

# Hit Miss Networks with Applications to Instance Selection

Elena Marchiori  
Department of Information and Knowledge Systems  
Faculty of Science  
Radboud University, Nijmegen, The Netherlands

## Abstract

In supervised learning, a training set consisting of labeled instances is used by a learning algorithm for generating a model (classifier) that is subsequently employed for deciding the class label of new instances (for generalization). Characteristics of the training set, such as presence of noisy instances and size, influence the learning algorithm and affect generalization performance. This paper introduces a new network-based representation of a training set, called hit miss network (HMN), which provides a compact description of the nearest neighbor relation between each pair of classes. We show that structural properties of HMN's correspond to properties of training points related to the one nearest neighbor (1-NN) decision rule, such as being border or central point. This motivates us to use HMN's for improving the performance of a 1-NN classifier by removing instances from the training set (instance selection). We introduce three new algorithms based on HMN for instance selection. HMN-C, which removes instances without affecting accuracy of 1-NN on the training set, HMN-E, which removes more instances than HMN-C, and HMN-EI, which applies iteratively HMN-E. Their performance is assessed on 22 datasets with different characteristics, such as input dimension, cardinality, class balance, number of classes, noise content, and presence of redundant variables. Results of experiments on these datasets show that accuracy of 1-NN classifier increases significantly when HMN-EI is applied. Comparison with state-of-the-art editing algorithms for instance selection on these datasets indicates best generalization performance of HMN-EI and no significant difference in storage requirements. In general, these results indicate that HMN's provide a powerful graph-based representation of a training set, which can be successfully applied for performing noise and redundancy reduction in instance-based learning.

**Keywords:** Graph-based training set representation, nearest neighbor, instance selection for instance-based learning.

## 1 Introduction

In supervised learning, a machine learning algorithm is given a training set, consisting of training examples called labeled *instances* (here called also *points*). Each instance consists of an input vector of values, one for each *variable* of the learning task, and has assigned a class label. A machine learning algorithm uses the training set to generate a so-called *model* that is subsequently used for deciding the class label of (*classify*) new instances. In particular, the 1-NN rule classifies an unknown point into the class of the nearest of the training set points. This rule does not rely on knowledge of the underlying data distribution (non-parametric classification). Moreover, for all distributions, its probability of error is bounded above by twice the Bayes' probability of error ([11]).

A central issue in 1-NN classification, and more generally in instance-based learning, concerns storage requirements. The basic 1-NN rule stores all training instances, hence can be slow when classifying new instances. Moreover, when the training set contains noisy instances, generalization accuracy can be negatively affected if these instances are stored as well (cf. [37]). Instance selection algorithms tackle these issues by selecting a subset of the training set in order to reduce storage and possibly also enhance accuracy of the 1-NN rule on new instances (generalization performance).

In this paper we introduce a new graph-based representation of a training set, called Hit Miss Network. In an HMN, nodes are instances of the considered training set. Edges are defined as follows: for each node  $x$  and for each class, there is a directed edge from  $x$  to its nearest neighbor among training set instances belonging to that class. Thus HMN represents a 'more specific' nearest neighbor relation, namely between *each* pair of classes.

We show that structural properties of HMN's correspond to properties of training instances related to the decision boundary of the 1-NN rule, such as being border or central point. These observations motivate the use of HMN for performing instance selection for the 1-NN rule. We introduce three new instance selection algorithms. The first algorithm, called HMN-C, discards instances corresponding to nodes of the HMN with no incoming edges (zero in-degree nodes). We prove that instance selection by means of this algorithm does not change the 1-NN classification of instances in the training set. The second algorithm, called HMN-E, employs a more aggressive deletion strategy, removing a larger number of training instances, including those with zero in-degree. The last algorithm, called HMN-EI, applies iteratively HMN-E. Such algorithms have the desirable properties of being order-independent and of having quadratic time complexity, which can be reduced using metric trees or other spatial data structures.

We assess effectiveness of the proposed algorithms with respect to generalization performance of the 1-NN rule and storage requirements, using 22 datasets with different characteristics, such as input dimension, cardinality, class balance, number of classes, noise content, and presence of redundant variables. Results of experiments show that HMN-EI improves significantly average accuracy of the 1-NN rule, and achieves significantly better performance than HMN-C and HMN-E. Experiments on the same datasets are conducted with the following three algorithms, which have been analyzed in Brighton and Mellish's paper on advances in instance selection [8]. Edited Nearest Neighbor (E-NN), designed for noise reduction [35], and two state-of-the-art editing algorithms: Iterative Case Filtering (ICF) [7] and the best of the Incremental Reduction Optimization algorithms introduced in [36] (DROP3). Comparison of the results shows that HMN-EI achieves best accuracy, with storage requirements similar to those of ICF and DROP3.

These results indicate that HMN's provide a powerful graph-based representation of training sets, with local structural properties useful for analyzing and enhancing 1-NN-based classification.

## Related Work

Graphs have been successfully used for representing relations between points of a given dataset, such as functional interaction between proteins (protein-protein interaction networks) or proximity (nearest neighbor graphs) [17]. Graph representations in the context of 1-NN instance-based learning mainly use proximity graphs. Proximity graphs are defined as graphs in which points close to each other by some definition of closeness are connected [3]. Representations of a dataset based on proximity graphs have been used with success to define

algorithms for reducing the size of the training set (e.g., [6]), for removing noisy instances (e.g., [28]), and for detecting critical instances close to the decision boundary (e.g., [4]), in order to improve storage and accuracy of 1-NN.

The nearest neighbor graph (NNG) is a typical example of proximity graph, where each vertex is a data point that is joined by an edge to its nearest neighbor. The minimum spanning tree (MST) is also a proximity graph. Graph-based applications to instance-based learning algorithms mainly use the Gabriel graph (GG). Both the NNG and MST are subgraphs of the GG. The GG is a subgraph of the Delaunay Triangulation (DT), the dual of the Voronoi diagram. The Voronoi diagram and correspondingly the DT of a point set capture all the proximity information about the point set because they represent the original 1-NN rule decision boundary. A so-called Voronoi condensed dataset is obtained by discarding all those points whose Voronoi cell shares a face with those cells that contain points of the same class [33]. The 1-NN decision boundary is the union of the common faces of the Voronoi diagram between Voronoi cell neighbors of different classes. The algorithm produces a decision-boundary consistent set. Voronoi condensing does not reduce the number of points to a great extent and its computational complexity in higher dimensions is exponential in the number of dimensions [33]. Exact computation of the Gabriel graph is cubic in the number of nodes.

In [4, 5, 6, 24, 28], faster algorithms for instance selection are investigated, which are based on the GG and the Reduced Neighborhood graph [22]. In particular, in [5] a specific data-structure for efficient computation of approximate Gabriel neighbor is proposed. Moreover, three instance selection algorithms are considered: Gabriel-Graph algorithm, ICF, and a so-called Hybrid. Hybrid incorporates E-NN, ICF, and the Gabriel graph rule. Specifically, it consists of the sequential application of a modified version of E-NN based on approximate Gabriel neighbor, a condensing step using Gabriel graph rule, and a filtering step of ICF. The authors provide a rather short discussion of results, and do not test the difference in quality of the average results of the algorithms.

For a thorough survey of graph-based methods for nearest neighbor classification, the reader is referred to [32]. There are two main differences between proximity graphs and HMN's. First, HMN's explicitly use the class label of points in the definition of edges. As a consequence, while proximity graphs can be applied to any dataset, HMN's are specifically defined for labeled data. Second, HMN's are directed graphs, while proximity graphs are not. Exact computation of HMN has quadratic time complexity. This bound can be reduced by using metric trees or other spatial data structures [18].

A new class of directed proximity graphs, called class cover catch digraphs (CCCD's) has been introduced in [23], which provide a graph-based representation of one (target) class versus a different (non-target) class. In a CCCD of two such classes, nodes are the target instances and the maximal covering balls centered on each target instance, where a maximal covering ball of a target point is the ball centered in that point with maximum radius, which does not contain any non-target point. Each maximal covering ball is connected to its center by a directed edge.

