Fast Condensed Nearest Neighbor Rule

Fabrizio Angiulli ANGIULLI@ICAR.CNR.IT
ICAR-CNR, Via Pietro Bucci 41C, 87036 Rende (CS), Italy

Abstract

We present a novel algorithm for computing a training set consistent subset for the nearest neighbor decision rule. The algorithm, called FCNN rule, has some desirable properties. Indeed, it is order independent and has subquadratic worst case time complexity, while it requires few iterations to converge, and it is likely to select points very close to the decision boundary. We compare the FCNN rule with state of the art competence preservation algorithms on large multidimensional training sets, showing that it outperforms existing methods in terms of learning speed and learning scaling behavior, and in terms of size of the model, while it guarantees a comparable prediction accuracy.

1. Introduction

The nearest neighbor decision rule (Cover & Hart, 1967) (NN rule in the following) 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 (Cover & Hart, 1967; Stone, 1977; Devroye, 1981). 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 to store all the previously classified data, and to compare then each sample point to be classified to each stored point. In order to reduced both space and time requirements, several techniques to reduce the size of the stored data for the NN rule have been proposed (see (Wilson & Martinez, 2000; Toussaint, 2002) 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, (Karaçali & Krim, 2002) showed that the VC dimension of a 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 (Wilfong, 1992). Several training set consistent condensation algorithms have been introduced in literature. We point out that, among the criteria characterizing condensation methods, the learning speed one is usually neglected. But, in order to manage huge amounts of data, methods exhibiting good scaling behavior are definitively needed. In this work we present a novel *order independent* algorithm for finding a training set consistent subset for the NN rule, called FCNN rule, and compare it with existing methods. The rest of the paper is organized as follows. In Section 2, existing approaches are briefly described and compared to the approach here proposed. 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, conclusions are drawn.

2. Related Works and Contribution

Starting from the seminal work of (Hart, 1968), several training set condensation algorithms have been introduced, also known as instance-based, lazy, memory-based, and case-based learners. These methods can be grouped into three categories depending on the objectives that they want to achieve (Brighton & Mellish,

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

2002), i.e. *competence preservation*, *competence enhancement*, and *hybrid approaches*. 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 are combined in order to achieve the same objectives of hybrid methods. 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 *per se* relevant task, improving both classification speed and classifier accuracy (Karaçali & Krim, 2002), and computationally hard (Wilfong, 1992). Next, we first briefly describe incremental 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 (Wilson & Martinez, 2000; Toussaint, 2002).

The concept of a training set consistent subset was introduced by P.E. Hart (Hart, 1968) together with an algorithm, called 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. The MCNN rule (for Modified CNN rule) (Devi & Murty, 2002) 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 centroids 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_m of points that are correctly classified and misclassified respectively by using the centroids C. If the set S_m 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_m 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 on 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 could require a lot of iterations to converge. In order to compute a small consistent subset S of the training set T, (Karaçali & Krim, 2002) proposed the following algorithm (NNSRM for Structural Risk Minimization using the NN rule). Assume that the training set T contains only two class labels. First, all pairwise distances among points having different class labels are computed. Let $\{x_i, y_i\}$ denote the pair having the ith 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 the 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 $k+1$. Nevertheless, the method is costly. Indeed, the complexity of the NNSRM algorithm is $\mathcal{O}(|T|^3)$. Furthermore, we note that the size of the condensed set computed is sensitive to the pairs having greatest distances. Indeed, assume that two points from opposite classes in the training set are far from the other training set points, and such that their distance is greater than the distance of each other pair of the remaining points from opposite classes. Then, a training set consistent subset must contain these two points and, hence, all the training set points. Finally, we recall the RNN rule (Gates, 1972), a costly post-processing step that can be applied to every competence preservation method, the SNN rule (Ritter et al., 1975), having worst case exponential running time, the MCS algorithm (Dasarathy, 1994), involving computation of the so called nearest unlike neighbors, that intends to compute a subset close to the minimal one, and approximate optimization techniques (Liu & Nakagawa, 2001), that can be applied only to small sized data set.

