are not as accurate as the DROP methods.

These methods were compared with the FCNN rules on the data sets *MNIST* and *MIT Face* (see Figure 12). The whole training set and some random samples having increasing size were considered to study the scaling behavior. If required by the method, the parameter  $k$  was set to 3 as done in [32].

Figure 12 shows on the left the execution time versus the training set size of the FCNN2, DROP3, ELGrow, and Explore algorithms on the two above described data sets. As already stated, the FCNN rule employed about one thousand seconds on the whole *MNIST* data set. On this data set, the other methods were very slow and it was decided to stop their execution. Thus, the largest sample considered for all methods included ten thousand examples. In this case, DROP3 required about 25,000 seconds, ELGrow and Explore about 7,000 seconds, and FCNN about 45 seconds. On the whole *MIT Face* training set, DROP3 employed about 75,000 seconds, ELGrow and Explore about 23,000 seconds, and FCNN about 25 seconds. It is clear that, as far as the learning speed is concerned, there is a difference of at least three orders of magnitude between the FCNN and the other competence enhancement methods on these medium-sized data sets. On larger data sets the hybrid methods are impractical.

Figure 12 shows on the center the relative subset size versus the training set sample size. The competence enhancement methods confirmed their expected behavior, since ELGrow and Explore achieved very high data reduction. The FCNN has the smallest reduction ratio but, except for very small sample sets, the size

<sup>10</sup> The implementations of these algorithms as available at the location <ftp://axon.cs.byu.edu/pub/randy/ml/drop> were used.

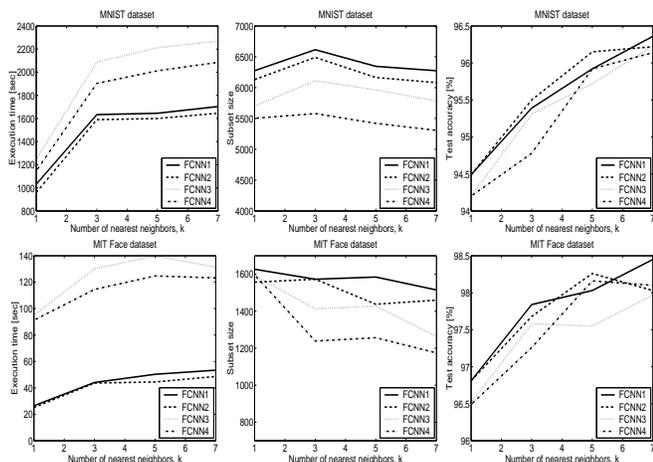


Fig. 11. The effect of considering  $k$  nearest neighbors on the *MNIST* and *MIT Face* data sets.

of its consistent subset is very close to that of DROP3, although slightly larger.

Finally, Figure 12 shows on the right the test accuracy achieved by using the subset computed by the various condensation methods. The ELGrow and Explore methods are the less accurate methods. As observed in [32], their aggressive storage reduction strategy may worsen their generalization capability. For example, on the *MIT Face* data set the subsets computed By ELGrow and Explore always misclassify the examples of the minority class. The loss in accuracy of ELGrow/Explore w.r.t. FCNN on the *MNIST* is about 10%. On the contrary, the DROP3 method confirms its good generalization capabilities due to a noise-filtering pass. It must be pointed out that FCNN performed remarkably well compared to instance-based competence enhancement methods. Indeed, other than performing better than ELGrow/Explore, it has the same accuracy of DROP3 on the *MNIST* data set, while the difference in accuracy between these two methods is about one percent on the *MIT Face* data set.

(a) <i>MNIST</i> data set				
	Execution time	Subset size		Test accuracy
FCNN1	1,031.5	6,275 (10.5%)		94.5
FCNN2	959.3	6,130 (10.2%)		94.5
FCNN3	1,244.6	5,704 (9.5%)		94.2
FCNN4	1,146.7	5,503 (9.2%)		94.2
(b) <i>MIT Face</i> data set				
	Execution time	Subset size		Test accuracy
FCNN1	26.5	1,627 (5.8%)		96.8
FCNN2	25.2	1,557 (5.5%)		96.8
FCNN3	95.0	1,593 (5.7%)		96.5
FCNN4	91.2	1,596 (5.7%)		96.5

TABLE 6

Execution time (seconds), subset size, and test accuracy of the FCNN rules on the *MNIST* and *MIT Face* data sets.