CCCD's have been used for translating the so-called 'constrained class cover problem' (CCCP) to a problem on directed graphs. The CCCP amounts to find a minimum cardinality set of open covering balls with centers in target class points whose union covers the target class and does not contain any point of the non-target class.

The problem of finding an optimal solution to an instance of the CCCP has been shown to be equivalent to the one of finding a minimum cardinality dominating set in a general

digraph. For CCCD's with points on Euclidean  $L_2$  metric space, the problem can be solved in  $O(n^m)$  time, with  $n$  and  $m$  equal to the number of target and non-target points, respectively. Further information about analysis of CCCD's and their application to classification can be found in [14, 15, 16].

While both HMN's and CCCD are directed graphs, they describe different relations: HMN's describe the nearest neighbor relation between points of each pair of classes, while CCCD's describe the relation between maximal covering balls and target instances of one class.

The rest of the paper is organized as follows. After introducing the terminology used throughout the paper, the next section defines HMN's and discusses their properties. Section 3 presents a brief review of instance selection methods. Section 4 introduces HMN-C, HMN-E and HMN-EI. Section 5 describes experiments. Finally, in Section 6, we conclude and point to future work.

## Terminology

The following notions and terms will be used in the sequel.

- $X$ : a training set,
- $L = 1, \dots, c$ : class labels of  $X$
- $x$ : an element of  $X$ ,
- $|X|$ : the number of elements (cardinality) of  $X$ ,
- $X_i$ : the set of points of  $X$  with label  $i$ ,
- $label(x)$ : the class label of  $x$ ,
- $1\text{-NN}(x, l)$ : the nearest neighbor of  $x$  among those points (different from  $x$ ) with label  $l$ ,
- $G$ : a directed graph with nodes representing elements of  $X$ ,
- $e = (x, y)$ : an edge of  $G$ , with  $x$  the vertex from which  $e$  is directed and  $y$  the vertex to which  $e$  is directed,
- $d(x)$ : the number of edges where  $x$  occurs (the degree of  $x$ ),
- $d(G)$ : the total number of edges of  $G$  (the degree of  $G$ ),
- *in-degree* of  $x$ : the number of edges pointing to  $x$ ,

## 2 Hit Miss Networks

Suppose  $X$  consists of points from  $c$  different classes. In an HMN of  $X$ , a directed edge from point  $x$  to  $y$  is defined if  $y$  is the nearest neighbor of  $x$  in the class of  $y$ . Thus each point  $x$  has  $c$  outgoing edges, one for each class. When the classes of  $x$  and  $y$  are the same, we call  $x$  a *hit* of  $y$ , otherwise a *miss* of  $y$ . The name hit miss network is derived from these terms.

**Definition 2.1 (Hit Miss Network)** *The Hit Miss Network of  $X$ ,  $\text{HMN}(X)$ , is a directed graph  $G = (V, E)$  with*

- $V = X$  and
- $E = \{(x, 1\text{-NN}(x, l)) \text{ for each } x \in X \text{ and } l \in L\}$ .

**Definition 2.2 (Hit, Miss Points)** *Let  $G = \text{HMN}(X)$ . A hit of  $x$  (respectively, miss of  $x$ ) is any point  $y$  such that  $e = (y, x)$  is an edge of  $G$  and  $label(y) = label(x)$  (respectively,  $label(y) \neq label(x)$ ).*

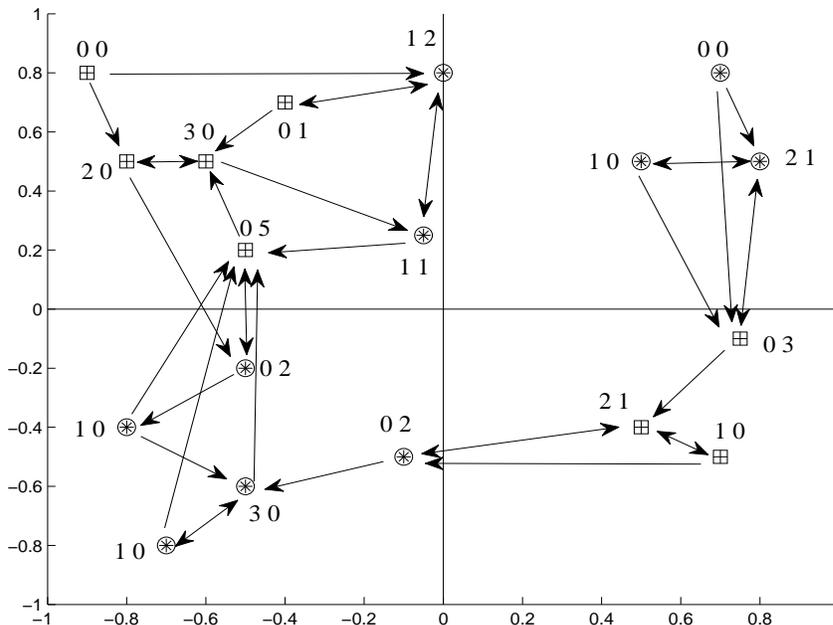


Figure 1: HMN graph of the training set for an artificial classification problem. Hit- and miss-degree of each node is written on the left and right side of the node, respectively.

We call *hit-degree* (respectively *miss-degree*) of  $x$  the number of hit (respectively miss) nodes of  $x$ .  $Hit(x)$  (respectively  $Miss(x)$ ) denotes the set of hit (respectively miss) nodes of  $x$ .

Figure 1 shows the HMN of the training set for a toy binary classification task. Observe that the two points with zero in-degree are relatively isolated and far from points of the opposite class, while points with high miss-degree are closer to points of the opposite class and to the 1-NN decision boundary.

Computing HMN requires quadratic time complexity in the number of points. Nevertheless, by using metric trees or other spatial data structures this bound can be reduced. For instance, using *kd* trees, whose construction takes time proportional to  $n \log(n)$ , nearest neighbor search exhibits approximately  $O(n^{1/2})$  behavior [18]. A recent fast all nearest neighbor algorithm for applications involving large point-clouds is introduced in [29].

By construction, the degree of  $G$ , and the degree  $d(x)$  of a node  $x \in V$  satisfy the following properties:

$$d(G) = c \cdot |X|$$

and

$$c \leq d(x) \leq |X| + c - 1.$$

HMN's describe the nearest neighbor relation of *each pair of classes* of the training set. Formally, it is easy to check that

$$HMN(X) = \cup_{i,j,i \neq j,i,j \in [1,c]} HMN(X_i \cup X_j).$$

Thus HMN’s can be constructed in an incremental way by adding nodes and edges for each pair of classes, implying that if a new class is added, one does not need to recompute the entire HMN.

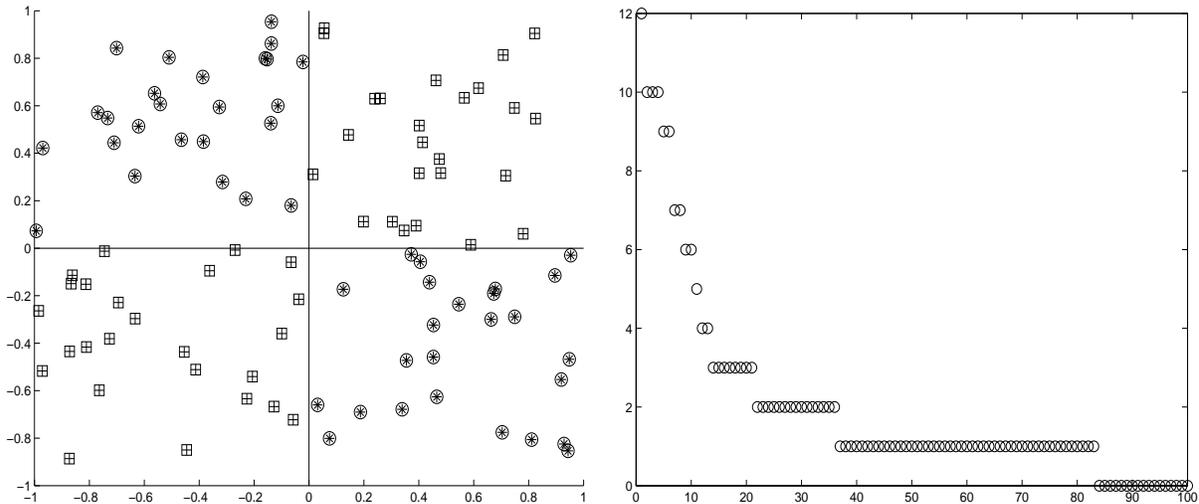


Figure 2: A XOR problem data set (left) and plot of sorted in-degrees (y-axis) of nodes (x-axis), in decreasing order, of the corresponding HMN graph (right).

