On the suitability of Prototype Selection methods for kNN classification with distributed data

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

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

Abstract

In the current Information Age, data production and processing demands are ever increasing. This has motivated the appearance of large-scale distributed information. This phenomenon also applies to Pattern Recognition so that classic and common algorithms, such as the k-Nearest Neighbour, are unable to be used. To improve the efficiency of this classifier, Prototype Selection (PS) strategies can be used. Nevertheless, current PS algorithms were not designed to deal with distributed data, and their performance is therefore unknown under these conditions. This work is devoted to carrying out an experimental study on a simulated framework in which PS strategies can be compared under classical conditions as well as those expected in distributed scenarios. Our results report a general behaviour that is degraded as conditions approach to more realistic scenarios. However, our experiments also show that some methods are able to achieve a fairly similar performance to that of the non-distributed scenario. Thus, although there is a clear need for developing specific PS methodologies and algorithms for tackling these situations, those that reported a higher robustness against such conditions may be good candidates from which to start.

Keywords: Prototype Selection, Distributed data, k-Nearest Neighbour, Experimental study

*Corresponding author: Tel.: +349-65-903772; Fax: +349-65-909326

Email addresses: `jjvalero@dlsi.ua.es` (Jose J. Valero-Mas), `jcalvo@dlsi.ua.es` (Jorge Calvo-Zaragoza), `JuanRamonRico@ua.es` (Juan R. Rico-Juan)

1. Introduction

Nowadays, our society is strongly characterised by the large amount of information surrounding us. Since the start of the information-related technologies, data production has been reported as constantly growing [1], being this effect more remarkable in the past two decades with the popularization of the Internet. Data managing algorithms, thus, have to evolve to be able to cope with such requirements [2].

Such sources of information are often distributed in different nodes, especially when huge amounts of data are presented. A clear example of this paradigm would be *crowd-sourcing*, in which a large number of people work in parallel to perform a specific task (*eg.*, getting labelled data [3]), or *Big Data* [4]. In most cases it is very costly, and even infeasible, to collect all data to be stored in a single node. Therefore, a distributed computing that exploits all the information collected would be of great interest.

Data Mining (DM), considered the main task in the so-called Knowledge Discovery from Databases (KDD) process [5], aims at extracting meaningful patterns from data collections for their later application to the resolution of other problems [6]. In this context, the idea of having several sources of information seems pretty attractive since it stands as an excellent framework to work on. However, current techniques find difficult to manage such scenarios.

Among the different existing DM schemes, supervised learning is the one which aims at obtaining a function out of a set of labelled samples. Within this paradigm, the instance-based learning family comprises those algorithms that directly use the training samples for classification instead of building a model out of them [7].

The k-Nearest Neighbour (kNN) rule is one of the most common instance-based learning algorithms in Pattern Recognition (PR) [8]. For a given input, this algorithm hypothesises about its category by querying its k nearest neighbours of the training set, following a specified similarity measure. In addition to its straightforward implementation, it reports a very competitive performance in many disparate fields and applications. In turn, it presents important drawbacks, many of which become insurmountable in large-scale scenarios: time efficiency, memory consumption and sensitiveness to noise.

Data Reduction (DR) techniques, which constitute a family of preprocessing methods usually found in the KDD process, have been classically considered for tackling the drawbacks of instance-based classification. These techniques aim at reducing the training set size while trying to maintain, if not increasing, the classification accuracy [9]. Prototype Selection (PS) algorithms, as a particular example of DR, perform a selection of the most promising instances for classification.

Most PS algorithms require all instances to be processed at the same time. While this premise may be valid in classic problems, it cannot be assumed in aforementioned scenarios as the large quantity of data makes it infeasible. However, due to the distributed nature of this information, a straightforward approach might be to repeatedly apply PS to the different data subsets and then merge the results obtained.

The goal of this paper is to study the behaviour of classic PS algorithms for kNN classification in this distributed context. Particularly, we shall assess the influence of PS when applied to data spread over several partitions, which are eventually joined for creating a single training set of a kNN classifier. For that, we shall consider a simulated scenario that will allow us to measure the performance of the methods as data increasingly approaches to more distributed conditions.

The rest of the paper is organized as follows: Section 2 presents some related work to this topic; the framework proposed to apply PS with distributed data is described in Section 3; Section 4 details the experimentation performed; Section 5 analyses the results obtained and discusses their implications; finally, Section 6 concludes the present work.

2. Related work

As a representative example of instance-based learning, the kNN classification rule generally exhibits a very poor efficiency: since no model or classification function is built out of the training data, each time a new element has to be labelled all training information has to be consulted. This fact has two clear implications: on the one hand, high storage requirements; on the other hand, an elevated computational cost.

These shortcomings have been widely analysed in the literature and several strategies have been proposed to tackle them. In general, they can be divided into three categories:

- **Fast Similarity Search (FSS):** family of methods which base its performance on the creation of search indexes for fast prototype consulting in the training set.
- **Approximated Similarity Search (ASS):** approaches which work on the premise of searching sufficiently similar prototypes to a given query in the training set instead of retrieving the exact nearest instance.
- **Data Reduction (DR):** set of techniques devoted to lower the training set size while maintaining the classification accuracy.

While the two first approaches focus on improving time efficiency, they do not reduce memory consumption. Indeed, some of these techniques speed-up time response at the expense of increasing this factor. Therefore, when large datasets are present in a PR task, the DR framework rises as a suitable option to consider.

DR techniques aim at reducing the size of the initial training set while keeping the same recognition performance [5]. Among the different possible methodologies, the two most common approaches are Prototype Generation (PG) and Prototype Selection (PS) [10]. Both families reduce the initial training set size by discarding redundant information besides removing noisy instances. However, while PG creates new artificial data for replacing the initial information, PS simply selects the most promising elements from the training 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. Reader may check reference [11] for a detailed explanation and thorough study of PG techniques. On the other hand, due to its relevance in the present paper, we now introduce the basics of PS methods.

Given the importance of PS methods in terms of removing both redundant and noisy instances, many different approaches have been proposed. Although a wide range of taxonomies have been proposed for classifying the existing methods, we focus on a particular criterion which establishes three different families:

- **Condensing**: These techniques focus on keeping instances close to decision boundaries and discarding the rest. Accuracy on training set is usually maintained but generalization accuracy tends to decrease.
- **Editing**: These methods try to minimise the overlapping among the different classes, which generally take place close to the decision boundaries or because of class outliers. Although data reduction figures are lower than in the previous case, generalization accuracy is higher.
- **Hybrid**: Family of algorithms which looks for a compromise between the two previous methodologies, that is looking for the greatest reduction figure which can improve, or at least maintain, the generalization accuracy of the initial set.

Reader may check the work of Garcia et al. [12] for a thorough and more comprehensive explanation of taxonomy criteria for PS algorithms.

Although PS may seem a good option to tackle large-scale data, in practice it cannot be directly applied to these scenarios: even if memory requirements could be fulfilled and the algorithms could be applied, as most of them were not designed for so large datasets, an efficient performance might not be possible and incorrect results would be retrieved [13].

To this end, several strategies have been proposed in order to improve the scalability of PS algorithms. A reported successful methodology has been the information stratification, which basically selects a manageable and representative subset with equally-distributed classes as training out of the total amount of data [14]. This subset can be latter processed as for instance with evolutionary PS algorithms [15, 16] or with the use of memetic PS techniques [17]. With a similar idea, the MapReduce framework has been recently combined with PS in this massive data context [18].

Divide-and-conquer strategies have also been proposed in the literature. A first interesting approach can be found in the work of García-Osorio et al. [19]: the user selects a number of iteration rounds; in each round the initial set is divided into a number of disjoint subsets and a PS process is applied to each of them, receiving a vote each instance selected to be removed; after the established iterations, instances with the highest number of votes are removed. Another remarkable example is the one in [20], in which a number of subsets obtained from the training data are processed using PS and then combined using a voting scheme. A last work to be highlighted is the one of Haro-García et al. [21], in which the divide-and-conquer policy is recursively applied to the data: the initial set is divided into a number of subsets with equally-distributed classes; then each of them is processed with a PS algorithm; the resulting subsets are gathered into one and, then, the process starts over again. The training set is divided into two parts for performing a cross validation evaluation while performing the process and the algorithm iterates until the validation error starts to grow.