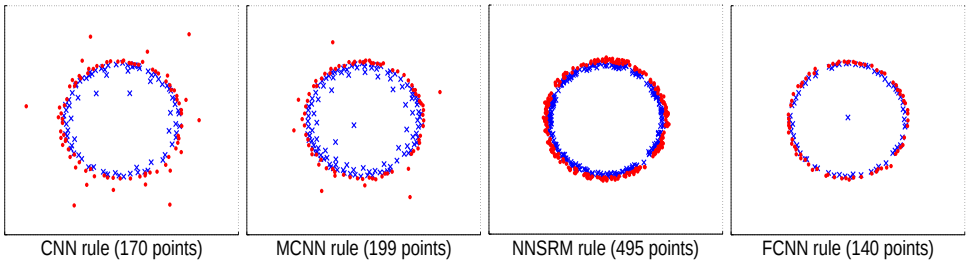

Figure 1. Example of training set consistent subsets computed by the CNN, MCNN, NNSRM, and FCNN rules.

2.1. Contribution

In this work we present a novel algorithm for the computation of a training set consistent subset for the NN rule. The algorithm, that we call Fast CNN rule (FCNN for short), works as follows. First, the consistent subset S is initialized to the centroids[1] of the classes contained in the training set T. Then, during each iteration of the algorithm, for each point p in S, a point q of T belonging to the Voronoi cell of p,[2] 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, has sub-quadratic time complexity, 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 by 9,000 points uniformly distributed into the unit square and partitioned in two classes by a circle of diameter 0.5. As already noted elsewhere (Wilson & Martinez, 2000), 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. But, 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. (*i*) We present a novel *order independent* algorithm, called FCNN rule, for the computation of a training set consistent subset for the NN rule. It is the second proposal of an order independent algorithm after the MCNN rule (Devi & Murty, 2002). (*ii*) We prove that the FCNN rule has worst case sub-quadratic time requirements, and describe an implementation exploiting the triangular inequality that sensibly reduces the worst case computational cost. (*iii*) We compare the FCNN rule with state of the art competence preservation algorithms on large and high dimensional training sets, showing that the FCNN rule outperforms existing methods in terms of learning speed and learning scaling behavior, and in terms of size of the model, while it guarantees a comparable prediction accuracy.

[1]Given a set S of points having the same class label, the centroid of S is the point of S which is closest to the geometrical center of S.

[2]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.

3. Algorithm

We first give 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. We denote by $l(p)$ the label associated to p. Given a point q, the NN rule $\mathrm{NN}(q,T)$ assigns to q the label of the nearest neighbor of q in T, i.e. $\mathrm{NN}(q,T) = l(nn(q,T))$. A subset S of T is said to be a *training set consistent subset of* T if, for each $p \in T$, $l(p) = \mathrm{NN}(p,S)$. Let S be a subset of T, and let p be an element of S. We denote by $\mathrm{Vor}(p,S,T)$ the set $\{q \in T \mid \forall p' \in S, \mathrm{d}(p,q) \leq \mathrm{d}(p',q)\}$, 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 $\mathrm{Voren}(p,S,T)$ the set $\{q \in \mathrm{Vor}(p,S,T) \mid l(q) \neq l(p)\}$. 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, $\mathrm{Voren}(p,S,T)$ is empty.

3.1. The FCNN Rule

The algorithm FCNN rule is shown in Figure 2. It initializes the consistent subset S with a seed element from each class label of the training set T. In particu-

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 $(p \in S)$
      $\Delta S = \Delta S \cup \{rep(p, \text{Voren}(p, S, T))\}$;
  }
  return(S);

Figure 2. The FCNN rule.

lar, the seeds employed are the centroids of the classes in T. The algorithm is incremental: 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 $rep(p, \text{Voren}(p, S, T))$ in Figure 2, is selected and inserted into S. Given $p \in T$ and a subset $X \subseteq T$, the representative $rep(p, X)$ of X w.r.t. p can be defined in different ways. We investigate the behavior of two different definitions for $rep(p, X)$. 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 $rep(p, X) = nn(p, X)$. The second definition, we call FCNN2 the variant using it, selects the class centroid in X closest to p, that is $rep(p, X) = nn(p, Centroids(X))$. 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. Figure 4 reports an example of execution of the FCNN1 rule. The data set considered is composed by 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 algorithm performs nine iterations and returns a subset composed by 54 points (the FCNN2 rule on the same data set performs ten iterations and computes a subset composed by 74 points). It is clear from these figures that the FCNN rule rapidly converges 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. 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.*

Figure 3 shows the implementation of the FCNN1 rule.

