Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Improving kNN multi-label classification in Prototype Selection scenarios using class proposals

Jorge Calvo-Zaragoza*, Jose J. Valero-Mas, Juan R. Rico-Juan

Department of Software and Computing Systems, University of Alicante, Carretera San Vicente del Raspeig s/n, 03690 Alicante, Spain

ARTICLE INFO

Article history:
Received 18 July 2014
Received in revised form
18 November 2014
Accepted 22 November 2014
Available online 3 December 2014

Keywords:
K-Nearest Neighbor
Multi-label classification
Prototype Selection
Class proposals

ABSTRACT

Prototype Selection (PS) algorithms allow a faster Nearest Neighbor classification by keeping only the most profitable prototypes of the training set. In turn, these schemes typically lower the performance accuracy. In this work a new strategy for multi-label classifications tasks is proposed to solve this accuracy drop without the need of using all the training set. For that, given a new instance, the PS algorithm is used as a fast recommender system which retrieves the most likely classes. Then, the actual classification is performed only considering the prototypes from the initial training set belonging to the suggested classes. Results show that this strategy provides a large set of trade-off solutions which fills the gap between PS-based classification efficiency and conventional kNN accuracy. Furthermore, this scheme is not only able to, at best, reach the performance of conventional kNN with barely a third of distances computed, but it does also outperform the latter in noisy scenarios, proving to be a much more robust approach.

1. Introduction

Since its first proposal in 1951 [1], the *k*-Nearest Neighbor rule (kNN) constitutes one of the most well-known algorithms in Pattern Recognition (PR) for supervised non-parametric classification [2], case in which statistical knowledge of the conditional density functions of the classes involved is not available. Most of the kNN popularity in classification tasks comes from its conceptual simplicity and straightforward implementation, which can be described as a distance comparison between elements. More precisely, given an input *x*, the NN (kNN) rule assigns to *x* the label of the nearest (k-nearest) prototypes of the training set. An interesting theoretical property of this rule is that its probability of error is bounded above by twice the Bayes error rate [3]. kNN algorithm is usually described as a *lazy* learner which, in opposition to *eager* learners, does not build a classification model out of the training data until a new element has to be classified. Inside this lazy learning family, kNN is an example of instance-based method, meaning that no classification rules are obtained out of the training data, being part or the total amount of training information itself used for the classification task [4].

Despite the commented kNN popularity in PR, this method suffers from several drawbacks, out of which three clearly limit its application [5]: the first one is that, as an instance-based classifier, storage memory requirements tend to be high for keeping all the training data; the second limitation is its low computational efficiency since, each time new data has to be classified, many distance computations are repeated due to the lack of a model; the third disadvantage is that this method is sensitive to noisy instances, especially for low *k* values.

Prototype Selection (PS) is one of the most common techniques for overcoming the commented drawbacks [6]. This family of methods reduces the size of the initial training set so as to decrease the aforementioned computational cost and sensitiveness to noise by removing both redundant and noisy instances from the initial training set. However, although this process is expected to either maintain or even increase the classification results, in practical situations the accuracy obtained tends to be lower than with the initial set.

In this paper, in order to tackle the commented issue, we propose a strategy which aims to combine the classification accuracy of retaining all the training set with the time efficiency PS methods provide in kNN classification. Our proposal first reduces the training set by using a PS algorithm; on that reduced set, we perform the classification of the new element but, instead of retrieving the most convenient class, a rank of classes is proposed according to their suitability; these proposals are then used for classifying the new element on a filtered version of the initial training data in which only the elements belonging to the previously ranked classes are considered for the classification task. This scheme is expected to provide a profitable way of approaching a multi-label classification scenario as a large quantity of prototypes could be discarded.

* Corresponding author. Tel.: +349 65 903772; fax: +349 65 909326.
E-mail addresses: jcalvo@dlsi.ua.es (J. Calvo-Zaragoza), jjvalero@dlsi.ua.es (J.J. Valero-Mas), JuanRamonRico@ua.es (J.R. Rico-Juan).

http://dx.doi.org/10.1016/j.patcog.2014.11.015
0031-3203/© 2014 Elsevier Ltd. All rights reserved.

The rest of the paper is structured as follows: Section 2 introduces some related proposals to this topic; Section 3 thoroughly develops our proposed approach; Section 4 explains the evaluation methodology proposed; Section 5 shows the results obtained as well as a thorough discussion about them; finally, Section 6 explains the general conclusions obtained from the work and discusses about possible future work.

2. Related work

Among the different stages which comprise the so-called Knowledge Discovery in Databases (KDD), Data Preprocessing (DP) is the set of processes devoted to provide the information to the Data Mining (DM) system in the suitable amount, structure and format. Data Reduction (DR), which constitutes one of these DP possible tasks, aims at obtaining a reduced set of the original data which, if provided to the DM system, would produce the same output as the original data [7].

DR techniques are widely used in kNN classification as a means of overcoming its previously commented drawbacks, being the two most common approaches Prototype Generation (PG) and Prototype Selection (PS) [8]. Both methods focus on reducing the size of the initial training set for lowering the computational requirements and removing noisy instances while keeping, if not increasing, the classification accuracy. The former method creates new artificial data to replace the initial set while the latter one simply selects certain elements from that set. The work presented here focuses on PS techniques, which are less restrictive than PG as they do not require extra knowledge to merge elements from the initial set. However, reader is referred to [9] for a detailed introduction and thorough study of PG techniques. On the other hand, below we introduce the basics of PS methods due to its relevance in the present paper.

As aforementioned, PS methods aim to reduce the size of the initial training set to lower the computational cost and remove noisy instances which might confuse the classifier. Given its importance, many different approaches have been proposed throughout the years to carry out this task. Due to this large range of possible strategies, many different criteria have been posed in order to establish a taxonomy for these methods. However, in this paper we restrict ourselves to a criterion which basically divides them into three different families:

- *Condensing*: The idea followed by these methods is to reduce as much as possible the dataset size by keeping only the closest points to the different class decision boundaries. While accuracy on training set is usually maintained, generalization accuracy is lowered.
- *Editing*: This approach eliminates instances which produce some class overlapping, typical situation of elements located close to the decision boundaries or noisy data. Data reduction rate is lower than in the previous case but generalization accuracy is higher.
- *Hybrid*: These algorithms look for a compromise between the two previous approaches, which means seeking the smallest dataset while improving, or at least maintaining, the generalization accuracy of the former set.

For a thorough explanation regarding taxonomy criteria for PS algorithms, the reader may check [5] in which an extensive introduction to this topic as well as a comprehensive classification taxonomy for the different methods is discussed.

Even though PS methods are expected to keep the same accuracy as with the initial training set, in practice it becomes difficult to fulfill this requirement, reason why much research has been recently devoted to enhance these techniques through data reduction and learning techniques [7]. Some explored lines to improve accuracy results have been the use of ensembles together with PS [10] or hybridizing Feature Selection (FS) schemes with PS using Evolutionary Algorithms (EA) [11,12]. On the other hand, and in order to solve the scalability issue these algorithms show for very large datasets, some common methods have been the use of stratification [13] and distributed approaches [14].

In this paper it is proposed a scheme that tries to overcome the aforementioned drawbacks of PS algorithms in a very different way. Here, PS is used just as a preprocessing stage for selecting the most promising labels, which will be used for the actual classification in the original dataset. It should be noted that this approach does not constrain the development of PS algorithms as its performance, as a second stage process, is highly influenced by the initial PS step. In fact, the better the underlying PS algorithm is used, the better the performance is expected to be achieved with our scheme.

3. Improving prototype selection k-Nearest Neighbor classification

Let T be a training set which consists of pairs $\{(x_i, y_i) | x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}_{i=1}^{|T|}$ drawn from an unknown function $f : \mathcal{X} \to \mathcal{Y}$. Typically, $\mathcal{X}$ is a feature space and $\mathcal{Y}$ is a discrete set of *labels* or *classes*. The main goal in supervised classification is to approximate this function f.

Given an input $x \in \mathcal{X}$, the k-Nearest Neighbor rule hypothesizes about $f(x)$ by choosing the most frequent label within its k nearest prototypes of T based on a dissimilarity function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+ \cup \{0\}$.

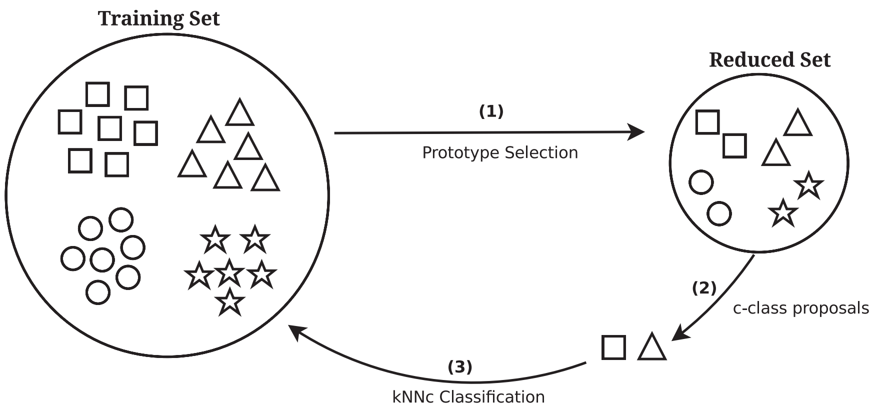

Fig. 1. General scheme of our classification strategy.

Similarly, a PS method takes T and gives a reduced set $R \subseteq T$ following some criteria (see Section 2). Due to the reduction of the original set, the approximation of the function may be different.

Considering the operation of kNN, a misclassification with R that is correctly classified with T has to be produced because of prototypes of the set $T \setminus R$. If we assume that PS is carried out due to time execution, then a profitable procedure is to recover the prototypes of $T \setminus R$ that play a key role in the approximation of $f(x)$. Obviously, finding out which ones of the whole set of prototypes must be reconsidered is not a trivial matter. In this work we propose a strategy that provides a heuristic solution to this situation.

Our classification strategy is based on a three-phase algorithm, which basically consists of the following steps (see Fig. 1 to find an illustration of the process):

1. A given PS algorithm is applied to the whole training set, producing a reduced set. This process is done just once in a preprocessing stage.
2. A new input x is given to the classification system. A reduced set of labels is selected as possible hypotheses for the input x taking into account only the reduced set. Specifically, we propose to select the c (parameter) nearest classes of input x.
3. The final hypothesis is decided using the kNN rule with the part of the initial training set restricted to the c labels proposed in the previous step (kNNc search).

The main idea is to use the reduced set as a fast recommending system, which only has to propose some of the possible labels. After that, the prototypes of those proposed labels are recovered and the final decision is then computed with them, thereby speeding-up the original NN classification.