As pointed out by several authors [19, 22], the division of the initial set into small data excerpts for their independent processing may seriously affect the overall accuracy as each subset only considers a (limited) part of the problem, therefore missing the general vision. While it seems that this effect may be palliated by forcing equality in the class distribution of the instances, this situation cannot be generalised to large-scale distributed situations since overall data distribution cannot be *a priori* known.

Thus, in this paper we address a kNN classification experimental study for quantifying the aforementioned degradation as data is arranged in an increasing number of subsets and independently processed with PS. As described above, some works focused on apparently similar ideas but they always assumed that information could be gathered altogether at the beginning of the task. Therefore, to the best of our knowledge, no work has measured this performance variation. A comparative experiment with increasing levels of distribution will be performed. We shall include the case in which no distribution is needed so as to empirically compare the drop caused by this scenario. We believe that this assessment could give remarkable insights about PS applied on corpora which can only be treated in a distributed fashion.

3. Prototype Selection on distributed data

As discussed in Section 2, scalability in PR techniques is not a trivial issue, reason why different approaches have been proposed to palliate that situation. Based on the divide-and-conquer policy, different hierarchical structures combined with PS techniques for reducing the complexity in the task have been proposed. However, it is considered that these approaches bring a decrease in performance when compared to the ideal case of treating all the information as a whole.

In order to assess the aforementioned accuracy drop in distributed scenarios, we consider the most straightforward divide-and-conquer strategy: an initial dataset is arranged in several subsets; PS is performed in each subset; the resulting sets are then united with a given merging criterion; finally, classification is performed using kNN. This scheme is shown in Fig. 1.

One consideration to take into account is that, in terms of assessing the influence of the performance degradation as the information is increasingly distributed, we contemplate the particular case of managing a single partition ($p = 1$). Given that real distributed sets cannot be evaluated in such terms, experimentation shall be performed in a simulated context: using datasets comprising a large number of instances but still manageable by the considered techniques. Conclusions obtained may be extrapolated to real cases.

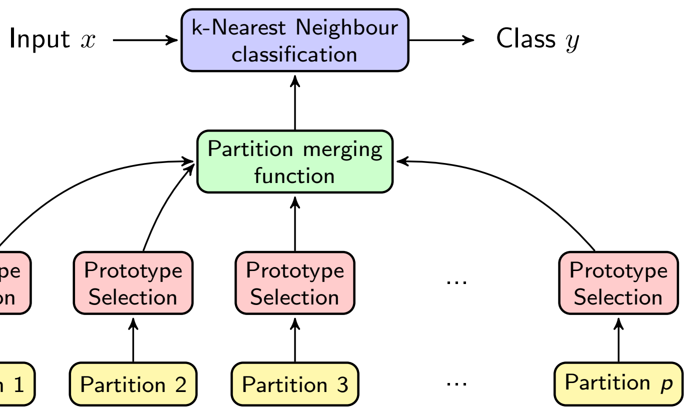

Figure 1: Description of the experimental set-up implemented: initial data set is distributed in p different partitions, being a PS process applied to each subset; a *merging function* is then applied over the PS results, which in our case consists of mixing all these sets, thus obtaining the training set for the kNN classifier.

Common large-scale distributed sources comprise crowd-sourcing, high sensorisation, blind labelling or information gathered from social networks, among many others. Information obtained in these scenarios usually presents an elevated percentage of noisy samples, thus it seems interesting to analyse this effect in the experimentation. For that, noise shall be induced in the evaluation data by swapping the labels of pairs of prototypes randomly chosen. Considered noise rates (percentage of prototypes affected) shall be 20% and 40% as they represent typical values in this kind of experimentation [23].

A basic random strategy is followed for the distribution of the information among the different subsets with the only constraint of having the same amount of instances in each set. No consideration about the resulting class distribution is taken into account. Note that in a real case this would not be possible either.

The merging criterion shall simply gather the results obtained by the single PS schemes applied to the different subsets considered, without any kind of post-processing stage.

It might be argued that the proposed partitions hierarchy, together with the distribution and merging policies, may somehow seem simple and naïve. While this may be true, the experimental and exploratory nature of the present work must be considered: understanding the flaws of such straightforward approach should give some insights about more complex structures capable of mitigating them.

3.1. Prototype Selection algorithms

The collection of PS algorithms considered is now presented. We have selected a representative set covering the different approaches as well as both classical and advanced strategies:

- Condensing Nearest Neighbour (CNN) [24]: starting with an empty set S, prototypes of the initial set are consulted randomly. A prototype is included in S if, and only if, it is not correctly classified with the current prototypes of S. Since prototypes of the initial set are consulted randomly, each computation may give a different reduced set S.

- Editing Nearest Neighbour (ED) [25]: prototypes of the initial training set are consulted randomly. A prototype is removed from the set if, and only if, it is not correctly classified with the current state of the initial set. As in the case of CNN, it may produce different results at each computation because of randomness. A common extension is the Repeated Editing (RED) [26] method, which computes iteratively ED algorithm until convergence.

- Repeated Editing Condensing Nearest Neighbour (RCNN) [27]: it consists in applying ED and CNN algorithms alternatively. The process is repeated while changes are produced in the training set.

- Fast Condensing Nearest Neighbour (FCNN) [28]: based on seeking the centroids of each class label, this methods computes a faster, order-independent condensing strategy. Repeated Editing Fast Condensing Nearest Neighbour (RFCNN) technique, a combination of the ideas of RCNN and FCNN, is also included.

- Farther Neighbour (FN) and Nearest to Enemy (NE) rank methods [29]: following a voting heuristic, a quality score is given to each prototype. These measures are mapped onto probabilities, and prototypes of each class are selected until filling a maximum probability mass (specified as a tuning parameter).

- Decremental Reduction Optimization Procedure 3 (DROP3) [30]: after applying a noise filtering to remove order-dependency, prototypes are ordered according to the distance to their nearest neighbours. Then, starting from the furthest ones, prototypes which do not affect the generalization accuracy are removed.

- Iterative Case Filtering Algorithm (ICF) [31]: seeks for a subset of prototypes that maximises classification accuracy using kNN. This search is commanded by premises of coverage and reachability.

- Cross-generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation (CHC) [32]: a good representative evolutionary algorithm for PS purposes. The configuration of this algorithm has been the same as in [33], that is $\alpha = 0.5$, Population = 50 and Evaluations = 10000.

All these algorithms will be confronted experimentally considering several values of k for the kNN classifier (1, 3 and 5) and number of partitions p (1, 3, 5, 10, 20 and 50).

4. Experimentation

Experimentation is carried out using seven datasets: the NIST Special Database 3 of the National Institute of Standards and Technology (NIST3), from which a subset of the upper case characters was randomly selected; the United States Postal Service (USPS) handwritten digit dataset [34]; the MNIST dataset of handwritten digits [35]; the MPEG-7 shape silhouette dataset (Core Experiment CE-Shape-1 part B) [36]; the Handwritten Online Musical Symbol (HOMUS) dataset [37]; and two corpora from the UCI Machine Learning Repository [38]:

Name	Instances	Classes	Dissimilarity
NIST3	5200	26	ED
USPS	9298	10	ED
MPEG-7	1400	70	ED
MNIST	10000	10	ED
HOMUS	15200	32	DTW
Penbased	10992	10	Euclidean
Letter	20000	26	Euclidean

Table 1: Summary of the datasets used in the experimentation.

Penbased and Letter. Due to the different nature of representation among the datasets, different feature extraction and dissimilarities for the kNN are used. For datasets based on images of isolated symbols (NIST3, USPS, MPEG-7, MNIST), 8-neighbourhood Freeman Chain Code [39] contours are extracted and compared using the edit distance (ED) [40]. For the online symbols of the HOMUS, Dynamic Time Warping (DTW) [41] over raw data is applied. Finally, Penbased and Letter are represented by numerical vectors and, therefore, Euclidean distance is used. Table 1 summarises the main features of the datasets utilized in our experimentation.