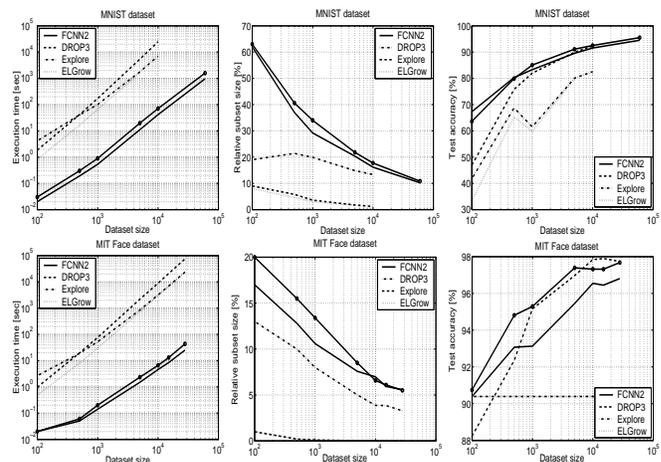


Fig. 12. Comparison with hybrid methods.

Since DROP3 incorporates a noise-filtering pass based on removing instances misclassified by its  $k$  nearest neighbors ( $k = 3$  in the experiments), Figure 12 reports also the behavior of the FCNN2 rule taking into account three nearest neighbors, as already observed, the FCNN slightly increases its execution time, with no appreciable difference on the size of the subset, and the accuracy improves and is identical to that of noise-filtering methods.

## 5 DISCUSSION AND CONCLUSIONS

This work introduces a novel algorithm, called the FCNN rule, for computing a training set consistent subset for the nearest neighbor rule.

The algorithm starts by selecting the centroid of each training set class and then, until consistency is achieved, selects for insertion a representative of the misclassified points of each Voronoi cell induced by the current subset. Four variants of the basic method are presented, called FCNN1-4 rules. In particular, the FCNN1 and FCNN2 rules augment the subset with all such representatives, while the FCNN3 and FCNN4 rules select only a representative per iteration. The FCNN1 (FCNN2 resp.) and the FCNN3 (FCNN4 resp.) rules are based on the same definition of representative.

Each of these rules has strengths and weaknesses that, generally speaking, can be summarized as follows.

The FCNN3 and FCNN4 are more careful in the choice of the points to select for insertion and, hence, the FCNN3 (FCNN4 resp.) returns a subset smaller than FCNN1 (FCNN2 resp.).

On the contrary, the FCNN1 and FCNN2 execute few iterations and are noticeably faster, with the FCNN2 being the fastest. This can be explained by noticing that it appears to be little sensitive to the complexity of the decision boundary. Indeed, it rapidly covers regions of the space far from the centroids of the classes and always performs about a few tens of iterations.

The FCNN1 is slightly slower than the FCNN2 and it requires more iterations, up to a few hundreds. But,

together with the FCNN3, it is likely to select points very close to the decision boundary and hence may return a subset smaller than that of the FCNN2. From what has been stated above, as far as the subset computed is concerned, the FCNN1 and FCNN3 are probably preferable when the classes are well-separated (for an example, see the subsets of the *Checkerboard* data set in Figure 7).

The FCNN4 rule presents the greatest area under the curve of the training set accuracy versus the current subset size. Hence, it can be stopped early, when a satisfactory degree of training set accuracy is achieved, to obtain a noticeably smaller condensed subset (as an example, the first thousand points selected on the *Checkerboard* data set guarantee a 99% accuracy on the training set, though the final set includes about seven thousands points). This strategy can be profitably used also with the other FCNN rules.

The FCNN rule is order independent, its worst case time complexity is quadratic with an often small constant pre-factor, and permits the triangle inequality to be effectively exploited to reduce computational effort (as witnessed by Figure 10). It was tested on large and high-dimensional training sets with very good results. For example, on an ordinary personal computer it was able to compute a consistent subset (including about four thousands out of about half a million points) of the *DARPA 1998* data set in more or less twenty seconds.

The FCNN rule was compared with the CNN, MCNN, and NNSRM algorithms.

Comparison on small data yielded the following outcomes. As regards classification accuracy, there are no appreciable differences, since all the methods present a comparable loss (about the 1.5%) with respect to the whole training set. The only exception is the NNSRM, that scored the same accuracy as the training set. Nevertheless, as far as the condensation size is concerned, the MCNN is the only rule comparable with the FCNN. Indeed, the CNN computes a subset from 10% to 15% larger than that of the FCNN, while the NNSRM tends to select almost all the training set objects and, hence, the slightly better accuracy it offers is not repaid by an appreciable reduction of the training set.

Existing condensation algorithms are too slow to be applicable to large data sets, since their complexity is super-quadratic or even cubic. Thus, improved implementations of the CNN and MCNN rules, called respectively fCNN and fMCNN, are introduced here, and compared to the FCNN. Experiments show that the FCNN rule outperforms even these enhanced methods while guaranteeing the same accuracy as the training set. It is worth noticing that, as far as the classification accuracy is concerned, this is the expected behavior of the compared methods, since their goal is to preserve the competence of the whole training set.

The observed superior learning speed is also substantiated by the learning behavior comparison. Indeed, the FCNN rules approach consistency more rapidly (see

Figure 8) and return a smaller subset.