Let us define $NN(x,k,T)$ as kNN rule for input x and training set T. Let $nearestLabels(c,x,R)$ denote the c-nearest labels of x, defined as a set $\mathcal{C}$ such that

$$\mathcal{C} \equiv \left\{ y \in \mathcal{Y} | \min_{(x',y') \in R : y' = y} d(x,x') < \min_{(x',y') \in R : y' \in \mathcal{Y} \setminus \mathcal{C}} d(x,x') \right\} \text{ s.t} |\mathcal{C}| = c$$

That is, the first c labels that appear if we query the prototypes of the set R in ascendant order to the distance to x.

Let $T_w = \{(x,y) \in T | y = w\}$ be the prototypes of the training set with label w. Then, kNNc search can be performed following Algorithm 1. Note that the algorithm receives the reduced set R since PS can be performed offline, before the test stage.

Algorithm 1. kNNc search.

```
Require: $k, c \in \mathbb{N}; R$
  $\mathcal{C} \leftarrow$ nearestLabels$(c, x, R)$
  $T' \leftarrow \varnothing$
  for all $w \in \mathcal{C}$ do
    $T' \leftarrow T' \cup T_w$
  end for
  $h \leftarrow$ NN$(x, k, T')$
```

Our strategy requires an extra parameter: the scalar value c, which determines how many classes are recommended. This parameter allows tuning the classification since it is expected to affect inversely the accuracy and the computational time. In the experimentation section these two parameters will be analyzed in depth. Additionally, some dissimilarity $d(\cdot,\cdot)$ measure is required over the sample space since it is needed for both the kNN rule and *nearestLabels* function.

4. Experimental setup

This section presents the evaluation methodology for the assessment of the proposed approach, for which the most relevant issues are the classification strategies, the datasets utilized and the performance measurement. These three aspects are described in the following subsections.

4.1. Classification strategies

Our main goal is to compare the performance of our strategy against classical PS-based classification. To this end, we selected a representative set of PS algorithms published in the literature:

- *Condensing Nearest Neighbor* (*CNN*) [15]: Obtains a subset S out of the training set T such that every member of T is closer to a member of S of the same class than to a member of a different class. Prototypes of T are consulted randomly so different computations may give a different subset S.
- *Editing Nearest Neighbor* (*ED*) [16]: Selects a set S that starts equal to the original training set T. Each element of S which does not agree with its neighborhood is removed. As it happens with CNN, its result depends on the order the prototypes are consulted. A common extension to this technique is Multi-Editing (MED) [17], which computes repeatedly the ED algorithm until no more prototypes are removed.
- *Multi-Edit Condensing Nearest Neighbor* (*MCNN*) [18]: Applies ED algorithm and then applies CNN. The process is repeated until convergence is achieved.
- *Fast Condensing Nearest Neighbor* (*FCNN*) [19]: Computes a fast, order-independent condensing strategy based on seeking the centroids of each label. We also add a Multi-Edit Fast Condensing Nearest Neighbor (MFCNN) technique which combines the ideas of MCNN and FCNN.
- *Farther Neighbor* (*FN*) *and Nearest to Enemy* (*NE*) *rank methods* [20]: Give a probability mass value to each prototype following a voting heuristic. Then, prototypes are selected according to a parameter specified by the user that indicates the probability mass desired for each class in the reduced set.
- *Decremental Reduction Optimization Procedure 3* (*DROP3*) [21]: This algorithm applies an initial noise filtering step so as to eliminate the dependency on the order of presentation of the instances; after that, these instances are ordered according to the distance to their nearest neighbors and then, starting from the furthest ones, instances which do not affect the generalization accuracy are removed.
- *Iterative Case Filtering Algorithm* (*ICF*) [22]: Approach which bases its performance on the coverage and reachability premises to select the instances subset able to maximize the prototypes classification accuracy following the NN rule.
- *Cross-generational elitist selection, Heterogeneous recombination and Cataclysmic mutation algorithm (CHC)* [23]: Evolutionary algorithm commonly used as a representative of Genetic Algorithms in PS. The configuration of this algorithm has been the same as in [24], that is α=0.5, Population=50 and Evaluations=10,000.

All these algorithms will be confronted experimentally in order to measure its performance as PS base strategy of a kNNc search compared to the results obtained with the retrieval step proposed. Several values of k (1, 3, 5 and 7) and c (2 and 3) will be analyzed. Furthermore, kNN rule with the whole training set (no previous PS performed) will also be included.

4.2. Datasets

Our experiments are carried out with two different isolated character datasets: the NIST SPECIAL DATABASE 3 (NIST3) of the National Institute of Standards and Technology, from which a subset of the upper case characters was randomly selected (26 classes, 6500 images); and The United States Postal Office (USPS) handwritten digit dataset [25] (9298 images). In both cases, contour descriptions with Freeman Chain Codes [26] are extracted and the edit distance [27] is used as dissimilarity measure. Additionally, we include experiments with the Handwritten Online Musical Symbol (HOMUS) dataset [28]. This dataset is specially interesting for our work because it contains 15,200 prototypes of 32 different classes. Due to its good results in the baseline experimentation with this data, we will use Dynamic Time Warping [29] as dissimilarity measure.

We focused on datasets with many class labels since we consider that the main idea of kNNc is expected to provide interesting results in such data.

4.3. Performance measurement

In order to analyze the impact of our strategy in the PS-based classification, we take into account the following metrics of interest: accuracy of the strategy, the number of distances computed during the classification and the time in milliseconds. The two latter figures provide theoretical and empirical efficiency measures, respectively. Additionally, we provide an accuracy upper bound for kNNc classification measured as the percentage of times for which the correct label is within the c-classes' proposals. Another interesting property of PS-classification is the tolerance to noise. In order to analyze this metric, we will add synthetic noise to our data by swapping the labels of pairs of prototypes randomly chosen. The noise rates (percentage of prototypes that change their label) considered are 0.1, 0.2, 0.3 and 0.4 since these are the common values in this kind of experimentation [30].

Given some PS algorithms, the previous metrics are measured for both values considered for c as well as for PS-based classification without the c-classes retrieval step (except for the upper bound).

These measures allow us to analyze the performance of each considered strategy. Nevertheless, no comparison between the whole set of alternatives can be established so that we can determine which is the best one. The problem is that PS algorithms try to minimize the number of prototypes considered in the training set at the same time they try to increase classification accuracy. Most often, these two goals are contradictory so improving one of them implies a deterioration of the other. From this point of view, PS-based classification can be seen as a Multi-objective Optimization Problem (MOP) in which two functions want to be optimized at the same time: minimization of prototypes in the training set and maximization of the classification success rate. The usual way of evaluating this kind of problem is by means of *non-dominance* concept. One solution is said to dominate another if, and only if, it is better or equal in each goal function and, at least, strictly better in one of them. Therefore, the best solutions (there may be more than one) are those that are non-dominated.

Thus, the considered strategies will be compared by assuming a MOP scenario in which each of them is a 2-dimensional solution defined as (acc,$dist$) where acc is the accuracy obtained by the strategy and $dist$ is the number of computed distances during its classification process. To analyze the results, the pair obtained by each scheme will be plotted in *2D* point graphs where the non-dominated set of pairs will be enhanced. In the MOP framework, the strategies within this set can be considered the best without defining any order among them.

5. Results

This section shows the results obtained using the approach presented in Section 3 with the experimentation described previously.

In sight of the large amount of experimentation carried out because of the number of possible combinations of schemes, noise scenarios and datasets considered, it is unpractical to present all the obtained results due to space limitations. Thus, figures presented actually constitute the average values of the three considered evaluation datasets.

For the sake of clarity, we are showing the obtained results in two different sections: a first one in which the considered datasets are evaluated in their current form and a second one in which the same evaluation is carried out with synthetic noise added to the data.

5.1. Non-added noise scenario

Results presented in this first subsection are obtained without adding artificial noise to the datasets. They can be checked in Table 1.

An initial remark to begin with is that, as no information is discarded, conventional kNN achieves the highest accuracy for all k values when only considering PS. However, the amount of distances computed is the maximum among all the algorithms.

ED and MED algorithms do not significantly reduce the size of the set, maintaining the accuracy in relation to the scores achieved by the kNN implementations. Due to this fact, the introduction of the kNNc approach does not produce a remarkable improvement over the simple PS implementation: accuracy is slightly increased as well as the amount of distances to be computed.

On the other hand, CNN and its extensions exhibit an interesting behavior: all of them achieve a great reduction rate, especially MCNN and MFCNN, as well as a great performance in terms of accuracy (for instance, the latter performs roughly 10% of the distances kNN does but obtaining only 4% less in terms of accuracy). On top of that, the introduction of kNNc does improve results in this case. Let us take the 3NN3 with CNN case: although the number of calculated distances is increased with respect to the PS classification, the accuracy is improved to the point of reaching performance of kNN with barely a third of distances to be computed.

EN and FN methods obtain some of the highest reduction rates (roughly ranging from 1% to 13% of the distances computed by kNN), also depending on its parameterization (the probability mass selected), though accuracy figures are noticeably affected: results get to achieve 15% points less in terms of accuracy with respect to the best result. As in the previous case, the inclusion of kNNc seems to come with some overall upturn: setting $c=3$, the accuracy is improved, in the best case scenario, to just 1% lower than the best score, despite being the number of distances to be computed around 29% of the maximum.

Hybrid algorithms DROP3 and ICF achieve great reduction rates as well (around 6–14% of the total of distances with respect to kNN), but they also experiment a significant decrease in their accuracies, with figures of about 10% and 20% lower than the maximum score. However, when using the proposed approach, there is a remarkable improvement: for instance, in the 3NN3 case, DROP3 increases its accuracy in a 10%, a result roughly 1% lower than the kNN one, computing just a fourth of the maximum number of distances.