The experiments were carried out as follows: the current dataset is divided into training and test sets with a percentage of 80 % and 20 % of the total data, respectively. The training set is distributed at random uniformly into p nodes. Therefore, a node can have a lot more elements of a class than other, otherwise in number of prototypes. PS algorithms operate independently on each node and the surviving prototypes are collected in a higher node. In this node, prototypes are used to classify the test set following the kNN rule. This process is repeated 5 times, whereby a 5-fold cross-validation is performed. The average of these experiments will be presented.

In order to assess the performance achieved by each of the distributed scenarios considered, we take into account the following metrics of interest: i) classification accuracy achieved by the strategy, and ii) the number of distances computed during the classification, directly related to the number of prototypes in the training set. Nevertheless, a recurring issue in PS-related research is the comparison between different approaches. Without putting any of the metrics ahead of the other one, it is hard to establish a criterion for comparing two given methods in which one achieves a higher accuracy but the other gets a higher reduction.

From this point of view, PS performance can be considered as a Multi-objective Optimization Problem (MOP) in which two functions are meant to be optimised at the same time: achieving the minimum number of prototypes in the training set while maximising the classification success rate. This kind of problems are usually evaluated in terms of the so-called *non-dominance* concept. One solution is said to dominate another if, and only if, it is better or equal in each goal and, at least, strictly better in one of them. Non-dominated elements, which may be more than one, constitute the optimal solutions under this criterion.

Considering the scheme proposed in Section 3 and the aforementioned experimentation, we now introduce the results obtained. While a thorough discussion of these figures is developed in Section 5, we shall now introduce some general remarks obtained with an initial inspection of the results.

Figures presented constitute the average of the results obtained for each single dataset. Due to the influence of the induced noise in the overall performance, this information is presented in two different sections: a first one in which the datasets suffer no alteration and a second one in which the proposed noise ratios are induced.

4.1. No induced noise experiment

Table 2 shows the results obtained in the scenario without induced noise. Note that, due to this absence, results among the different k configurations do not significantly differ. Bold values represent non-dominated results for each single partition value.

As expected, the best accuracy is obtained when no PS algorithm is applied as no data is discarded. Although this case would be infeasible in the real

k	Algorithm	Accuracy (%)						Distances (%)					
		1	3	5	10	20	50	1	3	5	10	20	50
	ALL	**92.7**	92.7	92.7	92.7	92.7	92.7	**100.0**	100.0	100.0	100.0	100.0	100.0
	ED	**91.4**	**90.8**	90.6	90.7	90.6	90.4	**94.7**	**91.4**	89.3	85.6	80.2	70.7
	RED	**91.2**	**90.7**	90.6	90.8	90.5	90.3	**94.6**	**91.2**	89.1	85.1	79.6	70.1
	CNN	90.0	**90.7**	**91.0**	**91.2**	**91.6**	**91.9**	21.8	**28.3**	**32.0**	**38.1**	**45.1**	**55.7**
	RCNN	**89.1**	**88.9**	**88.9**	**89.5**	89.4	89.1	**16.4**	**19.6**	**21.5**	**23.8**	25.7	27.2
	FCNN	**90.1**	**90.6**	**90.8**	**91.1**	**91.5**	91.6	**21.7**	**27.8**	**31.4**	**37.0**	**43.3**	52.9
	RFCNN	89.1	88.8	**88.8**	**89.4**	**89.5**	**89.2**	16.5	19.8	**21.4**	**23.4**	**24.6**	**25.4**
1	1-NE$_{0.10}$	**72.4**	**76.8**	79.0	81.2	**83.6**	85.9	**8.8**	**9.6**	10.1	11.2	**12.9**	16.9
	1-NE$_{0.20}$	80.1	82.4	83.6	84.9	**86.5**	87.9	10.7	12.2	13.1	14.7	**16.8**	21.3
	1-NE$_{0.30}$	**84.9**	85.9	**86.8**	**87.5**	**88.4**	**89.5**	**13.6**	15.8	**17.1**	**19.1**	**21.8**	**26.8**
	1-FN$_{0.10}$	**79.0**	81.5	**82.9**	83.6	**84.8**	86.5	**10.1**	10.9	**11.4**	12.1	**13.4**	17.0
	1-FN$_{0.20}$	84.1	**85.3**	86.0	**86.7**	87.4	88.3	13.6	**15.0**	15.6	**16.6**	18.2	22.0
	1-FN$_{0.30}$	86.9	87.5	88.0	88.2	88.9	89.5	18.1	20.0	20.7	21.9	23.7	27.8
	ICF	78.4	83.6	84.7	86.0	86.1	84.6	22.7	29.5	33.4	38.3	43.2	46.8
	DROP3	84.3	87.5	87.9	87.9	87.2	85.2	15.5	20.1	22.9	27.2	31.5	36.1
	CHC	**83.0**	**84.1**	**84.4**	83.1	81.5	79.0	**9.9**	**10.9**	**11.4**	11.4	12.5	15.7
	ALL	92.6	92.6	92.6	92.6	92.6	92.6	100.0	100.0	100.0	100.0	100.0	100.0
	ED	91.0	90.4	90.4	90.7	90.4	90.1	94.6	91.1	88.7	84.4	78.6	68.8
	RED	**90.8**	90.2	90.3	90.5	90.2	89.8	**94.3**	90.5	87.8	82.9	76.2	65.8
	CNN	89.5	90.2	90.7	91.1	91.6	91.9	22.3	29.1	33.0	39.3	46.5	56.7
	RCNN	**89.0**	**88.9**	**88.7**	89.3	89.4	89.1	**16.2**	**19.3**	**21.0**	23.2	24.9	26.1
	FCNN	90.0	**90.5**	**90.7**	**91.1**	91.5	91.6	21.7	**27.7**	**31.3**	**36.9**	43.3	52.8
	RFCNN	88.8	88.6	88.6	**89.4**	**89.2**	**89.0**	16.3	19.3	20.8	**22.6**	**23.7**	**24.2**
3	1-NE$_{0.10}$	72.3	76.5	**79.1**	81.1	83.4	**86.1**	8.6	9.4	**9.9**	10.9	12.7	**16.7**
	1-NE$_{0.20}$	80.0	**82.6**	83.5	**85.0**	86.5	87.9	10.5	**12.0**	12.9	**14.5**	16.6	21.1
	1-NE$_{0.30}$	84.8	85.9	86.7	87.4	88.4	89.4	13.4	15.6	16.9	18.9	21.6	26.6
	1-FN$_{0.10}$	79.0	81.6	82.7	83.7	84.8	**86.6**	9.9	10.7	11.2	11.9	13.2	**16.8**
	1-FN$_{0.20}$	84.1	85.1	**86.1**	86.5	**87.4**	88.4	13.4	14.7	**15.4**	16.4	**18.0**	21.8
	1-FN$_{0.30}$	86.9	87.4	87.8	88.2	88.9	89.5	17.8	19.7	20.5	21.7	23.5	27.6
	ICF	79.1	83.2	83.9	84.5	83.2	79.3	22.1	28.4	31.8	35.8	39.1	40.4
	DROP3	80.4	85.1	85.8	86.1	84.2	80.2	13.1	17.6	20.4	24.7	28.7	33.0
	CHC	**83.3**	83.0	82.5	80.4	74.3	75.6	**11.4**	12.6	12.9	12.3	13.3	16.8
	ALL	92.2	92.2	92.2	92.2	92.2	92.2	100.0	100.0	100.0	100.0	100.0	100.0
	ED	90.7	90.3	90.2	90.4	90.2	90.0	94.5	90.7	88.0	83.3	76.9	66.9
	RED	**90.3**	90.0	89.8	90.0	89.8	89.4	**94.1**	89.8	86.6	80.9	73.7	63.4
	CNN	88.8	90.1	90.6	91.2	91.6	91.9	22.4	29.7	34.0	40.6	48.0	58.8
	RCNN	**88.4**	**88.4**	**88.7**	89.2	89.1	89.0	**16.1**	**19.1**	**20.6**	22.6	23.9	25.2
	FCNN	90.0	90.6	90.7	90.9	91.3	**91.8**	21.7	27.8	31.4	36.9	43.3	**52.9**
	RFCNN	**88.5**	**88.5**	**88.5**	**89.1**	**89.1**	88.8	**16.1**	**19.2**	**20.5**	**22.0**	**22.6**	23.1
5	1-NE$_{0.10}$	72.3	76.8	79.0	**81.3**	83.6	86.1	8.4	9.2	9.8	**10.8**	12.5	16.5
	1-NE$_{0.20}$	**80.2**	82.5	83.5	84.9	86.5	**88.0**	**10.3**	11.8	12.7	14.3	16.4	**21.0**
	1-NE$_{0.30}$	84.7	**86.0**	86.7	87.5	88.4	89.2	13.2	**15.5**	16.7	18.8	21.4	26.4
	1-FN$_{0.10}$	79.0	**81.7**	82.8	**83.8**	**84.7**	86.6	9.8	**10.6**	11.0	**11.8**	**13.1**	16.7
	1-FN$_{0.20}$	84.1	85.3	86.1	86.6	87.4	**88.5**	13.3	14.6	15.2	16.3	17.8	**21.6**
	1-FN$_{0.30}$	87.0	87.3	87.9	88.3	88.7	**89.6**	17.7	19.6	20.4	21.6	23.4	**27.4**
	ICF	79.1	82.6	83.4	83.7	81.8	77.9	21.9	28.1	31.3	35.0	37.8	38.6
	DROP3	79.8	85.0	85.5	85.0	82.8	78.8	12.7	17.5	20.4	24.6	28.9	33.4
	CHC	83.3	82.2	81.5	75.1	73.6	75.1	11.5	13.4	13.9	13.0	13.8	17.3