Figure 2 shows a training set for a XOR classification task, and the sorted in-degrees of its HMN graph. The in-degree distribution follows a Power law, where very few nodes have high in-degree. If we randomly permute the class labels of the training set then the degree distribution changes, with lower in-degree values and more nodes having small in-degree (cf. Figure 3).

These observations indicate that the local structure of HMN provides information about properties of the training points, and motivate us to use HMN’s for defining a new instance selection technique. Before that, in the next section we review briefly instance selection algorithms.

### 3 Instance Selection Algorithms

In instance-based learning, the training set is stored, and the machine learning algorithm computes a distance between the new instance and the stored ones in order to classify new instances. In particular, in the one nearest neighbor algorithm (1-NN) the class label of a new instance is the one of the stored instance with minimum distance.

Instance selection techniques, here also called *editing* techniques, select a subset of the training set in order to improve the storage and possibly the generalization performance of an instance-based learning algorithm. In this paper we focus on the 1-NN classifier.

Research on instance selection started with the seminal work of [19]. Subsequent research focussed mainly on three types of training set condensation techniques [8]: competence preservation, competence enhancement, and hybrid approaches.

- *Competence preservation* algorithms compute a *training set consistent subset* by remov-

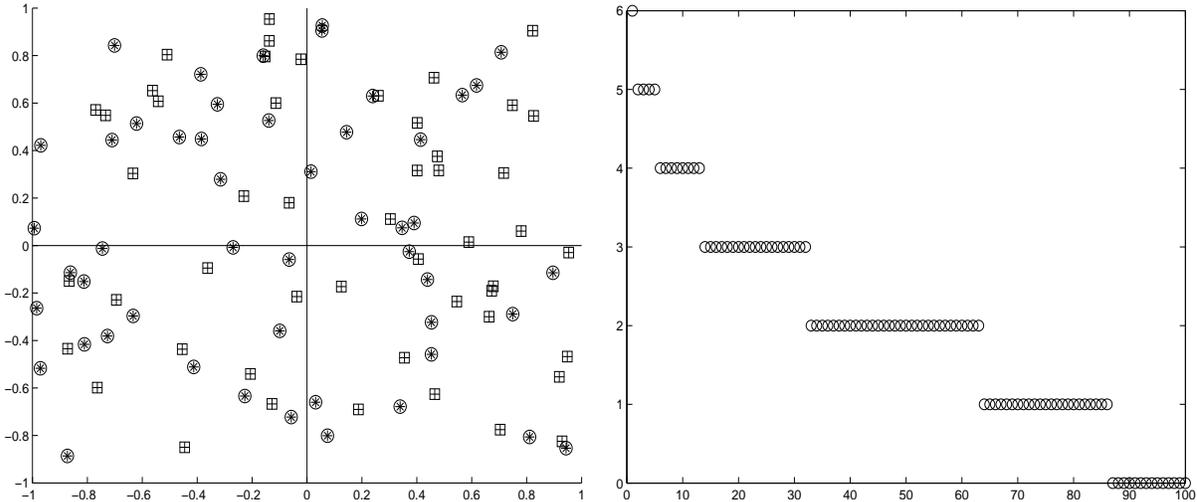


Figure 3: Training points of a XOR problem data set with labels randomly permuted (left figure) and plot of in-degrees, sorted in decreasing order, obtained by applying HMN (right figure).

ing irrelevant points that do not affect the classification accuracy of the training set (e.g., [2, 12]).

- *Competence enhancement* methods remove noisy points in order to increase classifier accuracy. Noise reduction techniques can remove exception instances or border instances which cannot be distinguished from true noise by the technique, hence can possibly affect negatively the generalization performance of the classifier that uses only the selected instances (e.g., [34, 35]).
- *Hybrid* methods aim at finding a subset of the training set that is both noise free and does not contain irrelevant points (e.g., [8, 36]). Alternative methods use prototypes instead of instances of the training set (cf., e.g., [25]).

In [37], Wilson and Martinez present a comprehensive survey of concepts and issues related to reduction techniques for instance-based learning algorithms, including a thorough experimental comparison of algorithms. Other, more recent surveys of instance selection techniques are [8, 20, 21]. In particular, in their paper [8] on advances in instance selection, Brighton and Mellish compare experimentally Edited Nearest Neighbor (**E-NN**) and the state-of-the-art editing algorithms Iterative Case Filtering (**ICF**) and Decremental Reduction Optimization Procedure 3 (**DROP3**). **E-NN** is an algorithm generally considered in comparative experimental analysis of editing methods mainly because it provides useful information on the amount of 'noisy' instances contained in the considered datasets, and on the improvement of accuracy obtained by their removal. Iterative Case Filtering uses **E-NN** as pre-processing noise reduction step, followed by an iterative procedure for deleting 'superfluous points'. Also **DROP3** begins with the application of a simple noise reduction step, followed by another simple type of heuristic for discarding 'superfluous points'.

Results of an extensive comparative experimental analysis performed in [37] and in [8] indicate that **ICF** and **DROP3** are cutting-edge instance selection algorithms, achieving best

K-MN accuracy and storage reduction on a large number of learning tasks over many other editing methods. These algorithms, together with E-NN, are described in more detail below and used to assess comparatively the performance of HMN-based editing algorithms.

### 3.1 Edited Nearest Neighbor

Wilson in [35] introduced the *Edited Nearest Neighbor* (E-NN), where each point  $x$  is removed from  $X$  if it does not agree with the majority of its  $K$  nearest neighbors. This editing rule removes noisy points as well as points close to the decision boundary, yielding to smoother decision boundaries.

### 3.2 Iterative Case Filtering

In [7] Brighton and Mellish proposed the *Iterative Case Filtering* algorithm (ICF), which first applies E-NN algorithm iteratively until it is not possible to remove any point, and next iteratively removes other points as follows. At each iteration, all points for which the so-called *reachability* set is smaller than the *coverage* one are deleted. The reachability of a point  $x$  consists of the points inside the largest hyper-sphere containing only points of the same class as  $x$ . The *coverage* of  $x$  is defined as the set of points that contain  $x$  in their reachability set.

### 3.3 Decremental Reduction Optimization

The family of Decremental Reduction Optimization (DROP) algorithms was first introduced by Wilson and Martinez in [36], and further extended and analyzed in [37]. It consists of five algorithms DROP1-5. DROP1 is the basic removal rule, which removes a point  $x$  from  $X$  if the accuracy of the K-NN rule on the set of its associates does not decrease. Each point has a list of  $K$  nearest neighbors and a list of associates, which are updated each time a point is removed from  $X$ . A point  $y$  is an *associate* of  $x$  if  $x$  belongs to the set of  $K$  nearest neighbors of  $y$ . If  $x$  is removed then the list of  $K$  nearest neighbors of each of its associates  $y$  is updated by adding a new neighbor point  $z$ , and  $y$  is added to the list of associates of  $z$ . Moreover, for each of the  $K$  nearest neighbors  $y$  of  $x$ ,  $x$  is removed from the list of associates of  $y$ .