The CHC evolutionary algorithm, just as the EN and FN methods when set to 0.1, performs one of the highest reduction rates as depicted in the 1NN case, in which the number of distances is reduced to just the 2% of the maximum, obtaining an accuracy close to 82% of the total. As in the other selection algorithms, when applying the kNNc method to CHC there is a general accuracy

Table 1
Average results obtained when no noise is added to the datasets. Bold elements correspond to the non-dominated points. Normalized results (%) of the different algorithms are obtained referring to the ALL method with the same *k* value.

k	Algorithm	Red. set size	Accuracy (%)			Upper Bound (%)		Distances (%)			Time (%)		
			PS	kNN2	kNN3	kNN2	kNN3	PS	kNN2	kNN3	PS	kNN2	kNN3
1	ALL	6898.7	90.6	90.6	90.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	ED	6231.2	89.4	90.2	90.4	95.5	97.4	90.6	91.6	92.0	90.7	103.4	107.7
	MED	6231.7	89.4	90.2	90.4	95.5	97.4	90.5	91.5	92.1	91.6	103.4	109.9
	MCNN	849.0	**86.9**	89.9	90.2	94.6	96.8	**11.8**	22.4	26.6	12.8	25.6	30.8
	CNN	1589.3	87.0	90.2	90.5	95.9	98.0	22.7	31.5	36.1	24.1	36.0	42.8
	MFCNN	830.9	86.8	90.0	90.4	94.9	97.0	11.7	21.9	26.9	12.1	24.8	29.4
	FCNN	1537.9	87.0	90.2	90.5	95.8	98.0	22.0	30.8	35.3	22.9	35.2	40.9
	1-FN$_{0.10}$	233.8	79.6	86.1	88.1	89.9	93.8	3.5	14.4	19.9	3.1	14.5	20.5
	1-FN$_{0.20}$	539.0	84.1	88.2	89.3	92.6	95.4	7.9	18.4	23.7	7.4	19.6	24.9
	1-FN$_{0.30}$	928.8	85.9	89.1	89.8	93.8	96.3	13.7	23.6	28.6	13.1	25.5	30.6
	1-NE$_{0.10}$	106.6	**75.3**	83.4	85.7	87.1	91.1	**1.6**	12.8	18.4	1.4	13.3	19.3
	1-NE$_{0.20}$	271.5	81.4	87.0	88.2	91.2	94.2	4.1	15.0	20.4	3.6	15.6	21.1
	1-NE$_{0.30}$	512.9	**84.9**	88.4	89.3	93.0	95.8	**7.7**	18.2	23.4	7.0	18.7	24.7
	2-FN$_{0.10}$	228.2	79.1	86.0	87.9	89.9	93.5	3.4	14.4	19.9	3.1	14.4	21.1
	2-FN$_{0.20}$	522.8	83.4	88.1	89.2	92.6	95.5	7.8	18.3	23.6	7.3	19.7	24.3
	2-FN$_{0.30}$	896.7	85.5	89.0	89.8	93.7	96.4	13.3	23.3	28.2	12.8	25.2	30.3
	2-NE$_{0.10}$	102.5	74.2	82.9	85.3	86.6	90.8	1.6	12.8	18.3	1.3	13.3	19.5
	2-NE$_{0.20}$	255.4	80.7	86.6	88.1	90.8	94.1	3.9	14.8	20.3	3.4	15.2	21.7
	2-NE$_{0.30}$	480.0	84.4	88.3	89.3	93.0	95.7	7.3	17.8	23.1	6.7	18.9	25.4
	DROP3	759.8	81.6	88.1	89.3	92.8	95.9	10.5	20.8	26.0	9.3	19.0	23.7
	ICF	987.3	71.8	81.8	85.0	85.9	91.0	14.2	24.2	29.2	14.1	23.7	27.9
	CHC	158.1	**81.7**	86.9	88.6	90.5	94.1	**2.3**	13.4	19.0	2.0	11.5	16.1
3	ALL	6898.7	90.9	90.9	90.9	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	ED	6232.0	89.4	90.4	90.6	95.4	97.4	90.6	91.6	92.1	93.8	106.7	111.7
	MED	6231.9	89.4	90.4	90.6	95.5	97.4	90.5	91.5	92.1	93.6	106.2	111.6
	MCNN	813.3	**85.8**	**89.8**	90.2	94.2	96.6	**10.9**	**21.2**	26.1	11.8	24.1	30.1
	CNN	1611.5	87.1	90.5	**90.9**	96.0	98.1	23.0	31.9	**36.1**	25.4	38.8	43.6
	MFCNN	831.3	86.8	**90.3**	**90.6**	94.9	97.1	11.7	**21.9**	**26.9**	12.7	25.5	30.9
	FCNN	1537.7	87.0	90.3	**90.8**	95.8	98.0	22.0	30.9	**35.3**	23.3	35.6	41.8
	1-FN$_{0.10}$	233.9	79.7	86.4	88.3	90.0	93.7	3.5	14.4	19.9	3.2	15.3	21.1
	1-FN$_{0.20}$	540.0	83.8	88.3	89.5	92.5	95.4	8.0	18.4	23.7	7.6	19.9	25.7
	1-FN$_{0.30}$	930.8	85.8	89.2	90.0	93.7	96.3	13.7	23.6	28.6	13.5	26.0	31.6
	1-NE$_{0.10}$	106.7	75.2	83.4	85.7	86.9	91.0	1.6	12.8	18.4	1.4	14.0	19.7
	1-NE$_{0.20}$	270.8	81.4	**87.1**	88.3	91.3	94.2	4.1	**15.0**	20.4	3.7	15.9	22.5
	1-NE$_{0.30}$	512.5	84.9	**88.7**	89.6	93.1	95.8	7.7	**18.2**	23.4	7.3	19.7	25.9
	2-FN$_{0.10}$	227.9	79.5	86.2	88.1	89.9	93.5	3.4	14.4	19.9	3.1	15.1	21.0
	2-FN$_{0.20}$	522.5	83.4	88.2	89.5	92.6	95.5	7.8	18.3	23.6	7.4	19.2	25.2
	2-FN$_{0.30}$	896.5	85.4	89.1	90.0	93.7	96.3	13.3	23.3	28.2	12.9	25.0	31.4
	2-NE$_{0.10}$	102.4	74.3	83.2	85.5	86.7	90.8	1.6	12.8	18.3	1.3	13.1	19.4
	2-NE$_{0.20}$	255.2	80.7	86.7	88.2	90.8	94.1	3.9	14.8	20.3	3.5	16.0	22.4
	2-NE$_{0.30}$	479.2	84.4	88.6	**89.6**	93.0	95.8	7.3	17.8	**23.1**	6.6	19.0	25.3
	DROP3	513.1	78.3	86.2	88.3	90.2	94.2	7.0	17.6	23.0	6.4	16.3	21.0
	ICF	917.4	73.0	82.3	85.2	86.0	90.7	13.4	23.5	28.5	13.4	22.6	27.9
	CHC	236.3	81.2	86.8	88.4	90.5	93.9	3.6	14.6	20.1	3.2	12.7	17.6
5	ALL	6898.7	90.7	90.7	90.7	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	ED	6231.2	89.4	90.1	90.5	95.5	97.4	90.6	91.5	92.1	90.6	103.2	107.9
	MED	6231.4	89.4	90.1	90.5	95.5	97.4	90.5	91.5	92.1	91.4	102.6	110.3
	MCNN	799.0	85.5	**89.2**	90.0	93.9	96.5	10.6	**20.9**	25.9	11.3	23.3	29.5
	CNN	1599.1	87.1	90.2	90.7	96.0	98.1	22.8	31.7	36.0	24.0	36.4	42.0
	MFCNN	832.9	86.8	89.8	90.4	94.9	97.0	11.7	21.9	26.9	12.3	24.2	30.6

	FCNN	1538.1	87.1	90.0	90.6	95.8	98.0	22.0	30.9	35.3	23.0	35.4	40.7
	1-FN$_{0.10}$	233.5	80.1	86.3	88.2	90.1	93.7	3.5	14.4	19.9	3.2	15.2	21.2
	1-FN$_{0.20}$	539.9	84.0	88.2	89.4	92.6	95.4	8.0	18.5	23.7	7.6	19.7	25.7
	1-FN$_{0.30}$	929.8	85.9	89.0	89.8	93.8	96.2	13.7	23.6	28.6	13.1	25.2	30.8
	1-NE$_{0.10}$	106.5	75.2	83.3	85.7	86.9	91.1	1.6	12.8	18.4	1.3	13.2	19.0
	1-NE$_{0.20}$	271.0	81.5	86.9	88.2	91.2	94.3	4.1	15.0	20.4	3.7	16.1	21.8
	1-NE$_{0.30}$	512.8	84.9	88.5	89.5	93.0	95.8	7.7	18.2	23.4	7.0	19.1	25.0
	2-FN$_{0.10}$	227.7	79.5	86.1	88.0	90.0	93.5	3.4	14.4	19.9	3.0	14.7	20.6
	2-FN$_{0.20}$	523.1	83.4	88.1	89.3	92.6	95.5	7.8	18.3	23.6	7.4	19.2	25.2
	2-FN$_{0.30}$	897.4	85.4	88.9	89.8	93.7	96.4	13.3	23.3	28.2	12.6	24.5	31.1
	2-NE$_{0.10}$	102.5	74.2	83.0	85.3	86.7	90.7	1.6	12.8	18.3	1.3	13.1	19.4
	2-NE$_{0.20}$	255.5	80.9	86.6	88.1	90.9	94.1	3.9	14.8	20.3	3.4	15.7	20.8
	2-NE$_{0.30}$	480.0	84.4	88.3	89.4	93.0	95.8	7.3	17.8	23.1	6.4	18.5	24.0
	DROP3	466.4	77.6	85.4	87.7	89.3	93.6	6.3	17.1	22.5	5.7	15.3	20.0
	ICF	898.7	73.2	82.3	85.1	86.1	90.8	13.2	23.3	28.3	13.1	22.4	26.9
	CHC	265.2	80.9	87.0	88.6	90.8	94.3	4.1	15.1	20.6	3.6	13.2	17.8
7	ALL	6898.7	90.3	90.3	90.3	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	ED	6227.0	89.4	89.7	90.1	95.5	97.4	90.5	91.6	92.0	92.0	103.9	110.4
	MED	6231.6	89.4	89.7	90.1	95.5	97.4	90.5	91.5	92.1	92.2	103.1	111.8
	MCNN	796.6	85.2	88.9	89.6	93.8	96.5	10.6	20.8	25.9	11.2	23.2	29.2
	CNN	1608.8	87.2	89.9	90.2	95.9	98.0	22.9	31.7	36.2	24.1	36.5	41.8
	MFCNN	831.0	86.8	89.6	90.0	94.9	97.1	11.7	21.9	26.9	12.3	24.9	30.5
	FCNN	1539.2	87.1	89.8	90.2	95.8	98.0	22.0	30.9	35.3	23.2	35.4	41.5
	1-FN$_{0.10}$	233.8	79.7	85.9	88.0	89.9	93.9	3.5	14.4	19.9	3.2	16.0	21.1
	1-FN$_{0.20}$	539.5	84.0	88.0	89.0	92.6	95.4	8.0	18.5	23.7	7.6	19.7	25.7
	1-FN$_{0.30}$	929.8	85.9	88.7	89.5	93.8	96.3	13.7	23.6	28.6	13.5	26.3	31.3
	1-NE$_{0.10}$	106.7	75.0	83.2	85.4	86.9	91.1	1.6	12.8	18.4	1.4	13.4	19.9
	1-NE$_{0.20}$	271.4	81.4	86.8	88.0	91.3	94.3	4.1	15.0	20.4	3.7	15.8	21.8
	1-NE$_{0.30}$	513.3	84.9	88.3	89.2	93.0	95.8	7.7	18.2	23.4	7.1	19.2	25.3
	2-FN$_{0.10}$	228.0	79.0	85.9	87.7	89.9	93.4	3.4	14.4	19.9	3.1	14.9	20.8
	2-FN$_{0.20}$	522.2	83.4	87.9	89.0	92.6	95.5	7.8	18.3	23.6	7.5	19.3	26.1
	2-FN$_{0.30}$	895.5	85.5	88.7	89.6	93.8	96.4	13.3	23.3	28.2	12.5	24.3	30.5
	2-NE$_{0.10}$	102.2	74.4	82.7	85.1	86.6	90.8	1.6	12.8	18.3	1.3	13.2	19.5
	2-NE$_{0.20}$	255.0	80.8	86.5	87.8	90.9	94.1	3.9	14.8	20.3	3.4	15.6	21.6
	2-NE$_{0.30}$	479.2	84.4	88.1	89.0	93.0	95.8	7.3	17.8	23.1	6.6	18.4	25.3
	DROP3	478.9	79.2	86.5	88.4	90.7	94.7	6.6	17.3	22.6	5.9	15.7	20.5
	ICF	887.9	73.6	82.2	84.8	86.0	90.6	13.0	23.1	28.2	13.1	22.4	26.8
	CHC	293.2	79.9	86.7	88.2	90.6	94.2	4.5	15.4	20.9	4.0	13.6	18.3

improvement of about 6% and 8% for $c=2$ and $c=3$ respectively, together with a 10% and 15% increase in the number of distances for the same cases.