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

Figure 3. The FCNN1 rule.

The algorithm maintains in the array *nearest*, for each training set point q, the closest point $nearest[q]$ of q in the set S, and in the array *rep*, for each point p in S, its current representative $rep[p]$ of the misclassified points lying in the Voronoi cell of p. During each iteration, the array *nearest* and *rep* must be updated. Let ΔS 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 ΔS, 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 ΔS, 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 FCNN rule requires at most* $|T| \cdot |S|$ *distance computations using* $\mathcal{O}(|T|)$ *space.*

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 Voroni cell induced by the

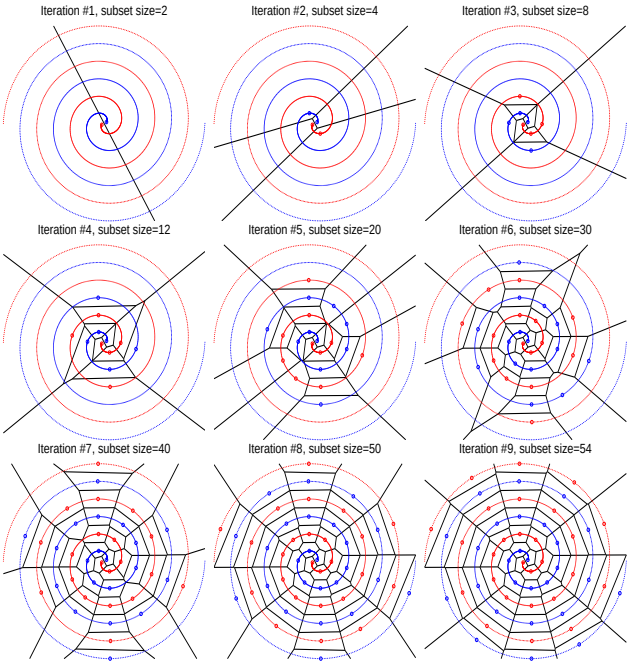

Figure 4. Example of execution of the FCNN1 rule.

points in S, the FCNN2 rule performs at the end of each main iteration a supplemental training set scan.

3.2. Exploiting the Triangular Inequality

In a metric space the FCNN rule can take advantage of the triangular inequality in order to avoid the comparison of each point in $(T - S)$ with each point in ΔS. During the generic ith main iteration, for each $p \in S_{i-1}$, the Voronoi cell $\mathrm{Vor}(p, S_{i-1}, T)$[3] is visited, and the points $q \in \mathrm{Vor}(p, S_{i-1}, T)$ are compared with the points in ΔS_i. Before starting the comparison, the distances among the point p and the points in ΔS_i are computed, and the points in ΔS_i are sorted in order of increasing distance from p. Then, instead of comparing each point q in $\mathrm{Vor}(p, S_{i-1}, T)$ with each point in ΔS_i, each q is compared with the points in ΔS_i having distance from p less than twice the distance from q and p. Indeed, by the triangular inequality, they are all and the only points of ΔS_i candidate to be closer to q than p. In the worst case, this implementation of the FCNN rule requires $|T| \cdot |S| + \sum_{i<j} |\Delta S_i| \cdot |\Delta S_j| \cdot \log |\Delta S_j|$ distance computations, i.e. slightly worse than the upper bound of Theorem 3.3, and has no additional space requirements. Nevertheless, the last implementation guarantees great savings in terms of distances computed, and definitively outperforms the previous one.

[3]Partitioning of points in Voronoi cells is induced by the array *nearest*.

Data set name	*Size*	*Features*	*Classes*
Census	190,495	13	2
Chessboard	64,000	2	2
Forest Cover Type	100,000	54	7
DARPA	458,301	22	5
Shuttle	43,500	9	7
Spiral	1,000,000	2	2

Table 1. Data sets used in the experiments.