Table 2: Average results when no noise is induced to the dataset. Each column depicts the number of partitions used in the configuration. Normalised results (%) of the different algorithms are obtained referring to the ALL method with the same k value. Bold values represent the non-dominated elements for each p configuration.

scenario, this figure acts as an upper bound in the results. When PS methods are introduced, results show two main trends as the number of partitions increases: on the one hand, editing methods (ED and RED) suffer a progressive accuracy drop paired with a systematic reduction in the number of prototypes; on the other hand, the rest of the methods improve their classification rate by allowing a higher number of prototypes.

Analysing the non-dominated elements, there is a large number of configurations accomplishing this criterion. For instance CNN-based methods, and especially RCNN and RFCNN, have a relevant presence in all the considered k values. NE and FN families also provide a remarkable amount of non-dominated elements, but it is highly related to its parametrisation. Finally, CHC stands as an interesting alternative as it achieves great reduction rates keeping high accuracy values, thus accomplishing the aforementioned criterion.

4.2. Induced noise experiment

Table 3 introduces the results obtained for experiments when a figure of 20 % of noise is induced in the training information, along with the non-dominated results for each configuration of p in bold.

An important remark to begin with is that, on average, the reduction rates depicted by the different algorithms are very similar to those obtained in the previous scenario. The most noticeable difference is found in the case of editing-based methods, in which there is a remarkable reduction in the number of remaining prototypes. This fact is highly reflected in the classification accuracy: best results are obtained by those methods including an edition process (ED, RED, RCNN, RFCNN). Since these methods remove elements which do not suit with their neighbourhood, they show a superior robustness in noisy scenarios. FCNN, ICF, DROP3 and CHC, on the contrary, suffer from a noticeably accuracy decrease, even when considering high k values. Finally, a special mention must be done to both NE and FN rank methods as they achieve a very competitive performance in both accuracy and reduction without the need for an edition process.

Table 4 reports the results when a 40 % of noise is induced in the data. On average, figures in this case show similar trends to the ones described in the previous case, simply showing a more accused performance degradation due to

k	Algorithm	Accuracy (%)						Distances (%)					
		1	3	5	10	20	50	1	3	5	10	20	50
1	ALL	76.1	76.1	76.1	76.1	76.1	76.1	100.0	100.0	100.0	100.0	100.0	100.0
	ED	**89.4**	**89.0**	**88.6**	87.9	**87.5**	**86.8**	**71.1**	**68.7**	**67.1**	64.1	**60.0**	**53.5**
	RED	**89.2**	**89.0**	**88.5**	**88.0**	**87.3**	**86.8**	**71.0**	**68.4**	**66.7**	**63.6**	**59.5**	**52.7**
	CNN	67.5	66.5	66.6	66.5	67.3	68.6	59.4	61.0	62.2	64.4	67.3	72.3
	RCNN	86.7	**86.4**	86.0	85.5	85.2	84.9	14.8	**17.4**	18.6	20.3	21.6	22.6
	FCNN	67.4	66.1	65.8	65.8	66.5	68.0	57.6	59.0	60.0	62.2	65.0	69.8
	RFCNN	86.5	86.2	85.7	85.3	85.2	**85.0**	14.8	17.2	18.4	19.8	20.8	**21.4**
	1-NE$_{0.10}$	**81.2**	81.1	81.5	81.6	**82.1**	81.1	**10.1**	10.5	10.9	11.7	**13.2**	17.0
	1-NE$_{0.20}$	**85.4**	84.9	**84.5**	84.1	83.4	81.7	**14.3**	14.8	**15.3**	16.4	18.2	22.3
	1-NE$_{0.30}$	86.7	86.0	85.3	84.2	82.8	80.4	19.7	20.2	20.8	22.2	24.4	28.8
	1-FN$_{0.10}$	81.7	**82.3**	**82.1**	**82.3**	82.2	80.5	10.7	**11.2**	**11.5**	**12.3**	13.5	17.1
	1-FN$_{0.20}$	85.3	85.1	**84.6**	84.0	83.4	81.2	15.4	16.1	**16.5**	17.4	18.9	22.7
	1-FN$_{0.30}$	86.7	85.6	85.1	84.0	81.7	79.9	21.0	21.7	22.3	23.4	25.3	29.3
	ICF	74.5	77.4	78.5	79.3	78.2	75.5	28.1	30.9	32.5	35.2	37.4	39.3
	DROP3	78.2	80.7	80.6	80.4	78.9	76.0	17.2	20.7	22.6	25.9	28.9	32.5
	CHC	**74.2**	**76.4**	**76.5**	**74.4**	73.3	**71.9**	**8.3**	**9.0**	**9.2**	**10.0**	12.0	**14.9**
3	ALL	85.6	85.6	85.6	85.6	85.6	85.6	100.0	100.0	100.0	100.0	100.0	100.0
	ED	**89.5**	88.8	88.6	**88.1**	87.3	86.6	**74.1**	71.3	69.3	**65.6**	61.0	53.5
	RED	89.2	88.5	88.4	87.9	**87.0**	**86.3**	73.8	70.4	68.0	63.9	**58.5**	**50.9**
	CNN	65.1	64.4	64.6	64.8	65.6	67.8	55.3	57.6	59.4	62.3	65.8	71.6
	RCNN	87.0	**86.4**	86.1	**86.0**	**85.7**	84.9	14.6	**17.2**	18.4	**19.9**	**21.3**	22.3
	FCNN	67.5	66.0	65.8	65.7	66.6	68.0	57.6	58.9	60.0	62.2	65.0	69.7
	RFCNN	**87.2**	**86.1**	86.2	85.8	**85.3**	**84.8**	**14.6**	**17.1**	18.2	19.4	**20.3**	**20.9**
	1-NE$_{0.10}$	81.0	**81.5**	81.5	**81.7**	81.9	81.1	10.0	**10.4**	10.8	**11.6**	13.1	16.9
	1-NE$_{0.20}$	85.4	**85.2**	84.5	84.3	**83.6**	81.7	14.2	**14.7**	15.2	16.3	**18.1**	22.2
	1-NE$_{0.30}$	86.7	86.2	85.1	84.1	82.5	80.3	19.5	20.1	20.7	22.1	24.3	28.6
	1-FN$_{0.10}$	**81.8**	82.2	82.0	82.3	82.2	81.0	**10.6**	11.1	11.4	12.2	13.4	17.0
	1-FN$_{0.20}$	85.3	85.1	84.5	84.2	83.3	81.1	15.3	15.9	16.4	17.3	18.8	22.6
	1-FN$_{0.30}$	86.6	85.5	85.0	83.6	82.5	80.2	20.9	21.6	22.1	23.3	25.2	29.2
	ICF	75.6	78.3	77.7	77.4	75.1	71.2	28.9	31.0	32.3	33.9	35.0	34.8
	DROP3	75.1	78.4	79.1	78.4	75.5	72.4	13.3	17.5	20.1	23.6	27.3	30.5
	CHC	73.2	73.2	73.0	**70.5**	66.8	67.1	10.0	11.0	10.7	**10.8**	12.8	15.9
5	ALL	**89.8**	89.8	89.8	89.8	89.8	89.8	**100.0**	100.0	100.0	100.0	100.0	100.0
	ED	88.9	88.4	88.4	87.8	86.9	86.3	75.1	72.1	69.7	65.9	60.7	52.9
	RED	88.8	88.2	87.9	**87.5**	**86.5**	**85.9**	74.6	70.7	68.0	**63.4**	**57.5**	**49.7**
	CNN	61.1	61.6	61.9	62.7	64.3	67.3	50.7	54.4	56.6	60.5	65.0	71.6
	RCNN	86.7	86.2	**86.6**	86.0	85.1	84.8	14.5	17.2	**18.2**	19.6	20.7	21.7
	FCNN	67.4	66.0	65.8	65.8	66.6	67.9	57.6	58.9	60.0	62.2	65.0	69.8
	RFCNN	**86.8**	85.8	**86.0**	**85.9**	**85.1**	**84.8**	**14.5**	17.1	**18.0**	**19.1**	**19.7**	**20.2**
	1-NE$_{0.10}$	81.1	81.4	81.6	81.6	**82.1**	**81.2**	9.9	10.3	10.7	11.5	**12.9**	**16.8**
	1-NE$_{0.20}$	85.4	85.2	**84.5**	**84.4**	83.5	81.3	14.1	14.6	**15.1**	**16.2**	17.9	22.1
	1-NE$_{0.30}$	86.7	86.2	85.1	84.0	82.7	80.6	19.4	20.0	20.6	22.0	24.2	28.6
	1-FN$_{0.10}$	81.7	82.3	82.0	82.2	**82.3**	81.2	10.5	11.0	11.3	12.1	**13.3**	16.9
	1-FN$_{0.20}$	85.3	84.8	84.6	84.1	83.4	80.8	15.2	15.8	16.3	17.2	18.7	22.4
	1-FN$_{0.30}$	86.7	85.5	85.1	84.0	82.9	80.0	20.8	21.5	22.1	23.2	25.1	29.1
	ICF	76.0	77.0	77.0	77.2	73.3	70.3	29.8	31.9	33.0	34.4	35.1	34.3
	DROP3	75.1	79.2	78.4	77.5	74.2	71.5	12.6	17.1	19.9	23.8	28.0	31.5
	CHC	71.3	71.0	69.8	66.7	65.6	66.5	11.0	12.0	11.9	11.8	13.2	16.2