On average, the accuracy improvement is more significant when passing from the basic PS scheme to the kNNc one than the gain obtained by increasing the number of proposals c, contrasting with the noticeable accuracy rise in the upper bounds in the same situation. This fact clearly points out that the major issue remains at the classification stage since, although the kNNc step is able to give highly accurate recommendations, the overall performance is not capable of reaching these upper limits.

The upper bound ratio does improve as the c value increases since a larger number c of classes are recommended. An increase in this c parameter causes a fixed rise in the number of distances to be computed since the classes in the datasets proposed are balanced. However, as it can be checked in the results, there is not such a linear relation between the upper bound figure and the number of distances computed: for instance, in MFCNN with $k=5$, the upper bounds are 94.9% for $c=2$ and 97% for $c=3$ with 21.9% and 26.9% of distances respectively, depicting that this 2% improvement is around a 5% increase in terms of computational cost but in order achieve a 100% upper bound (the remaining 3%), almost an additional figure of 73% of distances has to be computed. This non-linear behavior, which can be checked in all the other configurations as well, shows a clear dependency with the PS strategy used: a certain PS algorithm with an outstanding performance would require an elevated number of distances to show an improvement whereas an algorithm with a poor performance might exhibit a remarkable accuracy upturn without such distances increase. As a consequence, as the commented upper bounds are the ones which depict the maximum theoretical classification figures which can be expected, the obtained accuracy does also show this non-linearity with respect to the number of distances.

Finally, the increase in the k value does not have any noticeable effect on the accuracy obtained by each algorithm, possibly due to the fact that the datasets are hardly noisy.

As aforementioned, the PS-based classification can be seen as a MOP problem in which accuracy and distances computed have to be simultaneously optimized despite being typically opposed goals. Results of the strategies considered are shown in Fig. 2 facing these two metrics. Optimal solutions, defined using the non-dominance criterion described in Section 4, are remarked in this figure as well as being highlighted in Table 1. Since most of the algorithms gather in a small area, this particular region has been widened for a better visualization.

A first interesting outcome withdrawn from applying this criterion is that the kNN algorithm (with no PS) does not belong to the optimal set of solutions since kNN3 CNN scheme achieves the same accuracy with a lower number of distances computed.

Moreover, it can be also observed that, except for editing approaches, each main scheme – PS, kNN2 and kNN3, drawn in red, green and blue points respectively – entails a cloud of points in different regions of the space. Therefore, kNNc scheme is providing a great range of new options in the trade-off between distances and accuracy not explored in previous works. Furthermore, many kNN2 and kNN3 strategies are found within the optimal set of solutions. Therefore, the user is provided with a wide range of options from which to choose depending on the metric to emphasize (distances or accuracy). For example, let us assume a scenario like that depicted in Fig. 2 in which we are restricted to perform at maximum 25% of the distances. Thanks to 3NN2 scheme with MFCNN prototype selection, we could achieve an accuracy of 90.3% (just 0.6% below the best accuracy) with around 22% of distances computed.

5.2. Noisy scenario

In this subsection the figures obtained when synthetic noise is added are presented. Experimentation was carried out for each of the noise configurations considered in Section 4. As results show a qualitatively similar trend along these noise possibilities, remarks will not focus on a particular configuration but on the general behavior. In addition, and due to space limitations, results of only two of the noise scenarios tackled are shown: an intermediate situation (20% noise rate scenario), for which results can be verified in Table 2, and one for the most adverse situation considered (40% of synthetic noise rate), whose results can be checked in Table 3.

Synthetic noise addition to the samples changes the previous situation drastically. kNN, which scored the best results for each single k value, now exhibits a remarkable decrease in accuracy, becoming more noticeable as the noise rate is increased. As expected, the use of different k values does improve the accuracy, especially in the case of $k=7$, in which kNN scores the maximum classification rate, drawing in these terms with ED.

ED and MED algorithms are able to manage this noisy situation: the size reduction is sharper than in the previous case since the elements removed are the ones leading to confusion. As it happened in the non-added noise scenario, the use of kNNc with these algorithms does not carry a remarkable improvement in accuracy, though it does in the classification upper bounds, getting even to the point of decreasing the performance when low k values are used.

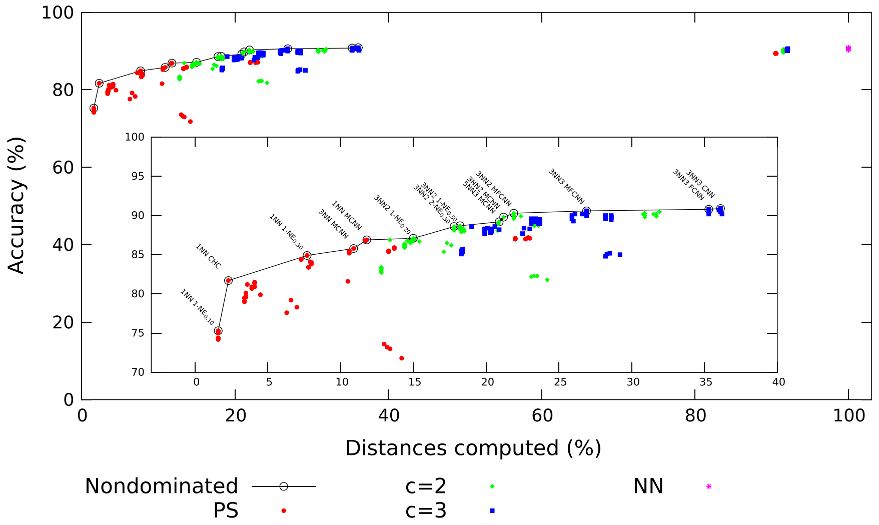

Fig. 2. Distances computed (%) against accuracy achieved by the algorithms. Average results when no noise is added to the samples. The non-dominated front is remarked.

Table 2
Average results obtained when 20% of noise is added to the datasets. Bold elements correspond to the non-dominated points. Normalized results (%) of the different algorithms are obtained referring to the ALL method with the same *k* value.

k	Algorithm	Red. set size	Accuracy (%)			Upper Bound (%)		Distances (%)			Time (%)		
			PS	kNN2	kNN3	kNN2	kNN3	PS	kNN2	kNN3	PS	kNN2	kNN3
1	ALL	6898.7	73.1	73.1	73.1	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	ED	4329.4	88.1	88.3	87.8	94.6	96.8	62.8	66.9	69.0	63.0	76.1	80.1
	MED	4338.1	88.1	88.1	87.8	94.6	96.8	63.0	67.0	69.1	63.7	76.5	81.3
	MCNN	713.6	84.2	87.7	87.5	93.8	96.1	10.0	20.6	25.0	10.2	22.7	27.9
	CNN	4094.3	64.4	70.9	72.4	87.4	94.6	59.2	63.8	66.4	60.6	73.0	79.3
	MFCNN	673.5	**84.6**	87.6	87.6	93.6	96.2	**9.6**	19.8	25.0	9.8	22.3	27.4
	FCNN	3953.2	63.9	70.7	72.3	87.2	94.4	57.1	61.9	64.3	57.7	68.8	77.4
	1-FN$_{0.10}$	288.5	81.6	85.6	86.2	90.5	93.8	4.2	15.0	20.5	3.8	15.8	21.8
	1-FN$_{0.20}$	683.9	84.5	87.4	87.4	92.9	95.7	9.9	20.1	25.2	9.2	21.8	26.8
	1-FN$_{0.30}$	1157.0	85.2	87.4	87.4	93.6	96.3	16.8	26.2	30.9	15.5	27.3	33.9
	1-NE$_{0.10}$	233.3	80.9	84.9	85.7	89.8	93.2	3.4	14.3	19.7	2.9	14.8	20.3
	1-NE$_{0.20}$	574.6	**84.4**	87.2	87.1	92.8	95.4	**8.2**	18.6	23.8	7.4	19.3	25.3
	1-NE$_{0.30}$	1022.7	85.6	87.7	87.4	93.8	96.2	14.7	24.3	29.1	13.3	25.4	31.1
	2-FN$_{0.10}$	257.5	**81.6**	85.5	86.2	90.4	93.9	**3.8**	14.7	20.1	3.3	14.9	20.6
	2-FN$_{0.20}$	609.8	**84.5**	87.2	87.1	92.7	95.5	**8.9**	19.3	24.4	8.1	20.0	26.1
	2-FN$_{0.30}$	1061.7	85.0	87.6	87.3	93.8	96.2	15.5	25.1	29.9	14.2	26.0	32.6
	2-NE$_{0.10}$	186.9	80.2	84.6	85.6	89.5	93.0	2.8	13.7	19.2	2.4	14.5	20.5
	2-NE$_{0.20}$	472.5	84.0	87.0	86.9	92.4	95.2	6.9	17.3	22.6	6.2	18.1	24.8
	2-NE$_{0.30}$	864.5	85.2	87.7	87.3	93.6	96.2	12.5	22.4	27.3	11.5	24.0	29.9
	DROP3	759.8	68.7	80.6	83.4	86.2	92.2	10.5	20.8	26.0	9.3	18.9	23.7
	ICF	987.3	57.5	71.8	76.6	77.3	86.3	14.2	24.2	29.2	14.0	23.4	28.1
	CHC	160.5	**67.8**	79.2	81.7	84.3	89.8	**2.4**	13.4	19.0	1.9	11.4	16.1
3	ALL	6898.7	82.9	82.9	82.9	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	ED	4321.9	88.1	89.5	89.4	94.7	96.8	62.7	67.1	68.7	62.7	74.7	80.8
	MED	4334.9	87.7	**89.5**	89.6	94.6	96.8	62.9	**66.9**	69.1	62.9	74.9	81.1
	MCNN	679.1	82.8	88.6	89.3	93.2	96.0	9.2	19.6	24.7	9.6	21.5	28.0
	CNN	4081.8	64.3	82.4	82.7	87.3	94.6	59.1	63.8	66.2	59.4	71.5	77.8
	MFCNN	675.5	84.5	**88.8**	89.3	93.5	96.2	9.6	**19.8**	25.0	9.8	21.7	28.0
	FCNN	3951.3	63.9	82.1	82.6	87.2	94.4	57.0	61.9	64.3	57.4	69.7	75.7
	1-FN$_{0.10}$	288.4	81.6	86.7	87.9	90.7	93.8	4.2	15.0	20.5	3.7	15.8	21.7
	1-FN$_{0.20}$	682.6	84.5	88.3	89.1	92.9	95.8	9.9	20.1	25.2	9.0	20.4	26.9
	1-FN$_{0.30}$	1155.8	85.1	88.8	89.2	93.7	96.3	16.7	26.2	30.9	15.7	27.7	34.0
	1-NE$_{0.10}$	233.5	80.9	**85.9**	87.2	89.9	93.2	3.4	**14.3**	19.7	2.9	14.4	21.2
	1-NE$_{0.20}$	575.0	84.3	**88.2**	**88.9**	92.8	95.4	8.3	**18.6**	**23.8**	7.5	19.3	26.0
	1-NE$_{0.30}$	1023.3	85.5	88.8	89.3	93.7	96.2	14.7	24.3	29.1	13.6	25.8	31.8
	2-FN$_{0.10}$	257.9	81.5	**86.5**	87.7	90.5	93.9	3.8	**14.7**	20.1	3.4	15.4	21.1
	2-FN$_{0.20}$	610.3	84.5	88.2	88.9	92.8	95.5	8.9	19.3	24.4	8.2	20.1	26.1
	2-FN$_{0.30}$	1061.6	85.0	88.9	89.2	93.8	96.3	15.5	25.1	29.9	14.1	26.0	32.2
	2-NE$_{0.10}$	187.6	**80.3**	**85.6**	87.0	89.6	93.0	**2.8**	**13.7**	19.2	2.3	13.9	20.3
	2-NE$_{0.20}$	472.7	**84.1**	**88.0**	88.7	92.4	95.1	**6.9**	**17.3**	22.6	6.1	18.2	23.9
	2-NE$_{0.30}$	864.4	**85.4**	88.8	89.3	93.6	96.2	**12.5**	22.4	27.3	11.5	24.1	29.8
	DROP3	513.1	65.5	79.7	83.5	83.7	90.1	7.0	17.6	23.0	6.2	15.9	20.5
	ICF	917.4	58.8	74.1	79.3	77.8	86.1	13.4	23.5	28.5	13.1	22.4	27.1
	CHC	238.8	68.8	82.0	85.5	84.9	90.4	3.6	14.6	20.1	3.0	12.4	16.8
5	ALL	6898.7	87.6	87.6	87.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	ED	4331.0	87.7	89.3	88.8	94.6	96.8	62.9	67.0	68.9	63.6	75.8	82.1
	MED	4325.3	87.9	89.1	**89.7**	94.6	96.8	62.8	66.8	**69.0**	63.3	74.9	81.9
	MCNN	669.4	82.8	88.2	**89.3**	93.1	96.1	9.0	19.4	**24.4**	9.6	22.2	28.1
	CNN	4102.3	64.4	82.9	87.0	87.3	94.6	59.3	63.9	66.5	59.8	72.0	78.3
	MFCNN	676.3	84.5	88.6	**89.4**	93.5	96.3	9.6	19.9	**25.0**	9.8	21.9	27.9