DROP2 is obtained from DROP1 by discarding the last update step, hence it considers all associates in the entire training set when testing accuracy performance in the removal rule. Moreover, the removal rule is applied to the points sorted in decreasing order of distance from their nearest neighbor from the other classes (nearest enemy). In this way, points furthest from their nearest enemy are selected first.

DROP3 applies a pre-processing step which discards points of  $X$  misclassified by their  $K$  nearest neighbors, and then applies DROP2.

DROP4 uses a stronger pre-processing step which discards points of  $X$  misclassified by their  $K$  nearest neighbors if their removal does not hurt the classification of other instances.

Finally DROP5 modifies DROP2 by considering the reverse order of selection of points, in such a way that instances are considered for removal beginning with instances that are nearest to their nearest enemy.

DROP3 achieves the best mix of storage reduction and generalization accuracy of the DROP methods (cf. [37]). Moreover, results of experiments conducted in [36, 37] show that DROP3 achieves higher accuracy and smaller storage requirements than several other methods, such as CNN [19], SNN [27], E-NN [35], the All k-NN method [31], IB2, IB3 [1], and the Explore

method [9]. Therefore it seems sufficient, in our opinion, to use DROP3 and ICF in order to compare our algorithms with the state-of-the-art ones.

## 4 Instance Selection with Hit Miss Networks

Zero in-degree nodes of  $\text{HMN}(X)$  include relatively isolated points, and points not too close to the decision boundary. This is illustrated in the HMN-C sub-plot of Figure 5, where zero in-degree nodes of the HMN for a XOR data set are highlighted in bold.

Zero in-degree nodes can be safely removed from  $X$  without affecting 1-NN classification of the training points. Formally, we have the following result.

**Proposition 4.1** *Let  $S$  be obtained by removing from  $X$  all points with zero in-degree. Then  $S$  is a decision-boundary consistent subset.*

### Proof

Suppose there exists  $x \in X$  s.t.  $1\text{-NN}(x, X) = y$ ,  $1\text{-NN}(x, S) = y_1$  and  $l(y) \neq l(y_1)$ . Then  $y \neq y_1$  and  $y$  has been removed. So  $y$  has in-degree equal to 0.

From  $1\text{-NN}(x, X) = y$  it follows that  $x$  is in  $\text{Hit}(y)$  or in  $\text{Miss}(y)$ , hence the in-degree of  $y$  is at least 1, which yields a contradiction.

Then  $l(y) \neq l(y_1)$  was false. Hence  $S$  is a training set consistent subset.

We call HMN-C (HMN for training set Consistent instance selection) the algorithm that removes from the training set all instances with zero in-degree.

HMN-C does not remove noisy instances. Therefore, we introduce an instance selection algorithm based on a more aggressive removal strategy, called HMN-E (HMN for Editing). HMN-E uses properties of the hit- and miss-degree of nodes for deciding which points of  $X$  to remove.

Pseudo-code of this algorithm is given in Figure 1. HMN-E is based on four if-then rules, described below.

1. The first rule removes  $x$  if its miss-degree is greater or equal than its hit-degree, that is  $|\text{Miss}(x)| \geq |\text{Hit}(x)|$ . This amounts to discard a point when it is isolated (that is, has zero in-degree), as well as when it has more 'miss' than 'hit' points.

In order to deal with unbalanced data sets, the terms of the inequality are weighted by the fraction of points of the same and other classes, respectively, resulting in rule R1 (lines 3-5 in Figure 1) which removes a point  $x$  from  $X$  if

$$w_{l(x)} * |\text{Miss}(x)| + \epsilon > (1 - w_{l(x)}) * |\text{Hit}(x)|, \quad (1)$$

where  $w_{l(x)} = |\{z \mid l(z) = l(x)\}|/|X|$  and  $\epsilon < 1$  ( $\epsilon = 0.1$  is used in our experiments).

2. On small datasets, application of R1 could remove too many points of one class. Rule R2 (lines 10-12 in Figure 1) handles this case. It checks if the size of a class becomes too small after application of R1. In such a case all points of that class having positive in-degree are added. The threshold used in the rule is set in such a way that the minimum size of a condensed class becomes equal to 4. We consider this to be a reasonable class storage lower bound for the condensed 1-NNrule.

```

1: compute HMN( $X$ )
2: for  $x$  in  $X$  do
3:   if  $w_{l(x)} * |Miss(x)| + \epsilon > (1 - w_{l(x)}) * |Hit(x)|$  then
4:     flag  $x$  for removal {rule R1}
5:   end if
6: end for
7:  $X_{R1,remove} = \{x \in X \text{ with flag for removal}\}$ 
8: for  $l$  in  $1 \dots c$  do
9:    $Left_l = \{x \notin X_{R1,remove} \mid l(x) = l\}$ 
10:  if  $|Left_l| < 4$  then
11:    unflag  $\{z \in X_{R1,remove} \mid l(z) = l, in-degree(z) > 0\}$  {rule R2}
12:  end if
13: end for
14: for  $x$  in  $X_{R1,remove}$  do
15:  if  $c > 3$  and  $|Miss(x)| < c/2$  and  $in-degree(x) > 0$  then
16:    unflag  $x$  {rule R3}
17:  end if
18:  if  $|Hit(x)| \geq |X_{l(x)}|/4$  then
19:    unflag  $x$  {rule R4}
20:  end if
21: end for
22: remove from  $X$  all  $x$  with flag for removal

```

Figure 4: Pseudo-code of HMN-E algorithm. Input: training set  $X$ . Output: subset of  $X$ .

3. Suppose for simplicity each class has equal size ( $|X|/c$ ). Then  $|Miss(x)| \leq \frac{(c-1)^2|X|}{c}$ , and  $|Hit(x)| \leq |X|/c - 1$ . Thus  $|Miss(x)|$  grows linearly with the number  $c$  of classes, while  $|Hit(x)|$  decreases in an inverse linear way. Then  $|Miss(x)|$  has more 'chance' to assume higher values in the presence of many classes. This motivates the introduction of the heuristic Rule R3 (lines 15-17 in Figure 1). For more than three classes, a point  $x$  with in-degree greater than 0 is added if it has a low number of 'miss' points, low with respect to  $c$ . Here we use as threshold half of the total number of classes.
4. Points with many 'hits' are closer to the 'centroid' of the class, hence are considered to be relevant for discriminating the classes, even when they are close to points of other classes. This case is implemented in rule R4 (lines 18-20 in Figure 1) which adds  $x$  if it is the 'hit' of at least 25% of the points of its class.

Rules R2 - R4 can be considered as 'rules of thumb'. The thresholds have been set to values considered reasonable for the general case, and not tuned on each specific dataset. They could possibly be improved by performing parameter tuning or by using domain knowledge on the data distribution of the learning task provided by the user.

In order to remove more "redundant" points, HMN-E can be applied iteratively. We repeat the application of HMN-E until the generalization accuracy of 1-NN on the original training set with the reduced set decreases. The resulting algorithm is called HMN-E *Iterated* (HMN-EI).

These three HMN-based editing algorithms are order independent, that is, their output does not depend on the order in which training points are processed.

#### 4.1 Comparison of the Methods on the XOR Problem

Figure 5 shows application of the considered editing algorithms to the training set of a XOR classification task. Points removed by an algorithm are shown in bold.

Observe that points considered 'noisy' by E-NN are close to the decision boundary. ICF deletes also 'safe' points far from the decision boundary (in order to enhance storage requirements).

Note that by construction points removed by HMN-C are also removed by HMN-E, and points removed by HMN-E are also removed by HMN-EI. Moreover, points removed by E-NN are also removed by ICF.

While HMN-EI selects only points with in-degree 1 and 2, ICF and DROP3 selects also points of higher degree. The figures do not show any other apparent set-theoretic relationship between the subsets of points removed by the methods.