4. Experimental Results

In this section we present a number of experiments performed using the FCNN rule, and a thorough comparison with the CNN and MCNN methods. The NNSRM rule is impracticable on large training sets, hence we do not consider it further. As for the FCNN rule, we used the algorithm exploiting the triangular inequality described in Section 3.2. As for the MCNN rule, we augmented the method with the technique exploiting the triangular inequality described in Section 3.2. As for the CNN rule, we avoided repeated distance computations. The data set employed are summarized in Table 1, and briefly described next. *Census* is composed by data extracted from the census bureau database[4]. *Chessboard* is a synthetic data set composed by points uniformly distributed into the unit square and grouped in two classes representing cells of a 8×8 chessboard. *Forest Cover Type* contains data from the US Forest Service[5]. The *DARPA* 1998 intrusion detection data consists of network connection records of several intrusions[6]. *Shuttle* was used in the European StatLog project[5]. *Spiral* is a synthetic data set composed by two concentric spirals representing two distinct classes.

Execution time, size and accuracy. Table 2 compares the FCNN1, FCNN2, MCNN, and CNN rules on the data sets above described, using the Euclidean distance. The table reports execution time[7], number of iterations performed, empirical complexity, size of the condensed set, and classification accuracy obtained using the condensed set. In order to estimate the classification accuracy of each rule, we used 10-fold cross-validation. 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 it-

[4]`www.census.gov/ftp/pub/DES/www/welcome.html`.
[5]`www.kdd.ics.uci.edu`.
[6]`www.ll.mit.edu/IST/ideval/index.html`.
[7]We ran the experiments on a Pentium IV 2.66GHz based machine.

Data set name	Method	Execution time [sec]	Iterations	Empirical complexity	Size	Accuracy [%]
Census	FCNN1	240.33	36	1.69	**24,614**	93.06
	FCNN2	**185.42**	26	**1.62**	25,248	93.05
	MCNN	4,283.73	19,251	1.86	27,666	93.02
	CNN	1,818, 77	**6**	1.84	27,836	**93.11**
	training set	–	–	–	171,445	94.61
Chessboard	FCNN1	4.17	54	1.56	**3,082**	**97.14**
	FCNN2	**3.03**	20	**1.47**	3,707	**97.14**
	MCNN	84.36	2,019	1.76	3,995	97.03
	CNN	28.30	**5**	1.75	3,812	97.11
	training set	–	–	–	57,599	98.75
Forest Cover Type	FCNN1	184.34	33	1.67	**15,972**	90.68
	FCNN2	**178.06**	24	**1.65**	16,255	**90.90**
	MCNN	2,341.81	7,938	1.86	16,560	90.66
	CNN	1,294, 25	**5**	1.84	16,920	90.69
	training set	–	–	–	90,000	92.82
DARPA	FCNN1	63.09	245	1.44	**3,252**	**99.22**
	FCNN2	**27.01**	23	**1.37**	3,803	99.20
	MCNN	583.91	2,566	1.56	4,275	99.21
	CNN	732.81	**14**	1.64	4,078	99.20
	training set	–	–	–	412,470	99.31
Shuttle	FCNN1	**0.35**	21	1.36	**225**	**99.75**
	FCNN2	0.67	16	**1.35**	257	99.69
	MCNN	1.67	144	1.45	286	99.71
	CNN	1.66	**4**	1.53	280	99.71
	training set	–	–	–	39,150	99.77
Spiral	FCNN1	**2.39**	9	**1.21**	**57**	**100.00**
	FCNN2	3.75	10	1.22	80	**100.00**
	FMCNN	22.01	46	1.37	88	**100.00**
	CNN	6.30	**2**	1.32	77	**100.00**
	training set	–	–	–	900,000	100.00

Table 2. Experimental results.