A variant of the FCNN taking into account  $k$  nearest neighbors has been introduced and experimented on two real data sets. By increasing  $k$ , the execution time slightly increases, the size of subset remains almost the same, and test accuracy may improve.

The comparison with hybrid methods on medium-sized (including from ten to twenty thousands examples) real-world (character and face recognition) high-dimensional (up to 784 features) data sets, showed that FCNN is at least three order of magnitude faster, and that it achieves both a reduction ratio and a test accuracy comparable to DROP3 (see Figure 12), although DROP3 incorporates a data set editing step able to filter out noise and producing a slightly smaller subset. ELGrow/Explore perform noticeable worse than FCNN in terms of accuracy, but have higher reduction ratios. From the scaling analysis, it is clear that hybrid methods are impractical on larger data sets. Thus, in real-world domains, the FCNN can compute a model comparable to that of hybrid algorithms. The FCNN is efficient even if the collection of data to be processed is very large, while other algorithms may be not able to manage the same data set sizes in a reasonable amount of time.

To conclude, it is worth pointing out that this is the first work providing condensation algorithms for the nearest neighbor rule that can be efficiently applied to large data sets.

**Acknowledgments.** The author would like to thank the anonymous reviewers for their useful comments.

## REFERENCES

- [1] D.W. Aha. Editorial, special ai review issue on lazy learning. *Artificial Intelligence Review*, 11(1-5):7–10, 1997.
- [2] D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
- [3] E. Alpaydin. Voting over multiple condensed nearest neighbors. *Artificial Intelligence Review*, 11:115–132, 1997.
- [4] F. Angiulli. Fast condensed nearest neighbor rule. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 25–32, 2005.
- [5] S. Bay. Combining nearest neighbor classifiers through multiple feature subsets. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.
- [6] S. Bay. Nearest neighbor classification from multiple feature sets. *Intelligent Data Analysis*, 3:191–209, 1999.
- [7] B. Bhattacharya and D. Kaller. Reference set thinning for the  $k$ -nearest neighbor decision rule. In *Proceedings of the 14th International Conference on Pattern Recognition*, 1998.
- [8] H. Brighton and C. Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining and Know. Discovery*, 6(2):153–172, 2002.
- [9] R.M. Cameron-Jones. Instance selection by encoding length heuristic with random mutation hill climbing. In *Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence*, pages 99–106, 1995.
- [10] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.
- [11] B. Dasarthy. *Nearest Neighbor (NN) Norms—NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, CA, 1991.
- [12] B. Dasarthy. Minimal consistent subset (mcs) identification for optimal nearest neighbor decision systems design. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3):511–517, 1994.

- [13] B. Dasarthy. Nearest unlike neighbor (nun): an aid to decision confidence estimation. *Optical Engineering*, 34:2785–2792, 1995.
- [14] F.S. Devi and M.N. Murty. An incremental prototype set building technique. *Pattern Recognition*, 35(2):505–513, 2002.
- [15] P. Devijver and J. Kittler. On the edited nearest neighbor rule. In *Proceedings of the Fifth International Conference on Pattern Recognition*, pages 72–80, 1980.
- [16] L. Devroye. On the inequality of cover and hart in nearest neighbor discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:75–78, 1981.
- [17] L. Devroye, L. Györfy, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, 1996.
- [18] K. Fukunaga and L.D. Hostetler.  $k$ -nearest-neighbor bayes-risk estimation. *IEEE Transactions on Information Theory*, 21:285–293, 1975.
- [19] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.
- [20] W. Gates. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 18(3):431–433, 1972.
- [21] P.E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516, 1968.
- [22] B. Karaçali and H. Krim. Fast minimization of structural risk by nearest neighbor rule. *IEEE Transactions on Neural Networks*, 14(1):127–134, 2003.
- [23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [24] C.L. Liu and M. Nakagawa. Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to hand-written character recognition. *Pattern Recognition*, 34(3):601–615, 2001.
- [25] G.L. Ritter, H.B. Woodruff, S.R. Lowry, and T.L. Isenhour. An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21:665–669, 1975.
- [26] C. Stanfill and D. Waltz. Towards memory-based reasoning. *Communications of the ACM*, 29:1213–1228, 1994.
- [27] C. Stone. Consistent nonparametric regression. *Annals of Statistics*, 8:1348–1360, 1977.
- [28] G. Toussaint. Proximity graphs for nearest neighbor decision rules: Recent progress. In *Proceedings of the Symposium on Computing and Statistics*, Montreal, Canada, April 17–20 2002.
- [29] I. Watson and F. Marir. Case-based reasoning: A review. *The Knowledge Engineering Review*, 9(4), 1994.
- [30] G. Wilfong. Nearest neighbor problems. *International Journal of Computational Geometry & Applications*, 2(4):383–416, 1992.
- [31] D.L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, 2:408–420, 1972.
- [32] D.R. Wilson and T.R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.