Figure 6 plots the sorted in-degrees of the considered XOR training set, where in-degree of points removed by a method are marked with triangles. As expected, points removed by ICF and not already deleted by E-NN have low in-degree. The majority of points removed by E-NN have high in-degree, showing the tendency of 'noisy' points to have high in-degree. HMN-E removes more points with high in-degree than E-NN, and it selects points with low, but not zero, degree. Finally, HMN-EI removes some more points with low degree.

## 5 Experiments

The following seven algorithms are considered: 1-NN (no instance selection), HMN-C, HMN-E, HMN-EI, E-NN, ICF, and DROP3. In order to assess their comparative performance, we im-

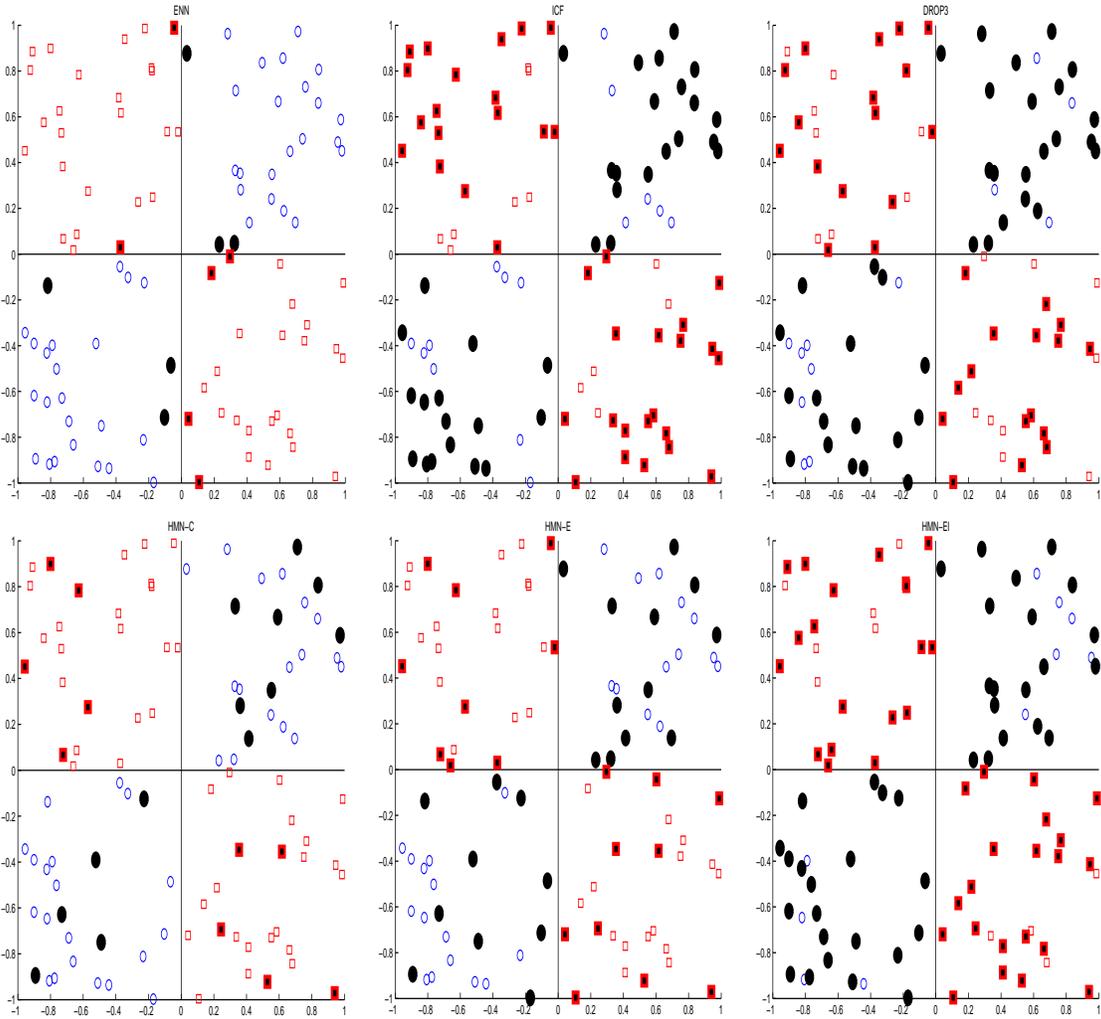


Figure 5: Effect of the algorithms on a XOR problem training set: removed points are shown with filled markers. Top row, from left to right: E-NN, ICF, DROP3. Bottom row, from left to right: HMN-C, HMN-E and HMN-EI.

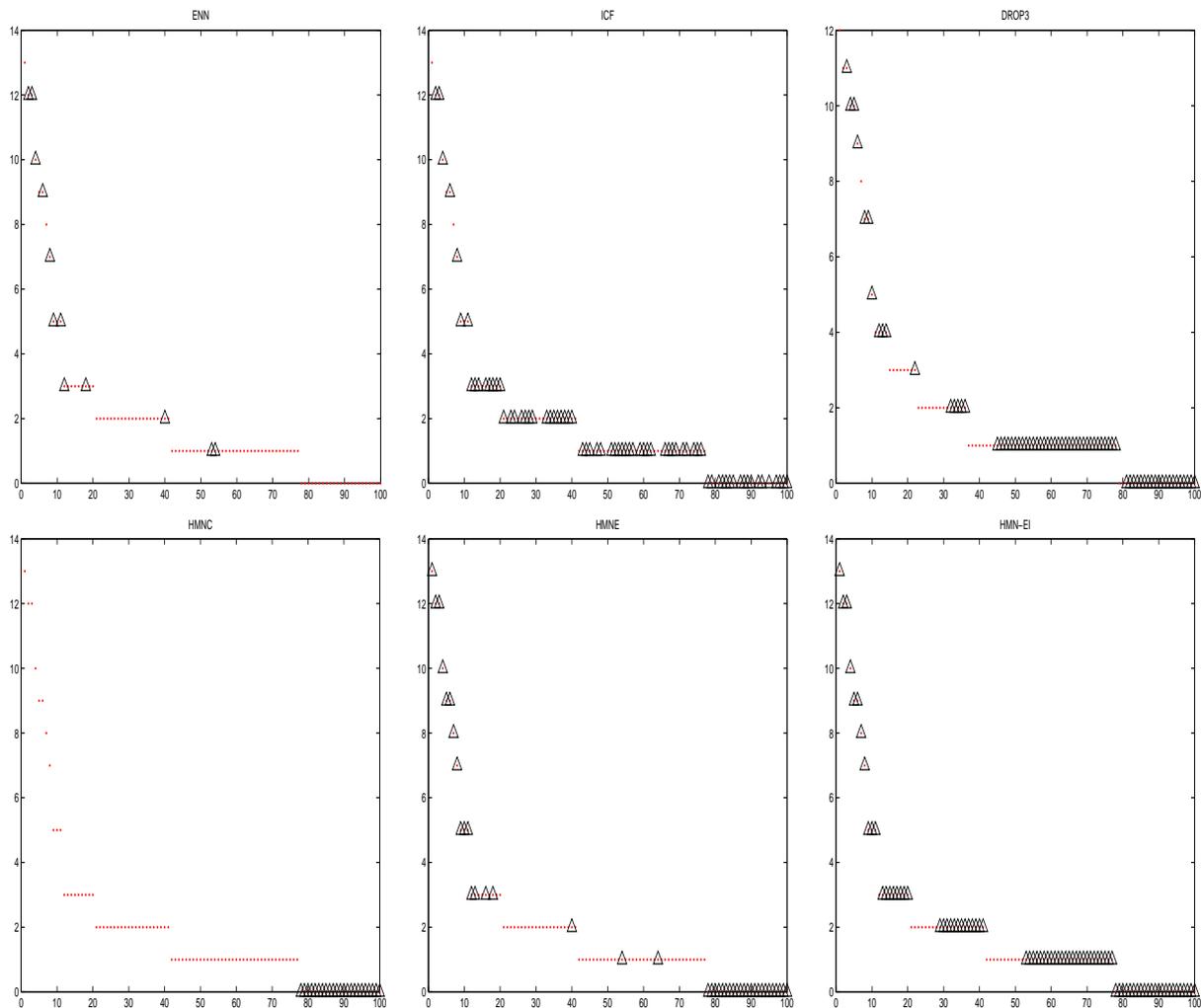


Figure 6: In-degree of nodes of the HMN built on the considered XOR training set, sorted in decreasing order. The in-degree of points removed by an algorithm are marked with triangles. Top row, from left to right: E-NN, ICF, DROP3. Bottom row, from left to right: HMN-C, HMN-E and HMN-EI.

plement the above algorithms and conduct extensive experiments on 22 Machine Learning benchmark datasets. All algorithms are tested using one neighbor.