Table 3: Average results when 20 % of noise is added to the dataset. Each column depicts the number of partitions used in the configuration. Normalised results (%) of the different algorithms are obtained referring to the ALL method with the same k value. Bold values represent the non-dominated elements for each p configuration.

the higher noise ratio. Main remarks, hence, are assumed to be the same than is the previous case.

k	Algorithm	Accuracy (%)						Distances (%)					
		1	3	5	10	20	50	1	3	5	10	20	50
1	ALL	63.4	63.4	63.4	63.4	63.4	63.4	100.0	100.0	100.0	100.0	100.0	100.0
	ED	**87.2**	86.0	**85.6**	**84.7**	**84.0**	82.4	**54.3**	52.3	**51.0**	**48.8**	**45.9**	41.3
	RED	**87.0**	**86.1**	**85.6**	**84.6**	**84.0**	82.7	**53.9**	**51.9**	**50.4**	**48.2**	**45.2**	40.7
	CNN	55.8	54.9	54.8	55.2	55.2	56.3	73.8	74.8	75.5	76.7	78.4	81.4
	RCNN	83.1	80.8	80.3	79.8	80.0	79.4	14.2	15.9	16.8	17.9	18.8	19.6
	FCNN	55.7	54.6	54.6	54.5	54.7	55.9	72.7	73.5	74.0	75.2	76.8	79.4
	RFCNN	82.7	80.9	80.7	79.7	79.8	78.9	14.1	15.7	16.5	17.5	18.0	18.7
	1-NE$_{0.10}$	81.5	81.1	**80.6**	**80.3**	78.7	74.1	11.0	11.3	**11.6**	**12.3**	13.6	17.3
	1-NE$_{0.20}$	83.7	82.3	81.9	80.2	77.8	72.4	16.3	16.6	16.9	17.9	19.6	23.3
	1-NE$_{0.30}$	79.9	78.3	77.4	75.1	72.4	69.3	23.1	23.4	23.9	24.9	26.7	30.6
	1-FN$_{0.10}$	**81.8**	80.8	80.6	79.7	78.4	73.9	**11.3**	11.6	11.9	12.6	13.8	17.4
	1-FN$_{0.20}$	83.5	82.3	80.9	79.3	76.8	72.0	16.7	17.1	17.4	18.3	19.9	23.5
	1-FN$_{0.30}$	79.7	77.5	76.5	74.3	72.0	68.8	23.5	24.0	24.5	25.5	27.2	30.8
	ICF	69.1	71.5	71.7	72.1	70.9	68.0	27.5	28.9	29.9	31.6	33.1	34.2
	DROP3	72.7	73.0	73.3	72.0	70.7	68.1	18.6	21.1	22.5	24.9	27.0	29.6
	CHC	**67.3**	**69.2**	**69.2**	**67.6**	**66.1**	64.9	**7.6**	**8.1**	**8.5**	**9.7**	**11.5**	14.1
3	ALL	75.1	75.1	75.1	75.1	75.1	75.1	100.0	100.0	100.0	100.0	100.0	100.0
	ED	**87.6**	**86.4**	**85.9**	**84.9**	**84.3**	**82.8**	**57.8**	**55.4**	**53.8**	**51.1**	**47.5**	**42.2**
	RED	**87.3**	86.1	**85.9**	84.7	**83.7**	**82.8**	**57.2**	54.5	**52.6**	49.1	**45.1**	**39.8**
	CNN	54.2	53.2	53.6	54.0	54.5	55.8	71.2	72.6	73.5	75.2	77.2	80.7
	RCNN	83.9	82.4	82.0	80.9	81.2	79.6	13.8	15.6	16.5	17.7	18.5	19.5
	FCNN	55.7	54.7	54.4	54.4	54.8	55.9	72.7	73.4	74.0	75.2	76.8	79.4
	RFCNN	83.2	82.0	82.3	81.1	81.0	79.5	13.8	15.5	16.3	17.2	17.7	18.4
	1-NE$_{0.10}$	81.6	81.1	80.5	80.0	78.6	**74.7**	10.9	11.2	11.5	12.2	13.5	**17.2**
	1-NE$_{0.20}$	83.7	82.4	81.8	79.9	77.3	72.8	16.2	16.5	16.9	17.8	19.4	23.2
	1-NE$_{0.30}$	80.6	79.1	77.7	75.2	72.7	69.2	23.0	23.3	23.8	24.8	26.7	30.4
	1-FN$_{0.10}$	81.8	81.1	80.6	79.9	78.3	73.8	11.2	11.5	11.8	12.5	13.7	17.3
	1-FN$_{0.20}$	83.6	82.3	81.1	79.4	76.9	72.2	16.6	17.0	17.4	18.3	19.8	23.3
	1-FN$_{0.30}$	80.2	77.8	76.9	74.4	72.6	68.5	23.4	24.0	24.4	25.4	27.1	30.7
	ICF	70.2	71.7	71.4	70.1	67.9	65.8	29.0	29.7	30.3	31.1	31.4	30.7
	DROP3	71.1	72.0	71.4	70.5	68.1	65.9	15.6	19.0	20.7	23.7	26.0	28.5
	CHC	64.6	65.3	65.1	62.7	59.8	**60.0**	8.8	9.4	9.6	10.3	12.3	**15.0**
5	ALL	85.2	85.2	85.2	85.2	85.2	85.2	100.0	100.0	100.0	100.0	100.0	100.0
	ED	**87.9**	86.4	85.9	84.8	84.3	82.4	**60.2**	57.6	55.6	52.5	48.4	42.3
	RED	87.2	85.9	85.7	84.1	83.7	**82.3**	59.6	56.3	54.0	49.8	45.2	**39.6**
	CNN	51.2	51.2	51.2	52.6	53.3	54.9	67.1	69.2	70.7	73.2	76.0	80.1
	RCNN	**84.3**	82.7	82.2	81.3	**81.7**	80.1	**13.7**	15.5	16.3	17.5	**18.1**	19.2
	FCNN	55.7	54.6	54.4	54.4	54.7	56.0	72.7	73.4	74.0	75.2	76.7	79.5
	RFCNN	**84.3**	**83.1**	**82.6**	**81.5**	**81.5**	**80.2**	**13.6**	**15.5**	**16.2**	**17.0**	**17.2**	**18.0**
	1-NE$_{0.10}$	81.6	**81.2**	80.6	80.2	**79.0**	74.7	10.8	**11.1**	11.4	12.1	**13.4**	17.1
	1-NE$_{0.20}$	83.7	82.3	81.6	80.1	77.5	73.0	16.1	16.4	16.7	17.7	19.3	23.1
	1-NE$_{0.30}$	80.2	78.5	77.4	75.0	73.0	68.9	22.9	23.2	23.7	24.7	26.5	30.3
	1-FN$_{0.10}$	81.7	80.8	80.4	79.7	78.5	74.2	11.1	11.4	11.7	12.4	13.6	17.2
	1-FN$_{0.20}$	83.6	81.9	80.8	79.4	76.8	71.7	16.5	16.9	17.3	18.1	19.7	23.3
	1-FN$_{0.30}$	79.6	77.4	76.9	74.5	72.0	68.8	23.3	23.8	24.3	25.2	27.0	30.6
	ICF	71.3	72.5	71.9	69.9	66.5	64.8	31.7	32.2	32.5	32.8	32.7	31.0
	DROP3	71.3	74.0	72.4	70.9	66.8	64.9	15.0	18.9	21.0	24.3	27.2	29.7
	CHC	63.8	63.0	61.9	59.1	58.8	58.8	10.2	10.5	10.6	11.1	12.7	15.2