erations and by the number of training set passes per iteration. Next, we comment on the results shown in Table 2. As for the number of iterations, we observe that the CNN rule requires few iterations to converge, the MCNN rule requires a lot of iterations, while the FCNN1 and FCNN2 require a little number of iterations. The FCNN rule requires more iterations than the CNN rule, but notably less than the MCNN rule. We recall that the CNN and the FCNN1 rule perform one training set pass per iteration, that the FCNN2 rule performs two training set passes per iteration, and that the MCNN rule performs several training set passes per iteration. As for the empirical complexity, the FCNN1 and FCNN2 rules outperform both the MCNN and the CNN rule. The FCNN2 rule performs in some cases slightly better than the FCNN1 rule, while the CNN and the MCNN rule seem comparable. As for the execution time, the FCNN1 and FCNN2 rules outperform both the two other rules by at least an order of magnitude. This is a consequence of the fact that the FCNN rule has a lower empirical complexity, requires few iterations to converge, and one (or two) training set pass per iteration. As for the size of the training set consistent subset, the set computed by the FCNN1 rule was always the smallest. Finally, as for the classification accuracy, the FCNN rules achieves almost always the best values, though the quality of classification is about the same for all the methods.

Learning behavior. Figure 5 shows the learning behavior of the FCNN1, FCNN2, MCNN, and CNN rules on the data sets *Census*, *Forest Cover Type*, and *DARPA*. The curves show (from left to right) the accuracy of the subset S_i computed during the ith iteration of the algorithms in classifying the overall training set T, the number of points $|\Delta S_i|$ added to S_i during each iteration, the size $|S_i|$ of the subset S_i versus the normalized iteration number $\frac{i}{i_{\max}}$, and the distances computed during each iteration. As for the accuracy (first column), we note that the FCNN2 rule smoothly converges to consistency, while the curve of the FCNN1 rule swings during the initial iterations (notice that the graphic of the DARPA data set is in logarithmic scale). We note also that the area under the curve of the FCNN2 rule is always the greatest. As for the size of ΔS_i (second column), interestingly the FCNN rule