The performance measures here used are (average) test accuracy of the classifier and (average) percentage of the training set removed by the method.

## 5.1 Datasets

The following 22 publicly available benchmark datasets used in previous studies on model selection for (semi)supervised learning, are considered.

1. Raetsch’s binary classification benchmark datasets have been used in [26]: they consists of 1 artificial and 12 real-life datasets from the UCI, DELVE and STATLOG benchmark repositories.

For each experiment, the 100 (20 for `Splice` and `Image`) partitions of each dataset into training and test set available in the repository are here used.

2. Chapelle’s benchmark datasets used in [10] are from two artificial binary classification and three real-life multi-class classification problems. Specifically, `g50c` and `g10n` are generated from two standard normal multi-variate Gaussians. In `g50c`, the labels correspond to the Gaussians, and the means are located in 50-dimensional space such that the Bayes’ error is 5%. In contrast, `g10n` is a deterministic problem in 10 dimensions, where the decision function traverses the centers of the Gaussians, and depends on only two of the input dimensions.

The three real world datasets are `Coil20`, consisting of gray-scale images of 20 different objects taken from different angles, in steps of 5 degrees, `Uspst`, the test data part of the USPS data on handwritten digit recognition, and `Text` consisting of the classes ‘`mac`’ and ‘`mwindows`’ of the `NewsGroup20` dataset.

For each experiment, the 10 partitions of each dataset into training and test set available in the repository are used.

3. Finally, we consider four standard benchmark datasets from the UCI Machine Learning repository: `Iris`, `Bupa`, `Pima`, and `Breast-W`.

For each experiment, 100 partitions of each dataset into training and test set are used. Each partition randomly divides the dataset into training and test set, equal to 80% and 20% of the data, respectively.

Thus the benchmark data consists of 3 artificial datasets (`Banana`, `g50c`, `g10n`) and 19 real-life ones, with different characteristics as shown in Table 1. In particular, Chapelle’s datasets are balanced, that is, all classes are represented by similar number of points, while some of Raetsch’s datasets are rather unbalanced.

## 5.2 Results

Cross validation is applied to each dataset. For each partition of the dataset, each editing algorithm is applied to the training set  $X$  from which a subset  $S$  is returned. The one nearest neighbor classifier that uses only points of  $S$  is applied to the test set. The average accuracy on the test set over the given partitions is reported for each algorithm (cf. Table 2, Table

Dataset	CL	VA	TR	Cl.Inst.	TE	Cl.Inst.
Banana	2	2	400	212-188	4900	2712-2188
B.Cancer	2	9	200	140-60	77	56-21
Diabetis	2	8	468	300-168	300	200-100
German	2	20	700	478-222	300	222-78
Heart	2	13	170	93-77	100	57-43
Image	2	18	1300	560-740	1010	430-580
Ringnorm	2	20	400	196-204	7000	3540-3460
F.Solar	2	9	666	293-373	400	184-216
Splice	2	60	1000	525-475	2175	1123-1052
Thyroid	2	5	140	97-43	75	53-22
Titanic	2	3	150	104-46	2051	1386-66
Twonorm	2	20	400	186-214	7000	3511-3489
Waveform	2	21	400	279-121	4600	3074-1526
g50	2	50	550	252-248	50	23-27
g10n	2	10	550	245-255	50	29-21
Coil20	20	1024	1440	70	40	2
Text	2	7511	1946	959-937	50	26-24
Uspst	10	256	2007	267-201-169-192-137 -171-169-155-175	50	6-5-9-4-3-3-4-5-5
Iris	3	4	120	40-40-40	30	10-10-10
Bupa	2	6	276	119-157	69	26-43
Pima	2	8	615	398-217	153	102-51
Breast-W	2	9	546	353-193	137	91-46

Table 1: Datasets used in the experiments. Raetsch’s benchmark repository available at <http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm>. Chapelle’s benchmark repository available at <http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/lds/>. Four popular benchmark datasets from UCI Machine Learning repository available at <http://mllearn.ics.uci.edu/MLRepository.html>. D = datasets, CL = number of classes, TR = training set, TE = test set, VA = number of variables, Cl.Inst. = number of instances in each class.

3). The average percentage of instances that are excluded from  $S$  is also reported under the column with label  $R$ . Average and median accuracy and training set reduction percentage for each algorithm over all the 22 datasets is reported near the bottom of the Table.

We compare statistically HMN-EI with each of the other algorithms as follows.

- First a paired t-test on the cross validation results on each dataset is applied, to assess whether the average accuracy for HMN-EI is significantly different than each of the other algorithms. In Tables 2, 3 a '+' indicates that HMN-EI’s average accuracy is significantly higher than the other algorithm at a 0.05 significance level. Similarly, a '-' indicates that HMN-EI’s average accuracy is significantly lower than the other algorithm at a 0.05 significance level. The row labeled 'Sig.acc.+/-' reports the number of times HMN-EI’s average accuracy is significantly better and worse than each of the other algorithms at a 0.05 significance level. A paired t-test is also applied to assess significance of differences

in storage reduction percentages for each experiment.

- Second, in order to assess whether differences in accuracy and storage reduction on all runs of the entire group of datasets are significant, a non-parametric paired test, the Wilcoxon Signed Ranks test<sup>1</sup> is applied to compare HMN-EI with each of the other algorithms. A '+' (respectively '-') in the row labeled 'Wilcoxon' indicates that HMN-EI is significantly better (respectively worse) than the other algorithm.

Dataset	1-NN	HMN-C	R	HMN-E	R	HMN-EI	R
Banana	86.4 +	85.6 +	19.7 +	88.2 +	38.5 +	<b>88.6</b>	57.9
B.Cancer	67.3 +	65.9 +	20.1 +	66.1 +	50.0 +	<b>69.2</b>	72.8
Diabetis	69.9 +	68.6 +	22.4 +	72.5 +	53.1 +	<b>73.5</b>	73.1
German	70.5 +	69.4 +	26.0 +	72.5	56.4 +	<b>72.9</b>	75.5
Heart	76.8 +	76.1 +	23.7 +	<b>81.7</b>	52.9 +	81.6	79.3
Image	<b>96.6</b> -	96.1 -	23.7 +	94.8 -	41.1 +	92.7	57.3
Ringnorm	65.0 +	63.4 +	33.5 +	<b>66.6</b> -	63.9 +	65.6	82.9
F.Solar	60.8 +	60.5 +	80.1 +	63.5 +	86.9 +	<b>64.7</b>	92.1
Splice	71.1 -	70.1 +	46.0 +	<b>72.3</b> -	71.7 +	70.7	86.6
Thyroid	<b>95.6</b> -	94.9 -	24.2 +	93.4	38.9 +	93.2	59.1
Titanic	67.0 +	66.9 +	79.6 +	70.9 +	84.9 +	<b>76.0</b>	94.7
Twonorm	93.3 +	92.8 +	39.4 +	95.7	60.4 +	<b>95.9</b>	83.5
Waveform	84.2 +	83.6 +	36.2 +	<b>86.0</b> -	58.0 +	85.4	79.9
g50c	79.6 +	80.2 +	42.7 +	<b>87.4</b> -	71.0 +	86.8	88.3
g10n	75.0 +	74.6 +	26.0 +	75.8 +	63.5 +	<b>79.2</b>	82.5
Coil20	<b>100</b> -	<b>100</b> -	6.7 +	<b>100</b> -	10.4 +	99.5	15.0
Text	<b>92.8</b> -	90.8 -	16.7 +	89.4 -	54.1 +	86.4	78.9
Uspst	<b>94.6</b> -	<b>94.6</b> -	12.5 +	94.4 -	20.3 +	93.6	29.8
Iris	<b>95.5</b>	95.0	24.7 +	95.1	38.7 +	95.4	75.2
Breast-W	95.7 +	95.5 +	50.7 +	<b>97.1</b>	54.9 +	96.9	71.8
Bupa	61.6 +	59.5 +	18.5 +	63.4 +	54.7 +	<b>64.5</b>	76.0
Pima	67.8 +	66.5 +	21.4 +	70.8 +	50.8 +	<b>71.7</b>	68.1
<b>Average</b>	80.3	79.6	31.6	81.7	53.4	<b>82.2</b>	<b>71.8</b>
<b>Median</b>	78.2	78.1	24.5	<b>83.9</b>	54.4	83.5	<b>75.8</b>
Sig.+/-	15/6	16/5	22/0	8/8	22/0	n/a	n/a
Wilcoxon	+	+	+	~	+	n/a	n/a