Table 4: Average results when 40 % of noise is added to the dataset. Each column depicts the number of partitions used in the configuration. Normalised results (%) of the different algorithms are obtained referring to the ALL method with the same k value. Bold values represent the non-dominated elements for each p configuration.

5. Discussion

This section is devoted to thoroughly study and discuss the experimentation results presented in Section 4. The main intention is to analyse the limitations in the studied distributed scheme and draw insights about possible improvements to be tackled.

For a clear and compact discussion, we approach the *estimated profit per prototype* as the ratio between the achieved classification accuracy and the number of distances computed. These figures are shown in Tables 5, 6, and 7 for the unaltered set, the 20% noise ratio, and the 40% noise ratio scenarios respectively. For each noise situation and PS algorithm, only the k configuration achieving the best average profit along the number of partitions is included.

Checking the reported profit figures, independently of the induced noise ratios, two main trends may be highlighted: while ED and RED methods depict an increase in the profit ratio as data is more distributed, most of the other techniques show the opposite tendency. The main reason for this to happen is that, as reported in Tables 2, 3 and 4, while editing approaches show a relatively stable classification accuracy along all the possible distributed configurations, their reduction rate is increased as data is progressively partitioned. On the other hand, most of the other methods depict the inverse tendency: while also showing a relatively stable classification accuracy, the resulting set size is increased, probably because of the naïve merging function which simply gathers the information from each partition without any further consideration.

In a more detailed analysis, for the case without induced noise, 1-NE$_{0.10}$ shows the best profit for all p scenarios considered given its good performance in terms of reduction and classification accuracy, result which is maintained in the case of 20 % of induced noise. Nevertheless, for the case of 40 % of induced noise, CHC achieves the profit for configurations $p = 1, 3, 5,$ and 50 while in the remaining cases the algorithm achieving the best score is again 1-NE$_{0.10}$.

In terms of the average profit, it must be pointed out that although both 1-NE$_{0.10}$ and CHC always report the best average profit scores, they also depict the highest dispersion figures. In general, it can be observed that high average profits are paired with elevated dispersion values while algorithms with low average profit values show a relatively steady behaviour as data is increasingly distributed (low dispersion). In these terms, RFCNN acquires special interest as it shows a good compromise between the profit and dispersion figures.

Figures 2, 3 and 4 present the results obtained facing classification accuracy against resulting set size for the unaltered training set, the 20% noise ratio, and the 40% noise ratio scenarios respectively. For a better comprehension, only the obtained non-dominated values (those highlighted in Tables 2, 3 and 4) for partitions $p = 1$, 5, 20, and 50 are shown as they sufficiently represent the general behaviour observed. The intention with these graphs is to depict at a glance the effects of data distribution in PS strategies.

When no noise is induced (Fig. 2), the clouds representing points for $p = 1$ and $p = 5$ are completely merged, slightly differing depending on the specific performance of each PS algorithm. For $p = 20$ and $p = 50$, however, clouds are shifted towards the right side of the graph in which, although accuracy is maintained, reduction rate is lower. On the other hand, when noise is induced, clouds become more separate. For instance, in the case of the 40 % noise ratio, points move towards the lower right corner (accuracy decrease and larger set size) as data is increasingly distributed. It should be noted that, due to the confusion introduced by this high induced noise rate, only few points from $p = 50$ (more specifically, when using the RFCNN algorithm) achieve a relatively similar performance to the $p = 1$ and $p = 5$ situations.

For the above mentioned, the initial idea of PS algorithms not being prepared for distributed scenarios is clearly supported. Nevertheless, these figures also show that some particular PS techniques cope better than others with these distributed scenarios, somehow suggesting them as possible candidates from which new algorithms may be built up. Particularly, in terms of the profit figures, algorithms 1-NE$_{0.10}$ and RFCNN stand as the ones achieving the best profit figure and the best robustness to the number of partitions among all methods considered, respectively.

5.1. Statistical significance analysis

In order to provide a compact interpretation, we perform several restrictions in terms of the total number of metrics and algorithms considered.

Algorithm	k	p=1	p=3	p=5	p=10	p=20	p=50	Average
ALL	1	0.93	0.93	0.93	0.93	0.93	0.93	0.93 ± 0.00
ED	5	0.97	1.00	1.03	1.09	1.17	1.35	1.10 ± 0.10
RED	5	0.96	1.00	1.04	1.11	1.22	1.41	1.12 ± 0.11
CNN	1	4.13	3.20	2.84	2.39	2.03	1.65	2.71 ± 0.62
RCNN	5	5.49	4.63	4.31	3.95	3.73	3.53	4.27 ± 0.49
FCNN	3	4.15	3.27	2.90	2.47	2.11	1.74	2.77 ± 0.60
RFCNN	5	5.50	4.61	4.32	4.05	3.94	3.84	4.38 ± 0.41
1-NE$_{0.10}$	5	**10.49**	**9.97**	**9.53**	**8.56**	**7.40**	**5.48**	**8.57** ± 1.25
1-NE$_{0.20}$	5	9.01	7.94	7.33	6.44	5.58	4.29	6.77 ± 1.18
1-NE$_{0.30}$	1	7.13	6.01	5.56	4.89	4.23	3.38	5.20 ± 0.94
1-FN$_{0.10}$	5	9.08	8.51	8.29	7.76	7.00	5.45	7.68 ± 0.91
1-FN$_{0.20}$	5	6.57	6.01	5.78	5.45	4.99	4.10	5.48 ± 0.62
1-FN$_{0.30}$	5	4.81	4.38	4.25	4.01	3.72	3.19	4.06 ± 0.40
ICF	5	3.61	2.94	2.66	2.39	2.16	2.02	2.63 ± 0.40
DROP3	3	6.28	4.86	4.21	3.49	2.93	2.43	4.03 ± 0.96
CHC	1	8.38	7.72	7.40	7.29	6.52	5.03	7.06 ± 0.84

Table 5: Ratio between classification accuracy and number of distances computed over the dataset without induced noise. For each algorithm, only the best average value among their different k configurations is included. Last column shows this value together with its dispersion ($\frac{\max - \min}{4}$). Bold values highlight the best profit results per column.