Table 2 (*continued*)

k	Algorithm	Red. set size	Accuracy (%)			Upper Bound (%)		Distances (%)			Time (%)		
			PS	kNN2	kNN3	kNN2	kNN3	PS	kNN2	kNN3	PS	kNN2	kNN3
	FCNN	3953.7	63.9	82.6	86.9	87.2	94.4	57.1	61.9	64.3	58.3	71.7	76.2
	1-FN$_{0.10}$	288.6	**81.7**	86.5	87.9	90.7	93.8	**4.2**	15.0	20.5	3.8	15.8	21.5
	1-FN$_{0.20}$	683.2	84.5	88.0	89.1	92.9	95.8	9.9	20.1	25.2	9.3	21.5	27.6
	1-FN$_{0.30}$	1156.4	85.2	88.5	89.3	93.6	96.3	16.7	26.2	30.9	15.7	28.0	33.7
	1-NE$_{0.10}$	233.0	80.9	85.6	87.3	89.8	93.2	3.4	14.3	19.7	2.9	15.1	20.9
	1-NE$_{0.20}$	574.7	84.4	87.9	88.9	92.8	95.4	8.2	18.6	23.8	7.5	19.6	25.7
	1-NE$_{0.30}$	1022.5	85.7	88.6	89.4	93.7	96.2	14.7	24.3	29.1	13.6	26.1	31.5
	2-FN$_{0.10}$	257.7	81.5	86.3	87.8	90.5	93.9	3.8	14.7	20.1	3.4	15.3	21.4
	2-FN$_{0.20}$	609.9	84.5	88.0	89.0	92.7	95.5	8.9	19.3	24.4	8.2	20.6	26.0
	2-FN$_{0.30}$	1061.2	85.0	88.6	89.3	93.7	96.2	15.5	25.1	29.9	14.2	25.5	32.9
	2-NE$_{0.10}$	187.2	80.3	85.4	87.1	89.5	93.1	2.8	13.7	19.2	2.4	15.0	20.9
	2-NE$_{0.20}$	472.2	84.0	87.8	88.7	92.5	95.2	6.9	17.3	22.6	6.1	17.9	24.1
	2-NE$_{0.30}$	863.9	85.4	88.6	89.3	93.6	96.2	12.5	22.4	27.3	11.6	24.1	29.9
	DROP3	466.4	65.3	79.2	83.6	83.1	89.5	6.3	17.1	22.5	5.7	15.4	19.9
	ICF	898.7	58.9	74.2	79.7	77.8	86.1	13.2	23.3	28.3	13.2	22.6	27.4
	CHC	269.0	67.9	81.3	84.9	85.2	90.9	4.2	15.1	20.6	3.6	13.2	17.8
7	ALL	6898.7	88.1	88.1	88.1	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	ED	4335.3	88.1	89.0	89.5	94.6	96.8	62.9	66.9	69.0	63.2	76.2	80.3
	MED	4335.1	87.2	89.0	89.5	94.6	96.8	62.9	67.0	69.1	62.5	74.3	80.9
	MCNN	673.3	82.0	88.0	89.0	93.1	96.0	9.0	19.4	24.4	9.4	21.9	27.6
	CNN	4092.4	64.3	82.7	87.2	87.3	94.5	59.2	63.9	66.3	59.2	71.1	78.0
	MFCNN	678.7	84.3	88.4	89.2	93.5	96.2	9.6	19.9	25.0	9.9	21.6	28.3
	FCNN	3949.4	63.9	82.5	87.2	87.2	94.4	57.0	61.8	64.3	56.9	69.6	74.6
	1-FN$_{0.10}$	288.8	81.7	86.3	87.6	90.6	94.0	4.2	15.1	20.5	3.7	15.6	21.4
	1-FN$_{0.20}$	683.2	84.6	87.9	88.9	92.9	95.8	9.9	20.1	25.2	9.1	21.3	26.6
	1-FN$_{0.30}$	1156.7	85.2	88.3	89.1	93.6	96.3	16.7	26.2	30.9	15.6	27.7	33.6
	1-NE$_{0.10}$	233.1	**81.0**	85.5	87.1	89.8	93.2	**3.4**	14.3	19.7	2.9	14.6	20.8
	1-NE$_{0.20}$	574.0	84.4	87.8	88.6	92.8	95.4	8.2	18.6	23.8	7.4	19.2	25.6
	1-NE$_{0.30}$	1021.9	85.6	88.5	89.2	93.7	96.2	14.7	24.3	29.1	13.7	25.8	32.6
	2-FN$_{0.10}$	257.8	81.6	86.1	87.6	90.4	93.9	3.8	14.7	20.1	3.4	15.4	21.7
	2-FN$_{0.20}$	609.4	84.5	87.9	88.7	92.7	95.5	8.9	19.3	24.4	8.1	19.7	26.3
	2-FN$_{0.30}$	1060.6	85.0	88.5	89.0	93.7	96.2	15.5	25.1	29.9	14.1	25.9	32.7
	2-NE$_{0.10}$	187.3	80.3	85.2	86.8	89.6	93.1	2.8	13.7	19.2	2.3	14.4	19.9
	2-NE$_{0.20}$	472.8	84.1	87.5	88.5	92.3	95.1	6.9	17.4	22.6	6.2	18.2	24.4
	2-NE$_{0.30}$	864.4	85.4	88.4	89.1	93.6	96.2	12.5	22.4	27.3	11.4	23.9	29.5
	DROP3	478.9	66.4	80.6	85.0	83.6	89.9	6.6	17.3	22.6	5.7	15.3	19.9
	ICF	887.9	59.3	74.2	79.7	77.9	86.2	13.0	23.1	28.2	12.9	22.1	26.9
	CHC	296.2	67.8	81.7	85.7	84.7	90.7	4.5	15.5	20.9	3.9	13.5	18.0

Table 3
Average results obtained when 40% of noise is added to the datasets. Bold elements correspond to the non-dominated points. Normalized results (%) of the different algorithms are obtained referring to the ALL method with the same *k* value.