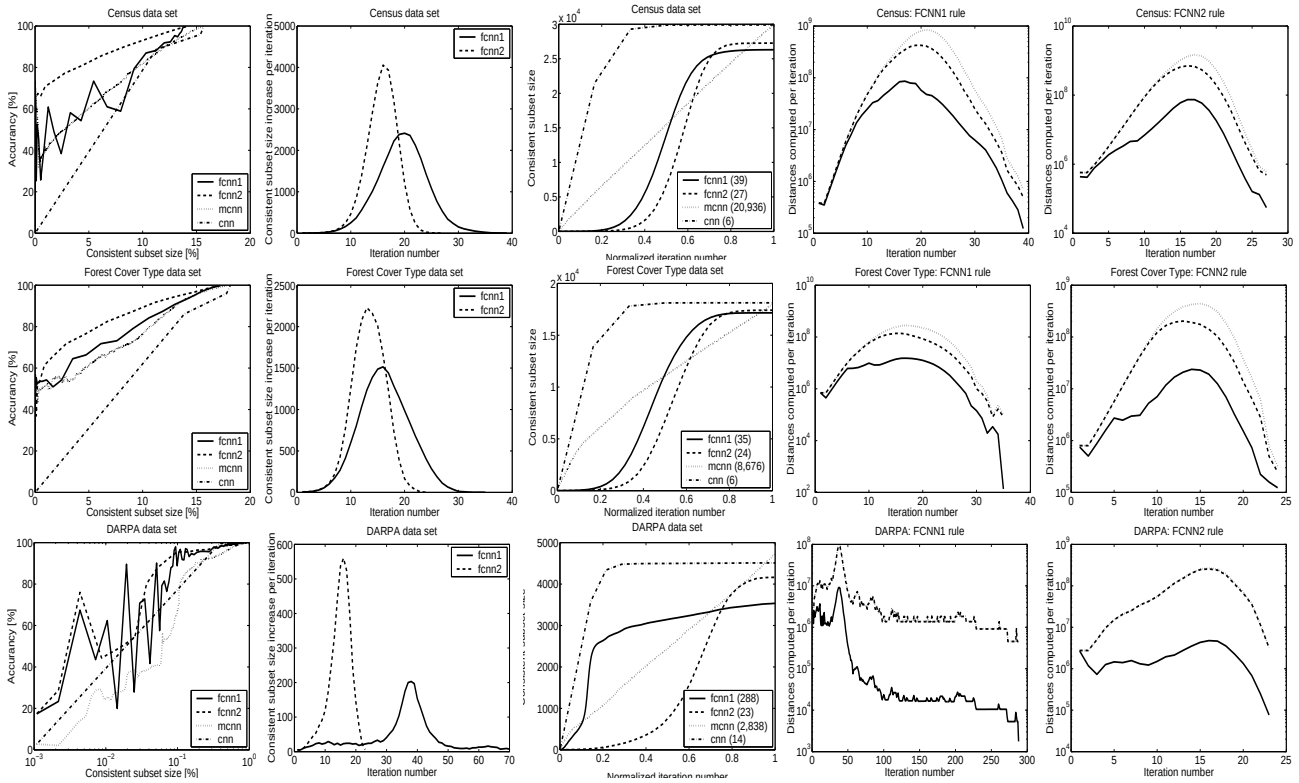

Figure 5. Learning behavior.

follows a gaussian-like distribution. Though the area under the curves of the FCNN1 and FCNN2 rules is nearly identical, we note that the FCNN1 rule reaches its maximum later, and hence requires more iterations to converge. Analogous considerations hold for the course of $|S_i|$ (third column). The number of iterations performed by the methods is reported in the figure legend. We note that the FCNN1 rule always computes the smallest consistent subset. The curve of the MCNN rule is a straight line as the number of points added during each iteration is almost constant and given by the number of class labels in the training set, the curve of the CNN rule is a step as almost all the points are selected during the first and second iteration, while the curves of the FCNN rule resemble a sigmoid. Finally, as for the distances computed (fourth and fifth columns), solid lines represent the number of distances actually calculated by the FCNN rule exploiting the triangular inequality, while dotted lines represent the worst case cost of the same algorithm. Dashed lines represent the distances required when the triangular inequality is not exploited. 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.

Scaling analysis. Figure 6 shows the scaling analysis on the training sets *Census*, *Forest Cover Type*, and *DARPA*. As for the execution time (first column), both the FCNN1 and FCNN2 rules clearly outperform the other two rules. The CNN rule performs better than the MCNN rule on the first two data sets. We note that the number of iterations performed (second column) by the FCNN and the CNN rules is almost constant with the training set size, while the iterations performed by the MCNN rule increases of some order of magnitude with the training set size. As for the empirical complexity (third column) the FCNN rule performs better. The FCNN2 rule was always the best. On the *Census* data set the empirical complexity increases with the size, while on the *DARPA* data set it decreases, for all methods. *Forest Cover Type* is hard to manage by the CNN and the MCNN rules, whose empirical complexity is constant, while it less difficult for the FCNN rule, whose empirical complexity decreases with the data set size. Finally, as for the consistent subset size (fourth column), the curves of the various methods have the same behavior, but it is worth to note that on these data sets the FCNN rules computes the smallest training set consistent subsets.