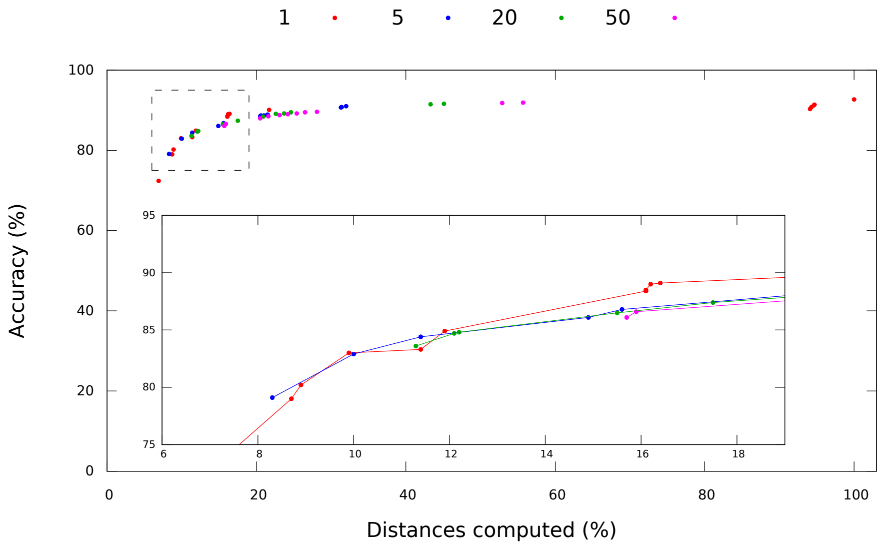

Figure 2: Distances computed (%) against accuracy achieved by the non-dominated schemes of each partition. Average results when no noise is added to the samples.

Algorithm	k	p=1	p=3	p=5	p=10	p=20	p=50	Average
ALL	5	0.90	0.90	0.90	0.90	0.90	0.90	0.90 ± 0.00
ED	1	1.26	1.30	1.32	1.37	1.46	1.63	1.39 ± 0.08
RED	1	1.26	1.30	1.33	1.38	1.50	1.73	1.42 ± 0.11
CNN	5	1.21	1.13	1.09	1.04	1.00	0.95	1.07 ± 0.05
RCNN	5	5.98	5.02	4.76	4.39	4.11	3.91	4.69 ± 0.28
FCNN	3	1.17	1.12	1.10	1.06	1.02	0.98	1.07 ± 0.04
RFCNN	5	5.99	5.04	4.78	4.50	4.32	4.20	4.80 ± 0.21
1-NE$_{0.10}$	5	**9.12**	**8.86**	**8.41**	**7.78**	**6.84**	**5.01**	**7.67** ± 0.96
1-NE$_{0.20}$	3	6.28	6.04	5.79	5.38	4.75	3.70	5.32 ± 0.59
1-NE$_{0.30}$	1	4.42	4.27	4.10	3.78	3.38	2.75	3.78 ± 0.38
1-FN$_{0.10}$	1	8.43	8.15	7.82	7.28	6.53	4.97	7.20 ± 0.79
1-FN$_{0.20}$	1	5.65	5.42	5.22	4.92	4.46	3.58	4.88 ± 0.46
1-FN$_{0.30}$	5	4.05	3.87	3.75	3.51	3.20	2.66	3.51 ± 0.30
ICF	3	2.65	2.53	2.42	2.28	2.15	2.05	2.35 ± 0.12
DROP3	5	5.96	4.63	3.94	3.32	2.77	2.37	3.83 ± 0.56
CHC	1	8.94	8.49	8.32	7.44	6.11	4.83	7.35 ± 0.92

Table 6: Ratio between classification accuracy and number of distances computed over the dataset with a 20 % of induced noise. For each algorithm, only the best average value among their different k configurations is included. Last column shows this value together with its dispersion ($\frac{\max - \min}{4}$). Bold values highlight the best profit results per column.

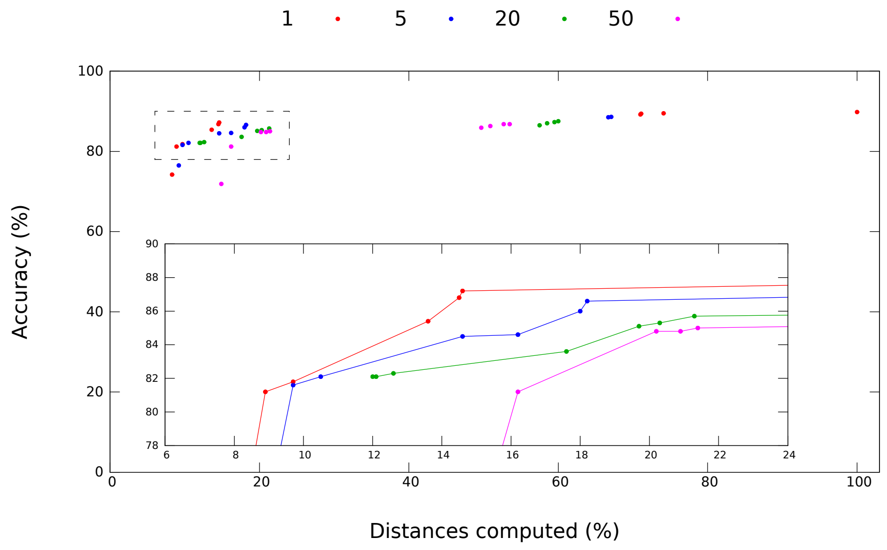

Figure 3: Distances computed (%) against accuracy achieved by the non-dominated schemes of each partition. Average results when 20 % of noise is added to the samples.

Algorithm	k	p=1	p=3	p=5	p=10	p=20	p=50	Average
ALL	5	0.85	0.85	0.85	0.85	0.85	0.85	0.85 ± 0.00
ED	1	1.61	1.64	1.68	1.74	1.83	2.00	1.75 ± 0.09
RED	1	1.61	1.66	1.70	1.76	1.86	2.08	1.78 ± 0.11
CNN	5	0.76	0.74	0.73	0.72	0.71	0.69	0.72 ± 0.01
RCNN	5	6.15	5.34	5.04	4.65	4.51	4.17	4.98 ± 0.29
FCNN	1	0.77	0.75	0.74	0.72	0.71	0.70	0.73 ± 0.01
RFCNN	5	6.20	5.36	5.10	4.79	4.74	4.46	5.11 ± 0.23
1-NE$_{0.10}$	5	8.16	7.88	7.53	**7.04**	**6.17**	4.47	6.88 ± 0.85
1-NE$_{0.20}$	5	5.20	5.02	4.90	4.53	3.99	3.12	4.46 ± 0.48
1-NE$_{0.30}$	3	3.42	3.30	3.17	2.94	2.65	2.19	2.94 ± 0.28
1-FN$_{0.10}$	3	7.87	7.51	7.26	6.77	6.03	4.39	6.64 ± 0.78
1-FN$_{0.20}$	3	5.04	4.84	4.66	4.34	3.86	3.03	4.29 ± 0.45
1-FN$_{0.30}$	3	3.30	3.14	3.04	2.83	2.58	2.15	2.84 ± 0.25
ICF	1	2.51	2.47	2.40	2.28	2.16	2.14	2.33 ± 0.08
DROP3	3	4.75	3.92	3.45	2.97	2.62	2.31	3.34 ± 0.40
CHC	1	**8.86**	**8.54**	**8.14**	6.97	5.75	**4.60**	**7.14** ± 0.99

Table 7: Ratio between classification accuracy and number of distances computed over the dataset with a 40 % of induced noise. For each algorithm, only the best average value among their different k configurations is included. Last column shows this value together with its dispersion ($\frac{\max - \min}{4}$). Bold values highlight the best profit results per column.