k	Algorithm	Red. set size	Accuracy			Upper bound		Distances			Time		
			PS	kNN2	kNN3	kNN2	kNN3	PS	kNN2	kNN3	PS	kNN2	kNN3
1	ALL	6898.7	60.4	60.4	60.4	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	ED	3037.8	85.1	85.4	83.9	93.4	96.1	44.1	50.2	53.5	44.0	55.1	63.2
	MED	3026.8	85.2	85.2	84.0	93.5	96.1	44.0	50.2	53.3	44.8	57.0	63.5
	MCNN	639.1	80.6	84.2	83.4	92.1	95.1	8.9	19.6	24.1	9.0	21.3	27.0
	CNN	5096.7	52.8	57.5	59.2	78.5	89.5	73.9	76.9	78.6	75.5	87.3	94.5
	MFCNN	596.3	80.4	83.7	83.7	92.2	95.6	8.4	18.7	24.0	8.5	20.8	26.9
	FCNN	4989.8	52.4	57.3	59.1	78.4	89.4	72.2	75.4	76.9	71.8	84.0	89.6
	1-FN$_{0.10}$	332.7	81.5	84.2	83.7	90.5	93.9	4.8	15.6	21.0	4.5	16.5	22.7
	1-FN$_{0.20}$	789.9	82.7	84.9	84.1	92.3	95.3	11.4	21.4	26.4	10.7	23.1	28.9
	1-FN$_{0.30}$	1342.9	78.1	81.1	81.9	91.4	95.1	19.4	28.5	33.1	18.3	30.4	36.7
	1-NE$_{0.10}$	304.6	81.6	84.4	83.7	90.6	93.9	4.4	15.2	20.6	3.9	16.0	21.7
	1-NE$_{0.20}$	740.8	83.2	85.0	84.1	92.4	95.5	10.7	20.7	25.8	9.8	22.4	27.5
	1-NE$_{0.30}$	1280.8	78.9	81.5	82.3	91.9	95.3	18.4	27.7	32.2	17.4	29.5	35.8
	2-FN$_{0.10}$	291.4	**81.8**	84.3	83.9	90.6	93.9	**4.3**	15.1	20.5	3.7	15.4	21.6
	2-FN$_{0.20}$	700.3	82.6	84.8	84.1	92.2	95.3	10.2	20.3	25.4	9.1	20.9	27.1
	2-FN$_{0.30}$	1187.9	81.1	83.7	83.1	92.4	95.5	17.2	26.6	31.3	15.3	27.1	33.4
	2-NE$_{0.10}$	246.9	81.5	84.2	83.5	90.3	93.4	3.6	14.5	19.9	3.0	14.4	20.2
	2-NE$_{0.20}$	613.4	**83.4**	85.2	84.1	92.6	95.6	**8.9**	19.1	24.3	7.7	18.7	25.7
	2-NE$_{0.30}$	1087.8	82.0	84.1	83.4	92.7	95.8	15.7	25.2	30.0	13.5	25.5	30.6
	DROP3	759.8	56.6	70.8	74.9	77.5	86.9	10.5	20.8	26.0	9.3	19.0	23.7
	ICF	987.3	47.3	62.7	69.6	68.3	80.7	14.2	24.2	29.2	14.0	23.6	28.1
	CHC	157.5	**54.5**	69.1	73.5	73.2	81.5	**2.3**	13.4	19.0	1.9	11.6	16.2
3	ALL	6898.7	72.5	72.5	72.5	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	ED	3043.8	85.5	87.8	87.7	93.5	96.1	44.2	50.4	53.5	45.3	57.9	64.2
	MED	3038.8	85.1	86.8	87.7	93.5	96.1	44.1	50.2	53.5	45.0	56.7	64.6
	MCNN	613.2	78.9	85.3	87.0	91.4	95.0	8.3	18.8	23.8	8.8	20.9	28.2
	CNN	5088.0	52.8	73.5	71.9	78.6	89.5	73.8	76.9	78.4	75.7	87.9	94.5
	MFCNN	596.9	81.2	86.8	87.4	92.2	95.6	8.4	18.8	23.9	8.8	21.5	27.7
	FCNN	4989.2	52.4	73.1	71.7	78.3	89.4	72.1	75.4	76.9	73.7	85.9	92.4
	1-FN$_{0.10}$	333.1	81.5	85.8	86.6	90.5	93.8	4.9	15.6	21.0	4.5	17.1	22.6
	1-FN$_{0.20}$	790.3	82.7	87.2	87.4	92.4	95.4	11.4	21.4	26.4	10.8	23.2	29.3
	1-FN$_{0.30}$	1343.0	78.4	86.2	86.5	91.5	95.1	19.4	28.5	33.1	18.6	31.2	37.1
	1-NE$_{0.10}$	304.7	81.6	86.0	86.7	90.5	94.0	4.4	15.2	20.6	3.8	15.8	21.6
	1-NE$_{0.20}$	741.3	83.2	87.3	87.5	92.4	95.6	10.7	20.7	25.8	9.9	21.8	29.6
	1-NE$_{0.30}$	1281.6	79.5	86.9	86.7	91.9	95.2	18.4	27.7	32.2	17.3	29.6	35.8
	2-FN$_{0.10}$	291.3	81.8	**86.0**	86.9	90.6	94.0	4.3	**15.1**	20.5	3.8	16.1	22.1
	2-FN$_{0.20}$	700.6	82.5	86.9	87.4	92.1	95.2	10.2	20.3	25.4	9.2	21.3	27.6
	2-FN$_{0.30}$	1188.2	81.0	87.1	87.0	92.5	95.5	17.2	26.6	31.3	16.1	28.8	34.7
	2-NE$_{0.10}$	247.3	81.5	**85.8**	86.4	90.4	93.5	3.6	**14.5**	19.9	3.0	14.8	20.8
	2-NE$_{0.20}$	614.0	83.3	87.2	87.5	92.5	95.5	8.9	19.1	24.3	8.0	20.2	26.0
	2-NE$_{0.30}$	1088.3	82.0	87.2	87.4	92.7	95.8	15.7	25.2	30.0	13.6	25.8	31.1
	DROP3	513.1	55.0	72.3	78.5	74.9	84.0	7.0	17.6	23.0	6.3	15.9	20.8
	ICF	917.4	48.2	66.3	74.3	69.1	81.0	13.4	23.5	28.5	13.3	22.7	27.8
	CHC	237.4	55.9	73.2	79.2	75.3	83.5	3.6	14.6	20.1	3.1	12.7	17.6
5	ALL	6898.7	82.8	82.8	82.8	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	ED	3038.8	85.3	87.8	88.4	93.5	96.1	44.1	50.3	53.4	44.1	56.3	62.7
	MED	3030.2	85.2	87.8	88.1	93.5	96.0	44.0	50.3	53.3	44.6	57.4	62.6
	MCNN	611.4	79.0	86.0	87.6	91.1	94.8	8.3	18.6	23.8	8.7	21.3	27.2
	CNN	5086.4	52.7	74.5	81.2	78.4	89.4	73.8	76.9	78.4	74.8	87.3	92.9
	MFCNN	594.1	81.3	**87.3**	88.2	92.3	95.6	8.4	**18.7**	24.0	8.6	20.8	27.4

Table 3 (*continued*)

k	Algorithm	Red. set size	Accuracy			Upper bound		Distances			Time		
			PS	kNN2	kNN3	kNN2	kNN3	PS	kNN2	kNN3	PS	kNN2	kNN3
	FCNN	4988.3	52.3	74.2	81.1	78.3	89.3	72.1	75.3	76.9	73.1	85.9	91.0
	1-FN$_{0.10}$	333.3	81.6	85.9	87.1	90.5	93.9	4.9	15.6	21.0	4.3	16.3	21.9
	1-FN$_{0.20}$	790.3	82.5	87.3	88.0	92.4	95.3	11.4	21.4	26.4	10.5	22.6	28.8
	1-FN$_{0.30}$	1342.9	78.7	86.7	88.0	91.5	95.1	19.4	28.5	33.1	18.5	30.7	37.0
	1-NE$_{0.10}$	304.9	81.7	86.0	87.3	90.5	94.0	4.4	15.2	20.6	3.9	16.0	21.6
	1-NE$_{0.20}$	741.0	83.1	87.4	88.0	92.4	95.6	10.7	20.7	25.8	9.6	21.7	28.0
	1-NE$_{0.30}$	1281.0	79.7	87.1	88.1	91.9	95.3	18.4	27.7	32.2	16.9	29.2	34.8
	2-FN$_{0.10}$	291.4	81.8	86.0	87.3	90.6	94.0	4.3	15.1	20.5	3.7	15.4	21.6
	2-FN$_{0.20}$	700.4	82.3	87.1	88.0	92.1	95.3	10.2	20.3	25.4	9.2	20.9	27.7
	2-FN$_{0.30}$	1187.6	81.2	87.5	88.3	92.5	95.5	17.2	26.6	31.3	15.5	27.4	33.8
	2-NE$_{0.10}$	247.4	**81.6**	85.8	86.8	90.5	93.5	**3.6**	14.5	19.9	3.0	14.5	20.7
	2-NE$_{0.20}$	613.5	83.4	**87.4**	88.1	92.6	95.5	8.9	**19.1**	24.3	7.7	19.6	25.6
	2-NE$_{0.30}$	1087.6	82.1	87.7	88.3	92.8	95.8	15.7	25.2	30.0	13.6	25.9	31.4
	DROP3	466.4	56.0	72.8	79.4	75.0	83.5	6.3	17.1	22.5	5.7	15.3	19.9
	ICF	898.7	48.3	66.8	75.5	69.1	81.2	13.2	23.3	28.3	13.1	22.6	27.2
	CHC	271.4	53.7	72.9	80.6	75.2	84.8	4.2	15.1	20.6	3.7	13.3	17.9
7	ALL	6898.7	85.5	85.5	85.5	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	ED	3045.2	85.5	87.9	88.4	93.5	96.1	44.2	50.5	53.4	43.9	56.9	61.3
	MED	3046.1	84.5	87.9	88.4	93.5	96.1	44.2	50.5	53.4	43.7	56.7	60.3
	MCNN	604.8	78.6	86.4	87.7	91.1	94.9	8.2	18.6	23.7	8.1	19.7	25.4
	CNN	5087.0	52.8	74.7	82.6	78.6	89.5	73.8	76.8	78.5	73.4	85.8	90.3
	MFCNN	593.5	81.2	87.2	**88.3**	92.2	95.5	8.4	18.7	**23.9**	8.3	20.7	25.8
	FCNN	4989.3	52.4	74.4	82.3	78.4	89.4	72.1	75.3	76.9	70.9	85.9	85.0
	1-FN$_{0.10}$	333.2	81.6	85.8	87.1	90.5	93.9	4.9	15.6	21.0	4.2	15.8	20.2
	1-FN$_{0.20}$	789.9	82.7	87.3	88.1	92.3	95.3	11.4	21.4	26.4	10.5	23.0	27.9
	1-FN$_{0.30}$	1343.1	78.4	86.5	87.9	91.5	95.1	19.4	28.5	33.1	17.4	29.3	33.7
	1-NE$_{0.10}$	304.8	81.7	85.9	87.2	90.6	94.0	4.4	15.2	20.6	3.9	16.8	20.8
	1-NE$_{0.20}$	741.0	83.2	87.4	88.2	92.5	95.6	10.7	20.7	25.8	9.0	20.5	25.1
	1-NE$_{0.30}$	1280.5	79.3	86.6	88.1	91.9	95.3	18.4	27.6	32.2	16.1	28.3	31.9
	2-FN$_{0.10}$	291.9	81.7	85.9	87.2	90.6	94.0	4.3	15.1	20.5	3.6	15.4	20.2
	2-FN$_{0.20}$	701.2	82.6	87.0	88.0	92.1	95.3	10.2	20.3	25.4	9.2	21.5	25.8
	2-FN$_{0.30}$	1188.6	81.1	87.1	88.3	92.5	95.6	17.2	26.6	31.3	15.3	28.5	31.2
	2-NE$_{0.10}$	246.8	81.6	85.7	86.7	90.3	93.4	3.6	14.5	19.9	2.9	15.1	18.2
	2-NE$_{0.20}$	613.3	83.4	87.3	88.1	92.5	95.5	8.9	19.1	24.3	7.6	18.8	24.5
	2-NE$_{0.30}$	1087.9	81.9	87.4	**88.4**	92.9	95.8	15.7	25.2	**30.0**	12.6	24.0	28.3
	DROP3	478.9	56.6	73.6	80.2	76.8	85.4	6.6	17.3	22.6	5.7	15.3	20.0
	ICF	887.9	48.8	67.1	75.6	69.4	81.1	13.0	23.1	28.2	12.9	22.4	26.9
	CHC	293.4	56.5	74.8	81.8	77.1	86.2	4.5	15.4	20.9	3.7	13.2	17.5

This particular effect is likely to happen since the samples added by our second step are, theoretically, the noisy ones discarded by the PS algorithm, thus confusing the classifier when low *k* values are used.