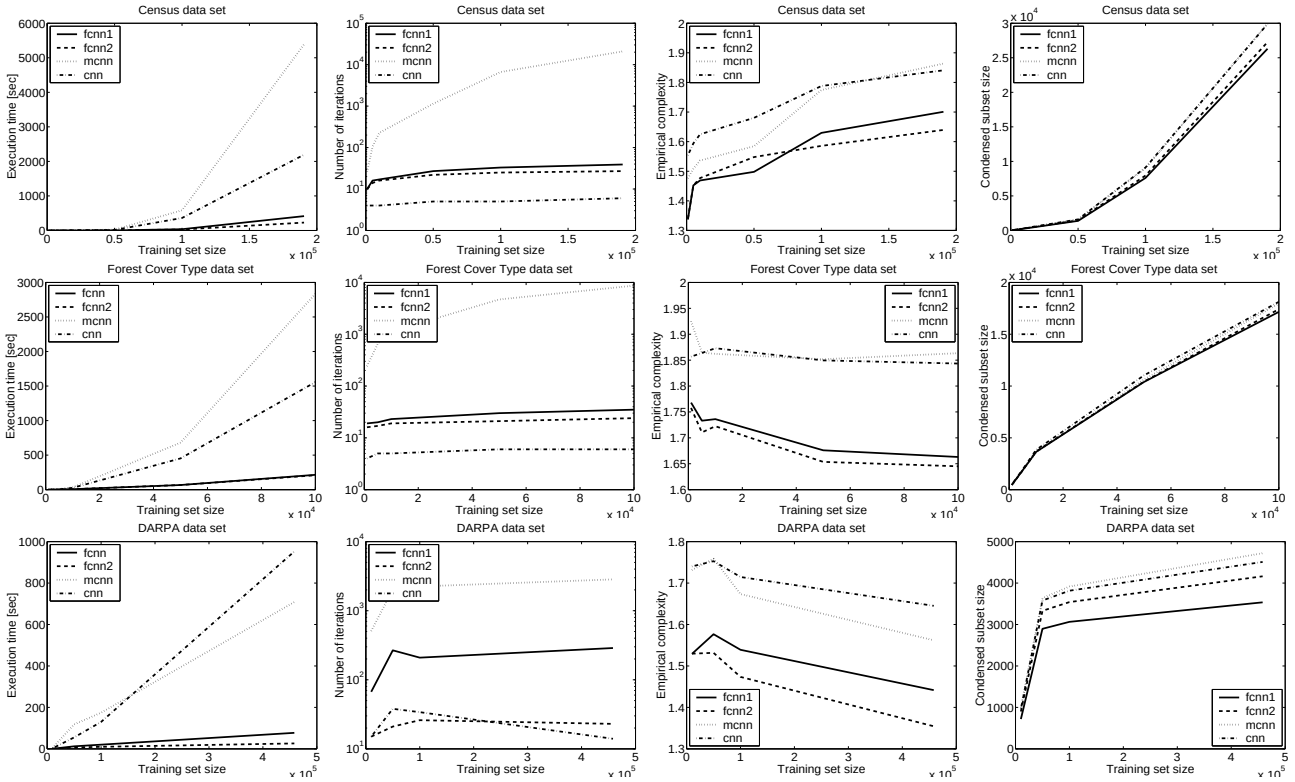

Figure 6. Scaling analysis.

5. Conclusions

We presented a novel order independent method for computing a training set consistent subset for the NN rule and compared it with existing state of the art competence preservation methods. The observed superior learning speed of the new method is substantiated by the learning behavior comparison. This work can be extended in several ways, e.g., studying the impact of different metrics on the FCNN rule, and the behavior of FCNN-based hybrid methods.

References

Brighton, H., & Mellish, C. (2002). Advances in instance selection for instance-based learning algorithms. *Data Mining and Know. Discovery*, *6*, 153–172.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Trans. on Inform. Th.*, *13*, 21–27.

Dasarathy, B. (1994). Minimal consistent subset (mcs) identification for optimal nearest neighbor decision systems design. *IEEE Trans. on Sys. Man. Cybernet.*, *24*, 511–517.

Devi, F., & Murty, M. (2002). An incremental prototype set building technique. *Pat. Recognition*, *35*, 505–513.

Devroye, L. (1981). On the inequality of cover and hart. *IEEE Trans. on Pat. Anal. and Mach. Intel.*, *3*, 75–78.

Gates, W. (1972). The reduced nearest neighbor rule. *IEEE Trans. on Inform. Th.*, *18*, 431–433.

Hart, P. (1968). The condensed nearest neighbor rule. *IEEE Trans. on Inform. Th.*, *14*, 515–516.

Karaçali, B., & Krim, H. (2002). Fast minimization of structural risk by nearest neighbor rule. *IEEE Trans. on Neural Networks*, *14*, 127–134.

Liu, C., & Nakagawa, M. (2001). Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to handwritten character recognition. *Pat. Recognition*, *34*, 601–615.

Ritter, G., Woodruff, H., Lowry, S., & Isenhour, T. (1975). An algorithm for a selective nearest neighbor decision rule. *IEEE Trans. on Inform. Th.*, *21*, 665–669.

Stone, C. (1977). Consistent nonparametric regression. *Annals of Statistics*, *8*, 1348–1360.

Toussaint, G. (2002). Proximity graphs for nearest neighbor decision rules: Recent progress. *Proc. of the Symp. on Computing and Stat.*. Montreal, Canada.

Wilfong, G. (1992). Nearest neighbor problems. *Int. J. of Computational Geometry & Applications*, *2*, 383–416.

Wilson, D., & Martinez, T. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning*, *38*, 257–286.