## APPENDIX

**Proof of Theorem: 3.2** (i) First, we note that the time required to compute the class centroids of the training set and to perform a single iteration of the algorithm is upper bounded by a polynomial in the number  $|T|$  of training set instances.

During a generic iteration of the algorithm at least one element of  $T - S$  is selected and inserted in  $S$ , otherwise the algorithm stops. Let's call  $m$  the number of class labels in the training set. Since subset  $S$  contains initially  $m$  elements, in the worst case the algorithm performs  $|T| - m + 1$  iterations.

The training set  $T$  being composed of a finite number of elements, it can be concluded that the overall time required by the method is finite.

(ii) The property is guaranteed by the termination condition. Indeed, the algorithm stops when the set  $\Delta S$  becomes empty, that is, if the condition of Theorem 3.1 is satisfied.

(iii) First of all we show that, given a (not necessarily proper) subset  $X$  of  $T$ , it holds that (a) the set  $Centroids(X)$ , composed of the class centroids of  $X$ , is order independent, (b) point  $nn(p, X)$  of  $X$  closest to  $p$  is order independent, (c) for each  $p \in X$ , the Voronoi cell  $Vor(p, X, T)$  is order independent, and (d) point  $p^*$  of  $X$  such that  $|Voren(p^*, X, T)|$  is maximum is order independent.

(a) Consider the set  $Centroids(X)$ . Let  $l_1, \dots, l_m$  be the class labels in  $X$ , and let  $X_i$  be the subset of  $X$  composed of the points of the class  $l_i$ . Then, the set  $Centroids(X)$  has the form  $\{c_1, \dots, c_m\}$ , where each  $c_i$  is the centroid of  $X_i$ , that is the point of  $X_i$  which is closest to the geometrical center  $C_i$  of  $X_i$ . Clearly,  $C_i$  is unique. As for  $c_i$ , usually it is unique, but ties are possible and are solved in favor of the lexicographically smallest point.

Let  $q = (q_1, \dots, q_d)$  and  $r = (r_1, \dots, r_d)$  be two points. Then  $q$  is lexicographically smaller than  $r$  if there is an integer  $j$ ,  $1 \leq j \leq d$ , such that  $q_1 = r_1, \dots, q_{j-1} = r_{j-1}$ , and  $q_j < r_j$ . If there is more than one lexicographically smallest point, then it is the case that these points are identical, and no matter which is selected. Thus, the set computed is independent of the order in which  $X_i$  is processed.

(b) Consider point  $nn(p, X)$ . Usually, the point of  $X$  closest to  $p$  is unique, but ties are possible and are solved in favour of the lexicographically smallest point. Nevertheless, it can be that there are two lexicographically smallest points  $q$  and  $r$  having different class labels  $l_q$  and  $l_r$ . In this case  $q$  is selected, provided that  $l_q < l_r$ . Hence, the point selected is independent of the order in which  $X$  is processed.

(c) Usually, for each point  $q \in T$  the nearest neighbor of  $q$  in  $X$  is unique, and thus also the Voronoi cell which it belongs to, but ties are possible and are solved as explained in point (b) above.

(d) We have already seen that the set  $Vor(p, X, T)$  is order independent. Thus, for each  $p \in X$  sets  $Voren(p, X, T)$  are order independent. Usually point  $p^* \in X$  maximizing  $|Voren(p^*, X, T)|$  is unique, but ties are possible and are solved in favor of the lexicographically smallest point.

It remains to show that, for each iteration  $i \geq 0$ , the set of points  $\Delta S_i$  selected by the algorithm is independent of the order in which training set elements are processed. The proof now proceeds by induction on the iteration number.

Let  $i = 0$ , then  $S_0 = \emptyset$  and  $\Delta S_0$  is the set  $Centroids(T)$ . Hence, the result follows from point (a).

Let  $i > 0$ , and assume that  $\Delta S_0, \dots, \Delta S_{i-1}$  are order independent. Then,  $S_i = S_{i-1} \cup \Delta S_{i-1} = S_{i-2} \cup \Delta S_{i-2} \cup \Delta S_{i-1} = \dots = \Delta S_0 \cup \dots \cup \Delta S_{i-1}$  is order independent. Furthermore, for each  $p \in S_i$ , (i)  $Voren(p, S_i, T)$  is order independent being a particular subset of  $Vor(p, S_i, T)$ , and (ii)  $rep(p, Voren(p, S_i, T))$  is order independent, as both  $nn(p, Voren(p, S_i, T))$  and  $nn(p, Centroids(Voren(p, S_i, T)))$  are order independent. Hence, it can be concluded that  $\Delta S_i$  is order

independent, and the result follows.

□