CNN and FCNN show some of the worst accuracies obtained in these experiments in terms of PS as they are very sensitive to noise: as stand-alone PS algorithms, they are not able to discard the noisy elements, thus leading to a situation in which there is neither an important size reduction nor a remarkable performance. Furthermore, the use of different *k* values does not upturn the accuracy results. On the other hand, the use of the second kNNc step does improve their accuracy, but still the results remain far from the classification bounds. However, as with ED and MED, introducing high *k* values enhances the obtained accuracy with respect to the low *k* values with the *c* class recommendation.

MFCNN and MCNN are not as affected as CNN and FCNN are at PS stage since they introduce an ED phase in the process: whereas the latter approaches obtained around 50% and 60% in terms of accuracy with around 60% and 70% of computed distances, the former algorithms do achieve precision rates around 80% with roughly 10% of the distances. The improvement obtained when using kNNc with these strategies is also noticeable with high *k* values. Moreover, as it happened in the non-added synthetic noise configuration, kNNc schemes are able to score results not significantly different from the ones achieved by the best performing algorithms (for instance, ED and MED with $k=7$) with just around 25% of the maximum distances computed.

EN and FN methods demonstrated to be interesting algorithms in the non-added noise scenario as they both obtained good accuracies while achieving some of the highest reduction rates. Attending now to the results obtained in the proposed noisy situations, these methods do also stand as an interesting alternative for PS as they behave amazingly well in both terms of accuracy and size, especially the 2-EN and 2-FN configurations: whereas, on average, hardly any of these algorithms score lower accuracies than 80%, the 2-EN and 2-FN ones are always able to score precision results above that mark, in some situations with distance ratios in the range from 3% to 10%. Including kNNc does improve the performance (as in the other cases, for the most part when using high-enough *k* values) and in spite of not outperforming other strategies (an exception to this assertion is the 2-$NE_{0.30}$ case with $k=7$ and $c=3$ in Table 3), accuracies obtained are not significantly different from the best performing algorithms. In addition, distances computed roughly range from 10% to 30% of the maximum, constituting a remarkable trade-off result. It is important to highlight that, in sight of the results obtained, these particular algorithms stand as an attractive alternative to some of the other studied methods for any noise situation as they do not perform any editing operation.

Hybrid algorithms DROP3 and ICF, just as CNN and FCNN, are not capable of coping with noisy situations either since accuracy results are similar or even lower (for instance, the ICF method in which with 40% of synthetic noise is not able to reach 50% of accuracy in any of the proposed PS schemes). However, it must be pointed out that, despite achieving similar accuracy rates, hybrid algorithms do it with a lower amount of distances: as an example, check the 7NN case with 40% of noise in which CNN achieves an accuracy of 52.8% with 73.8% of the total of distances while DROP3 gets 56.6% with just 6.6% of distances. On the other hand, adding the kNNc scheme remarkably enhances the accuracy achieved by these algorithms (in the 7NN3 configuration of ICF, the accuracy is improved in almost 30% with respect to the PS situation) but it also noticeably increases the number of distances to be computed up to, on average, 15% more.

Regarding the CHC evolutionary algorithm when tackling noisy situations, although it still shows one of the highest reduction figures amongst the studied methods (rates around 2–5%), its classification performance is significantly affected as no result is higher than 70%. In this case, the inclusion of kNNc has a similar effect to the hybrid algorithms as it shows a notorious accuracy increase (fixing $c=3$, classification figures improve around 20% and 25% for noise rates of 20% and 40% respectively) paired with a rise in the number of distances of about 10% and 15% points when setting $c=2$ and $c=3$ respectively.

In terms of the classification upper bounds defined with kNNc, as in the scenario without synthetic noise, bounds get higher as the number *c* of class proposals is increased. On average, and as already commented, there is not such a significant increase between $c=2$ and $c=3$ as the one observed when comparing PS and $c=2$. An exception to this remark can be observed, though, in both CNN and FCNN strategies as well as with the hybrid (DROP3 and ICF) and the evolutionary (CHC) algorithms in which, as they are not capable of coping with the noise effects, a high number *c* of class proposals are required.

As in the non-added noise scenario, it is possible to check the non-linear relation between the upper bound and the number of distances to be computed. For instance, for a figure noise of 40%, the 2-$FN_{0.10}$ algorithm with $k=1$ retrieves upper bounds of 90.6% for $c=2$ and 93.9% for $c=3$ with distance figures of 15.1% and 20.5% respectively, which shows that there is 3.3% of improvement with just a 5.4% increase in the total of distances to be computed. These

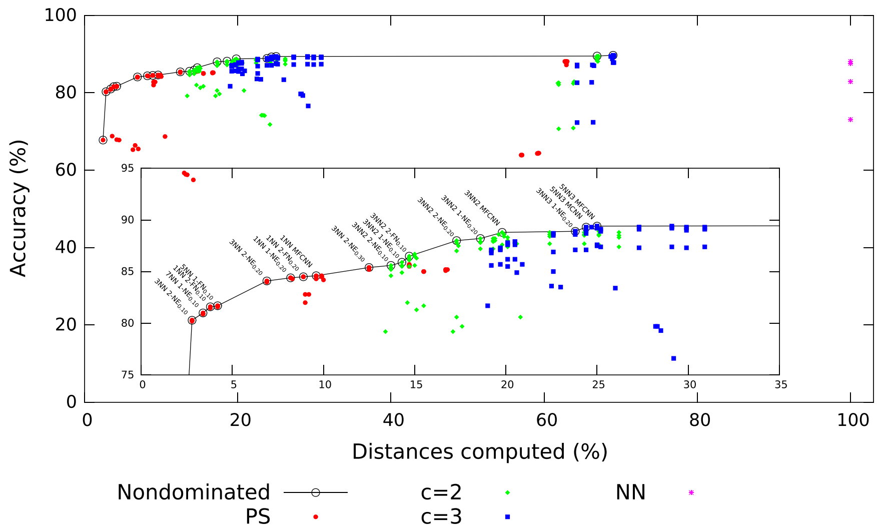

Fig. 3. Distances computed (%) against accuracy achieved by the algorithms. Average results when 20% of noise is added to the samples. The non-dominated front is remarked.

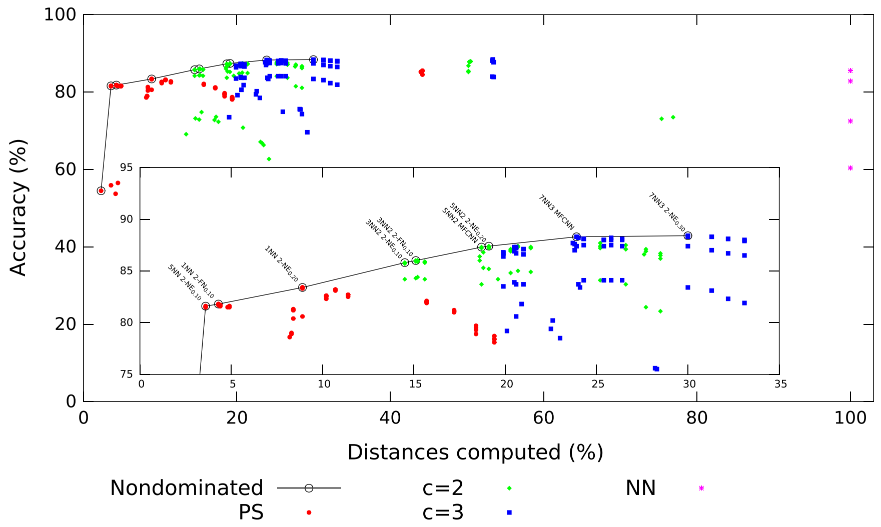

Fig. 4. Distances computed (%) against accuracy achieved by the algorithms. Average results when 40% of noise is added to the samples. The non-dominated front is remarked.

last figures clearly contrast with the 79.5% increase in the distances required so as to be able to improve the 93.9% bound figure to the maximum (barely, 6%). Also, as in the scenario without synthetic noise, the accuracy follows the same non-linear trend of the upper limits but with the remarkable influence of the noise, which might seriously affect the performance: for instance, in the same case of the 2-FN$_{0.10}$ algorithm with $k=1$ and a noise figure of 40%, there is an accuracy improvement from the initial 81.8–84.3% when going from the initial PS to the $c=2$ scheme by computing close to 11% more of distances; however, the use of $c=3$ takes an additional cost of 5% in the distances but, instead of improving results, there is an actual accuracy decrease of almost 0.5%. This clearly shows that the non-linear behavior is not only dependent on the PS algorithm but also on the noise the system is dealing with.

On the other hand, Figs. 3 and 4 show the results of the strategies considered facing accuracy and distances computed. Note that the optimal strategies (non-dominated solutions) are remarked. As occurred in the scenario without noise, PS and kNNc schemes are present within this optimal set. Following the MOP scenario no order can be established within the non-dominated solutions. Nevertheless, it can be checked that now the kNNc strategies are the most numerous.

As it happened in the previous situation in which no noise was added, the basic kNN algorithm is again out of the optimal set of solutions as now kNN3 is not only able to reach its performance with a lower number of distances computed, but it also does obtain a better classification accuracy. Specially interesting are the cases of 5NN3 MFCNN and 5NN3 MCNN, for 20% of noise, and 7NN3 MFCNN and 7NN3 2-NE$_{0.30}$, for 40% of noise, which achieve better performance with just around 25–30% of distances computed.

5.3. Statistical significance test

The aim of this section is to assess whether the inclusion of the second step of the kNNc scheme leads to significantly better classification accuracies. We shall therefore use the KEEL [31] software, which contains statistical tools that allow us to quantify the difference between the results with and without this step. Specifically, a Wilcoxon 1 × 1 test was performed between PS and each kNN2 configuration for the same algorithm as well as between kNN2 and kNN3. The first one checks whether there is a significant accuracy upturn between the kNN2 approach and the basic PS scheme, which is the main contribution of this paper. The second one is performed to assess whether the accuracy in kNN3 is significantly better than the one obtained in kNN2, which may justify providing more class proposals.

Table 4
Asymptotic p-value obtained in Wilcoxon 1 vs. 1 statistical significance test. First column assumes that accuracy of kNN2 is better than accuracy of PS. Second column assumes that accuracy of kNN3 is better than accuracy of kNN2. Bold values represent a level of significance higher than $\alpha=0.95$.