Table 2: Results of experiments on ML benchmark datasets. Each column labeled with the name of an algorithm reports its average test set accuracy on each dataset. R = percentage of training points removed. Best results are shown in bold. Average (Median) = average (median) results over datasets. Sig.+/- = number of times HMN-EI average accuracy (storage reduction) is significantly better (+) or significantly worse (-) than the other algorithm, according to a paired t-test at 0.05 significance level. Wilcoxon = a '+' indicates HMN-EI significantly better than the other algorithm at a 0.01 significance level according to a Wilcoxon test for paired samples, ~ indicates no significant difference.

Results of Table 2 show that HMN-EI achieves best generalization accuracy, significantly better than the one of 1-NN and of HMN-C. Moreover, HMN-EI outperforms significantly HMN-E with respect to storage requirements and achieves similar generalization performance.

<sup>1</sup>We used 'wilcoxon' Matlab routine by G. Cardillo.

For these reasons, HMN-EI is chosen for further comparison with state-of-the-art editing algorithms.

### 5.3 Comparison with Other Algorithms

Dataset	HMN-EI	R	ICF	R	E-NN	R	DROP3	R
Banana	<b>88.6</b>	57.9	86.1 +	<b>79.2</b> -	87.8 +	13.1 +	87.6 +	68.2 -
B.Cancer	69.2	72.8	67.0 +	<b>79.0</b> -	69.4	33.3 +	<b>69.7</b> -	72.9
Diabetis	<b>73.5</b>	73.1	69.8 +	<b>83.1</b> -	72.6 +	30.3 +	72.3 +	73.4
German	72.9	75.5	68.6 +	<b>82.2</b> -	<b>73.0</b>	30.1 +	72.0 +	74.3 +
Heart	<b>81.6</b>	79.3	76.7 +	<b>80.9</b> -	80.6 +	23.1 +	80.2 +	72.1 +
Image	92.7	57.3	93.8	<b>80.3</b> -	<b>95.8</b> -	3.4 +	95.1 -	64.9 -
Ringnorm	<b>65.6</b>	82.9	61.2 +	<b>85.5</b> -	54.8 +	35.3 +	54.7 +	80.6 +
F.Solar	<b>64.7</b>	92.1	61.0 +	52.0 +	61.3 +	39.8 +	61.4 +	<b>93.8</b> -
Splice	<b>70.7</b>	<b>86.6</b>	66.3 +	85.5 +	68.4 +	28.3 +	67.6 +	79.01 +
Thyroid	<b>93.2</b>	59.1	91.9 +	<b>85.6</b> -	94.0 -	4.0 +	92.7 +	65.7 -
Titanic	<b>76.0</b>	<b>94.7</b>	67.5	54.3 +	67.3 +	33.0 +	67.7 +	94.3
Twonorm	<b>95.9</b>	83.5	89.2 +	<b>90.7</b> -	94.1 +	6.4 +	94.3 +	72.7 +
Waveform	<b>85.4</b>	79.9	82.1	<b>86.8</b> -	<b>85.4</b>	15.7 +	84.9 +	73.6 +
g50c	<b>86.8</b>	<b>88.3</b>	82.2 +	56.3 +	82.2 +	19.7 +	82.8 +	77.7 +
g10n	<b>79.2</b>	<b>82.5</b>	73.0 +	53.9 +	74.0 +	22.8+	75.0 +	71.4 +
Coil20	99.5	15.0	98.5 +	<b>42.6</b> -	<b>100</b> -	0.0 +	95.5 +	64.4 -
Text	86.4	<b>78.9</b>	88.2 -	68.8 +	<b>91.6</b> -	7.7 +	88.0 -	66.7 +
Uspst	93.6	29.8	86.2	<b>87.8</b> -	<b>94.0</b>	4.7 +	91.4 +	67.3 -
Iris	95.4	<b>75.2</b>	95.3	69.7 -	<b>95.9</b> -	4.2 +	95.8 -	66.4 +
Breast-W	<b>96.9</b>	71.8	95.4 +	<b>93.8</b> -	96.6	4.1 +	96.8	74.2 -
Bupa	<b>64.5</b>	<b>76.0</b>	60.9 +	74.3 +	63.2 +	38.1+	63.1 +	73.8 +
Pima	<b>71.7</b>	68.1	67.9 +	<b>78.7</b> -	69.7 +	32.4 +	69.4 +	73.3 -
<b>Average</b>	<b>82.0</b>	71.8	78.6	<b>75.0</b>	80.5	19.5	79.9	<b>73.7</b>
<b>Median</b>	<b>83.5</b>	75.8	79.4	<b>79.8</b>	81.4	21.25	81.5	73.1
Sig.+/-	n/a	n/a	16/2	7/15	12/5	22/0	17/4	11/8
Wilcoxon	n/a	n/a	+	~	+	+	+	~

Table 3: Results of experiments on ML benchmark datasets of HMN-EI, ICF, Wilson’s editing, and DROP3.

From the results of the experiments reported in Table 3 we derive the following observations.

- On the g50c dataset, HMN-EI achieves highest average accuracy, significantly better than that of all other methods. With an average error of about 13%, close to twice the Bayes probability of error, HMN-EI performs almost optimally, and discards about 88% of the training data. This shows effectiveness and robustness of this method with respect to the presence of noise (on this type of classification task).
- On the g10n dataset, HMN-EI achieves significantly better performance than that of the other methods, indicating robustness to the presence of irrelevant variables (on this type of classification task).
- On datasets with more than three classes, HMN-EI has worse storage requirements than the other algorithms, but also generally higher accuracy, due to the more conservative

editing strategy (Rule 3) HMN-EI uses on datasets with many classes.

- Results of a paired t-test at a 0.05 significance level shows better accuracy performance of HMN-EI over ICF, E-NN and DROP3 (cf. row Sig.+/-) on 15, 12, and 17 of the datasets, and worse accuracy on 2, 5, and 4 datasets, respectively. Storage reduction of HMN-EI is 7, 22, and 11 times better, and 15, 0, and 8 worse, indicating better storage performance of ICF, according to this test. As shown, e.g., in [13], comparison of the performance of two algorithms based on the t-test is only indicative because the assumptions of the test are not satisfied, and the Wilcoxon test is shown to provide more reliable estimates.
- Results of the non parametric Wilcoxon test for paired samples at a 0.01 significance level indicate that the performance of HMN-EI on the entire set of classification tasks is significantly better than each one of the other algorithms with respect to accuracy, and that there is no significant difference in storage reduction between HMN-EI and state-of-the-art editing algorithms (cf. last row of the table).
- The three best performing instance selection algorithms, DROP3, ICF and HMN-EI have quadratic computational complexity in the number of instances (which can be reduced by using ad-hoc data structures such as kd-trees). ICF and HMN-EI are in principle slower than the other algorithms, due to their multiple passes over (selected) instances. Nevertheless, in our experiments these algorithms require a small number of iterations (about 7 for ICF and 3 for HMN-EI). Thus their computational complexity is not significantly worse than that of DROP3.

In summary, results of these experiments indicate effectiveness of HMN-based instance selection and robustness of HMN-EI with respect to the presence of high number of variables, training examples, multiple classes, noise and irrelevant variables. Comparison with results obtained by E-NN, ICF and DROP3 shows improved average accuracy and similar storage requirement of HMN-EI, ICF and DROP3 on these datasets.

## 6 Conclusions and Future Work

This paper proposed a new graph-based representation of a training set and showed how local structural properties of nodes provide information about the closeness of the points to the decision boundary of the 1-NN rule. We formalized these properties by means of the notions of *Hit* and *Miss* set, and used such notions for defining three algorithms for 1-NN's instance selection. We proved that HMN-C removes instances without affecting the accuracy of the 1-NN rule on the original training set (it computes a decision-boundary consistent subset). We showed that HMN-E and HMN-EI remove more points than HMN-C, including those close to the decision boundary. Results of extensive experiments indicated that HMN-EI significantly improves the generalization accuracy of 1-NN and reduces significantly its storage requirements.

We compared experimentally HMN-EI with a popular noise reduction algorithm (E-NN), and two state-of-the-art editing algorithms (ICF and DROP3). Results of extensive experiments on 19 real-life datasets and 3 artificial ones showed that HMN-EI achieved best average accuracy, and storage reduction similar to that of ICF and DROP3. This indicates that simple topological properties of the proposed graph-based dataset representation provide an effective tool for 1-NN's instance selection.

The design of condensing algorithms could also be based on an extension of HMN for describing the  $K$ -nearest neighbor relation between each pair of classes. We conducted preliminary experiments to investigate whether using more than one neighbor to classify new points affects the accuracy performance of the condensing algorithms here considered. Results of experiments on seven UCI ML datasets, using 3 and 5 neighbors for classifying new points, showed that HMN-EI still achieves best average accuracy. In general, the generalization performance increased (of about 1%, 2%) when 3 and 5 neighbors were used.

HMN's could be used for other applications. For instance, for measuring the difficulty of a learning task with respect to a given training set (cf. e.g. [38]), for enhancing classification techniques based on a notion of margin, such as Support Vector Machines (cf. e.g. [30]), for improving Boosting algorithms by means of editing techniques (cf. e.g. [34]), and more generally for tackling over-fitting in supervised learning.

## Acknowledgments

Many thanks to the Reviewers and to the Editor Leon Bottou for their constructive comments.

## References

- [1] D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
- [2] F. Angiulli. Fast nearest neighbor condensation for large data sets classification. *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1450–1464, 2007.
- [3] V. Barnett. The ordering of multivariate data. *J. Roy. Statist. Soc., Ser. A* 139(3):318–355, 1976.
- [4] B. Bhattacharya and D. Kaller. Reference set thinning for the  $k$ -nearest neighbor decision rule. *Proceedings of the 14th International Conference on Pattern Recognition*, (1):238–243, 1998.
- [5] B. Bhattacharya, K. Mukherjee, and G. Toussaint. Geometric decision rules for instance-based learning problems. In *Proceedings of the 1st International Conference on Pattern Recognition and Machine Intelligence (PReMI'05)*, number LNCS 3776, pages 60–69. Springer, 2005.
- [6] B.K. Bhattacharya. Application of computational geometry to pattern recognition problems. *PhD Thesis. Simon Fraser University, School of Computing Science, Technical Report*, TR 82-3, 1982.
- [7] H. Brighton and C. Mellish. On the consistency of information filters for lazy learning algorithms. In *PKDD '99: Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, pages 283–288. Springer-Verlag, 1999.
- [8] H. Brighton and C. Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, (6):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] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 57–64, 2005.
- [11] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.
- [12] B. V. Dasarathy. Minimal consistent set (mcs) identification for optimal nearest neighbor decision systems design. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3):511–517, 1994.
- [13] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
- [14] J. DeVinney and C.E. Priebe. The use of domination number of a random proximity catch digraph for testing spatial patterns of segregation and association. *Statistics and Probability Letters*, 73(1):37–50, 2005.
- [15] J. DeVinney and C.E. Priebe. A new family of proximity graphs: Class cover catch digraphs. *Discrete Applied Mathematics*, 154(14):1975–1982, 2006.
- [16] E.J. Wegman D.J. Marchette and C.E. Priebe. Fast algorithms for classification using class cover catch digraphs. *Handbook of Statistics*, 24:331–358, 2005.
- [17] S.N. Dorogovtsev and J.F.F. Mendes. *Evolution of Networks: From Biological Nets to the Internet and WWW*. Oxford University Press, 2003.
- [18] P.J. Grother, G.T. Candela, and J.L. Blue. Fast implementation of nearest neighbor classifiers. *Pattern Recognition*, 30:459–465, 1997.
- [19] P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.
- [20] N. Jankowski and M. Grochowski. Comparison of instances selection algorithms i. algorithms survey. In *Artificial Intelligence and Soft Computing*, pages 598–603. Springer, 2004.
- [21] N. Jankowski and M. Grochowski. Comparison of instances selection algorithms ii. results and comments. In *Artificial Intelligence and Soft Computing*, pages 580–585. Springer, 2004.
- [22] J.W. Jaromczyk and G.T. Toussaint. Relative neighborhood graphs and their relatives. *P-IEEE*, 80:1502–1517, 1992.
- [23] C.D.J. Marchette, C.E. Priebe, D.A. Socolinsky, and J.G. DeVinney. Classification using class cover catch digraphs. *Journal of classification*, 20(1):3–23, 2003.
- [24] K. Mukherjee. Application of the gabriel graph to instance-based learning. In *M.sc. project, School of Computing Science, Simon Fraser University*, 2004.

- [25] E. Pekalska, R.P. W. Duin, and P. Paclík. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189–208, 2006.
- [26] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.
- [27] 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(6):665–669, 1975.
- [28] J.S. Sánchez, F. Pla, and F.J. Ferri. Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognition Letters*, 18:507–513, 1997.
- [29] J. Sankaranarayanan, H. Samet, and A. Varshney. A fast all nearest neighbor algorithm for applications involving large point-clouds. *Comput. Graph.*, 31(2):157–174, 2007.
- [30] H. Shin and S. Cho. Neighborhood property based pattern selection for support vector machines. *Neural Computation*, (19):816–855, 2007.
- [31] I. Tomek. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(6):448–452, 1976.
- [32] G.T. Toussaint. Proximity graphs for nearest neighbor decision rules: recent progress. In *Interface-2002, 34th Symposium on Computing and Statistics*, pages 83–106, 2002.
- [33] G.T. Toussaint, B.K. Bhattacharya, and R.S. Poulsen. The application of voronoi diagrams to non-parametric decision rules. In *Proceedings of the 16th Symposium on Computer Science and Statistics*, pages 97–108, 1984.
- [34] A. Vezhnevets and O. Barinova. Avoiding boosting overfitting by removing "confusing" samples. In *Proceedings of the 18th European Conference on Machine Learning (ECML)*, volume 4701, pages 430–441. LNCS, 2007.
- [35] D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, (2):408–420, 1972.
- [36] D. Randall Wilson and Tony R. Martinez. Instance pruning techniques. In *Proc. 14th International Conference on Machine Learning*, pages 403–411. Morgan Kaufmann, 1997.
- [37] D.R. Wilson and T.R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.
- [38] D.A. Zighed, S. Lallich, and F. Muhlenbach. Separability index in supervised learning. In *PKDD '02: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 475–487. Springer-Verlag, 2002.