Noise (%)	k	kNN2 vs PS	kNN3 vs kNN2
0	1	**0.000032**	**0.000035**
	3	**0.000044**	**0.000038**
	5	**0.000051**	**0.000038**
	7	**0.000044**	**0.000044**
20	1	**0.00007**	0.117066
	3	**0.000035**	**0.00001**
	5	**0.000038**	**0.000047**
	7	**0.000048**	
40	1	**0.000065**	0.86278
	3	**0.000048**	**0.005203**
	5	**0.000044**	**0.000032**
	7	**0.000021**	**0.00001**

The significant (asymptotic) p-values considering all the experiments are shown in Table 4. These values represent the overlap between the two distributions, assuming that kNNc accuracy is better. We can consider the p-values as a confidence measure for the comparison. The significance of a low value is a high probability that the distributions compared are different.

As is shown in the first column, all the values are lower than 0.05, depicting that the inclusion of our second step leads to a significant accuracy improvement at a confidence level of 95%. Moreover, the second column shows that, except for the two particular configurations of $k=1$ with synthetic noise rates of 20% and 40%, proposing an additional label does lead to higher accuracy as the rest of the confidence values are also lower than 0.05.

6. Conclusions

k-Nearest Neighbor (kNN) classification is one of the most common, easy and simple algorithms for supervised learning which usually achieves an acceptable performance. Within this context, Prototype Selection (PS) algorithms have demonstrated their utility by improving some kNN issues such as computational time, noise removal or memory usage. Nevertheless, PS often leads to a decrease of the classification accuracy. To this end, we propose a two-step strategy in which the PS algorithm is exploited by using its reduced set to select the *c* nearest classes for a given input. Afterwards, only these *c* classes are taken into account in the classification stage with the original set. Therefore, some misclassification produced by using the reduced set can be corrected with neither increasing the computation too much nor requiring the whole training set to be stored in the memory at the same time.

Experimentation in which our strategy was faced against conventional PS-based classification was conducted. A representative set of PS algorithms was chosen and several metrics of interest were collected in classification experiments with some multi-label datasets.

Results showed that our proposal provides a new range of solutions in the trade-off between accuracy and efficiency. In the best cases, our strategy equals the accuracy of kNN classification with just 30% of distances computed. In addition, in the presence of noisy data, our search achieves a remarkably profitable performance since, in combination with the appropriate PS algorithm, it improves the kNN classification with a higher efficiency. Furthermore, in all cases considered, statistical tests revealed that kNNc accuracy is significantly better than the one obtained with just PS.

Some interesting conclusions were also drawn with respect to the tuning parameter *c*. The profitability of increasing *c* did show a non-linear tendency with respect to both the maximum achievable classification rate and the actual accuracy obtained. The improvement gain decreases as the number of recommendations *c* gets higher, depicting an asymptotic behavior. Therefore, an optimal *c* value may be found on the trade-off between accuracy and efficiency depending on the conditions of the considered scenario.

This work has opened some promising future work lines when computing a hypothesis in the second step. Results showed that there is a great gap between the upper bound of the classification (rate in which the correct label is within the *c* classes proposal) and the empirical classification rate. Therefore, other kind of search could be performed in this second step instead of resorting again to the kNN classification.

Conflict of interest

None declared.

Acknowledgements

This work was partially supported by the Spanish Ministerio de Educación, Cultura y Deporte through FPU Fellowship (AP2012–0939), the Spanish Ministerio de Economía y Competitividad through Project TIMuL (TIN2013-48152-C2-1-R), Consejería de Educación de la Comunidad Valenciana through Project PROMETEO/2012/017 and Vicerrectorado de Investigación, Desarrollo e Innovación de la Universidad de Alicante through FPU Program (UAFPU2014–5883).

References

[1] E. Fix, J.L. Hodges, Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties, US Air Force School of Aviation Medicine Technical Report 4 No. 3, 1951, p. 477.
[2] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd ed., John Wiley & Sons, New York, NY, 2001.
[3] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27. http://dx.doi.org/10.1109/TIT.1967.1053964.
[4] T.M. Mitchell, Machine Learning, 1st edition, McGraw-Hill, Inc., New York, NY, USA, 1997.
[5] S. Garcia, J. Derrac, J. Cano, F. Herrera, Prototype selection for nearest neighbor classification: taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 417–435. http://dx.doi.org/10.1109/TPAMI.2011.142.
[6] J. Derrac, N. Verbiest, S. García, C. Cornelis, F. Herrera, On the use of evolutionary feature selection for improving fuzzy rough set based prototype selection, Soft Comput. 17 (2) (2013) 223–238.
[7] S. García, J. Luengo, F. Herrera, Data Preprocessing in Data Mining, of Intelligent Systems Reference Library, vol. 72, Springer, Cham, Switzerland (2015) http://dx.doi.org/10.1007/978-3-319-10247-4.
[8] L. Nanni, A. Lumini, Prototype reduction techniques: a comparison among different approaches, Expert Syst. Appl. 38 (9) (2011) 11820–11828. http://dx.doi.org/10.1016/j.eswa.2011.03.070.
[9] I. Triguero, J. Derrac, S. García, F. Herrera, A taxonomy and experimental study on prototype generation for nearest neighbor classification, IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. 42 (1) (2012) 86–100. http://dx.doi.org/10.1109/TSMCC.2010.2103939.
[10] N. García-Pedrajas, A. De Haro-García, Boosting instance selection algorithms, Knowl. Based Syst. 67 (2014) 342–360. http://dx.doi.org/10.1016/j.knosys.2014.04.021.
[11] C.-F. Tsai, W. Eberle, C.-Y. Chu, Genetic algorithms in feature and instance selection, Knowl. Based Syst. 39 (0) (2013) 240–247. http://dx.doi.org/10.1016/j.knosys.2012.11.005.
[12] J. Derrac, C. Cornelis, S. García, F. Herrera, Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection, Inf. Sci. 186 (1) (2012) 73–92. http://dx.doi.org/10.1016/j.ins.2011.09.027.
[13] J.R. Cano, F. Herrera, M. Lozano, Stratification for scaling up evolutionary prototype selection, Pattern Recognit. Lett. 26 (7) (2005) 953–963. http://dx.doi.org/10.1016/j.patrec.2004.09.043.
[14] F. Angiulli, G. Folino, Distributed Nearest Neighbor-Based Condensation of Very Large Data Sets, IEEE Trans. Knowl. Data Eng. 19 (12) (2007) 1593–1606. http://dx.doi.org/10.1109/TKDE.2007.190665.
[15] P. Hart, The condensed nearest neighbor rule (corresp.), IEEE Trans. Inf. Theory 14 (3) (1968) 515–516. http://dx.doi.org/10.1109/TIT.1968.1054155.
[16] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. 2 (3) (1972) 408–421. http://dx.doi.org/10.1109/TSMC.1972.4309137.
[17] P.A. Devijver, J. Kittler, Pattern Recognition: A Statistical Approach, Prentice Hall, London, United Kingdom, 1982.
[18] B.V. Dasarathy, J.S. Sánchez, S. Townsend, Nearest neighbour editing and condensing tools-synergy exploitation, Pattern Anal. Appl. (2000) 19–30.
[19] F. Angiulli, Fast nearest neighbor condensation for large data sets classification, IEEE Trans. Knowl. Data Eng. 19 (11) (2007) 1450–1464.
[20] J.R. Rico-Juan, J.M. Iñesta, New rank methods for reducing the size of the training set using the nearest neighbor rule, Pattern Recognit. Lett. 33 (5) (2012) 654–660.
[21] D.R. Wilson, T.R. Martinez, Instance pruning techniques, in: Proceedings of the Fourteenth International Conference on Machine Learning, ICML '97, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 403–411.
[22] H. Brighton, C. Mellish, On the consistency of information filters for lazy learning algorithms, in: J. Zytkow, J. Rauch (Eds.), Principles of Data Mining and Knowledge Discovery, of Lecture Notes in Computer Science, vol. 1704, Springer, Berlin, Heidelberg, 1999, pp. 283–288. http://dx.doi.org/10.1007/978-3-540-48247-5_31.
[23] L.J. Eshelman, The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination, in: Proceedings of the First Workshop on Foundations of Genetic Algorithms. Bloomington Campus, IN, USA, July 15–18, 1990, pp. 265–283.
[24] J.R. Cano, F. Herrera, M. Lozano, On the combination of evolutionary algorithms and stratified strategies for training set selection in data mining, Appl. Soft Comput. 6 (3) (2006) 323–332. http://dx.doi.org/10.1016/j.asoc.2005.02.006.
[25] J. Hull, A database for handwritten text recognition research, IEEE Trans. Pattern Anal. Mach. Intell. 16 (5) (1994) 550–554. http://dx.doi.org/10.1109/34.291440.
[26] H. Freeman, On the encoding of arbitrary geometric configurations, IRE Trans. Electron. Comput. 10 (2) (1961) 260–268. http://dx.doi.org/10.1109/TEC.1961.5219197.
[27] R.A. Wagner, M.J. Fischer, The string-to-string correction problem, J. ACM 21 (1) (1974) 168–173. http://dx.doi.org/10.1145/321796.321811.
[28] J. Calvo-Zaragoza, J. Oncina, Recognition of pen-based music notation: the HOMUS dataset, in: Proceedings of the 22nd International Conference on Pattern Recognition, Stockholm, Sweden, 2014, pp. 3038–3043.
[29] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, in: A. Waibel, K.-F. Lee (Eds.), Readings in Speech Recognition, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990, pp. 159–165.
[30] N. Natarajan, I. Dhillon, P. Ravikumar, A. Tewari, Learning with noisy labels, in: Advances in Neural Information Processing Systems, 2013, pp. 1196–1204.
[31] J. Alcalá-Fdez, L. Sánchez, S. García, M.J.D. Jesus, S. Ventura, J.M. Garrell, J. Otero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms to data mining problems, Soft Comput. 13 (3) (2009) 307–318.

Jorge Calvo-Zaragoza received his M.Sc. degree in Computer Engineering from the University of Alicante, in July 2012. He currently holds a FPU Programme fellowship from the Spanish Ministerio de Educación. He is a Ph.D. candidate at the Department of Software and Computing Systems of the University of Alicante, whose research is focused on pattern recognition, soft computing and document analysis.

Jose J. Valero-Mas holds a B.Sc. and an M.Sc. degree in Telecommunications Engineering, from the University of Alicante and the Miguel Hernández University of Elche respectively, besides an M.Sc. in Sound and Music Computing from the Universitat Pompeu Fabra in Barcelona. He is currently a Ph.D. candidate at the Department of Software and Computing Systems of the University of Alicante. His research interests include music information retrieval, signal processing and pattern recognition.

Juan R. Rico-Juan received the M.Sc. degree and Ph.D. in Computer Science from the Polytechnic University of Valencia, in 1992 and from University of Alicante, in 2001, respectively. He is currently a permanent Lecturer in the Department of Software and Computing Systems at the University of Alicante. He is the author of 27 book chapters, 7 papers in international journals and 12 papers in international conferences. He has been involved in around 30 research projects (national and international) covering pattern recognition and artificial intelligence lines of work. His main research interests include rank prototypes for selection, quality prototypes for classification, incremental/adaptive algorithms, structured distances, mean string computation algorithms, ensemble classifiers and edition distances.