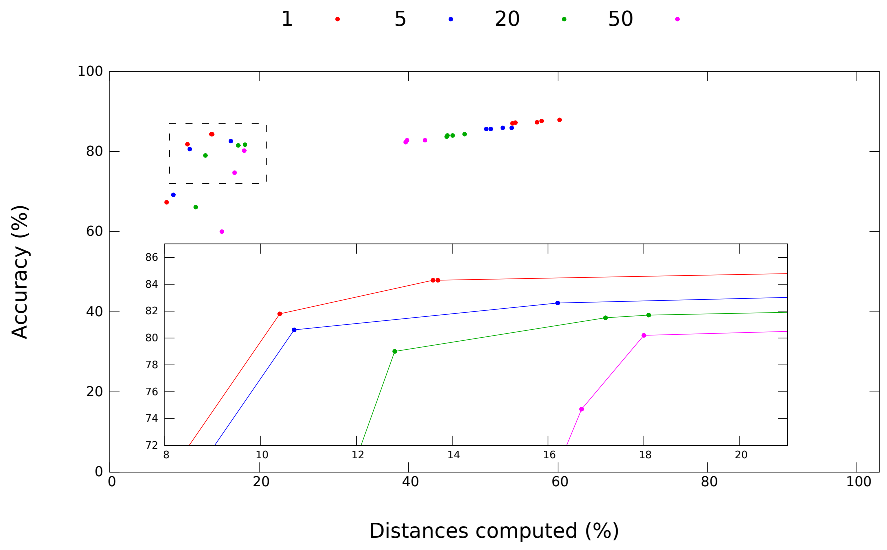

Figure 4: Distances computed (%) against accuracy achieved by the non-dominated schemes of each partition. Average results when 40 % of noise is added to the samples.

On the one hand, we focus on the proposed profit per prototype metric which properly summarises the classification accuracy achieved and the total number of distances computed in one single measure.

On the other hand, as some algorithms are clearly not competitive in terms of their profit, we restrict ourselves to the ones achieving the best average results of this metric for each noise configuration. In this regard we have selected the 1-NE$_{0.10}$, 1-FN$_{0.10}$, and CHC algorithms. Figure 5 shows graphically the evolution of the profit per prototype measure for these PS algorithms and for each noise figure as the number of partitions is increased.

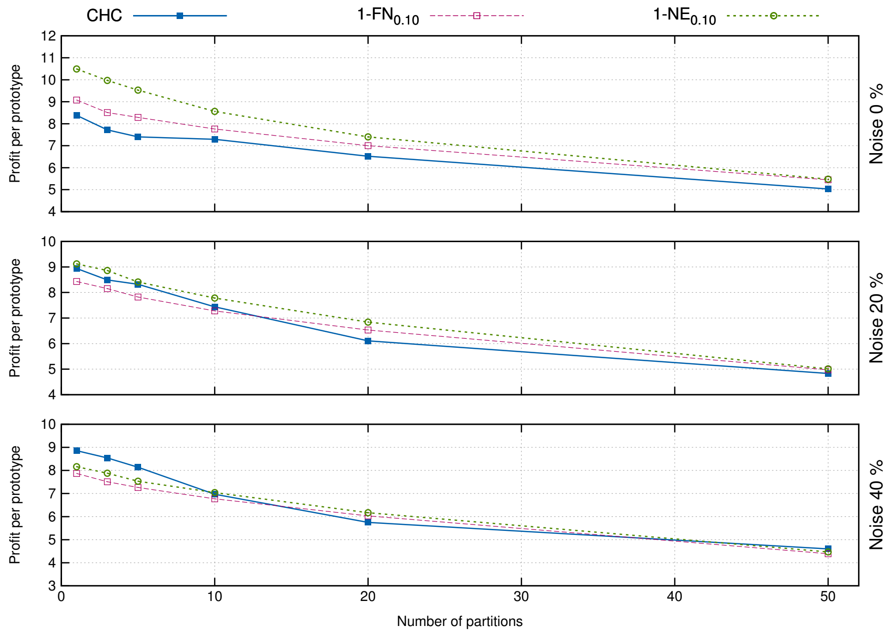

Figure 5: Evolution of the profit per prototype metric of the CHC, 1-NE$_{0.10}$, and 1-FN$_{0.10}$ algorithms as the number of partitions is increased for each noise configuration.

As expected, the tendencies depicted in Fig. 5 show a clear deterioration in the profit per prototype measure considered. It can be checked that this degradation curve is steeper for the initial partitioning of the data than for the latter ones. This profit loss makes sense since the ideal situation of gathering all information in one node ($p = 1$) moves to a distributed scheme ($p > 1$)

for which the PS algorithms are clearly not prepared. Furthermore, it can be checked that this deterioration tendencies become sharper as the induced noise rate is increased. Although the three selected algorithms are in general capable of dealing with noisy situations in non-distributed scenarios, the data partitioning proposed in this case seems to remarkably affect them.

For a rigorous and comprehensive interpretation of the results we now evaluate their significance by means of a statistical analysis. Particularly, we consider the use of the Wilcoxon rank-sum test [42] for a pairwise comparison of the aforementioned methods. In order to statistically compare the general performance of the aforementioned schemes in a distributed scenario, the profit value obtained for each number of partitions considered (p) constitutes a sample of the distribution for the test. The results obtained for this analysis when considering a significance value of p-value < 0.05 can be checked in Table 8.

	Noise 0 %			Noise 20 %			Noise 40 %		
	CHC	1-FN$_{0.10}$	1-NE$_{0.10}$	CHC	1-FN$_{0.10}$	1-NE$_{0.10}$	CHC	1-FN$_{0.10}$	1-NE$_{0.10}$
CHC	–	✗	✗	–	=	✗	–	=	=
1-FN$_{0.10}$	✓	–	✗	=	–	✗	=	–	✗
1-NE$_{0.10}$	✓	✓	–	✓	✓	–	=	✓	–

Table 8: Statistical significance analysis of the profit per prototype metric of the CHC, 1-NE$_{0.10}$, and 1-FN$_{0.10}$ algorithms for each noise configuration. Symbols ✓, ✗, and = state that the profit obtained by the methods in the rows is significantly higher, lower or not different to the profit obtained by the configurations in the columns. Symbol − is used for the case in which a particular method is compared to itself. Significance has been set to p-value < 0.05.

The first outcome we gather from these results is that 1-NE$_{0.10}$ is the algorithm which significantly achieves the best profit figures. Except for the CHC case in the 40 % of induced noise scenario in which both algorithms achieve statistically similar results, the 1-NE$_{0.10}$ method consistently improves the others.

CHC, nevertheless, does not show such a good performance. The statistical analysis shows that this algorithm is not able to improve any of the other methods. This is quite reasonable since, in spite of achieving some of the best reduction rates, it also suffers from remarkable accuracy drops as the data is more distributed.

Finally, 1-FN$_{0.10}$ may be seen as a slightly better candidate than CHC.

It generally shows significantly lower or similar profit figures than the other algorithms. However, for the particular case of when it is compared to the CHC algorithm in the scenario without induced noise, this rank method significantly improves the evolutionary algorithm.

6. Conclusions

This work presents an experimental study on the behaviour of PS methods in a simulated large-scale distributed scenario. Due to the impossibility of processing the large amount of data that this kind of scenarios pose, information has to be distributed and processed separately. For addressing this issue, we have presented a straightforward approach in which the initial data is split into several subsets, each of them processed using some PS technique. The resulting reduced sets are mixed following a *merging function* afterwards. In our case, this function simply includes all the gathered reduced sets in a single one, with no further processing.

A series of PS algorithms, comprising both conventional and advanced methods, have been tested with seven datasets considering several numbers of distributions and under different induced noise scenarios. Not surprisingly, as data distribution increases, algorithms noticeably decrease their performance both in terms of classification accuracy and reduction rate. This fact reinforces the initial premise of current PS algorithms not being prepared for distributed scenarios.

Nevertheless, two particular algorithms should be highlighted after these experiments. On the one hand, ranking methods, and particularly 1-NE, proved to be a very interesting option when considering a compromise between classification accuracy and the number of distances computed. On the other hand, RFCNN may be reported as one of the most reliable techniques among the considered ones due to its remarkable robustness in classification as the data in progressively distributed.

It must be noted that the naïve distributed approach proposed represents the most unfavourable situation possible: a plain hierarchy in which data is distributed in a number of partitions, no communication is allowed between them. Although results could be improved by incorporating more complex structures, the key point here is that classic PS algorithms cannot cope with these cases as they were created for situations in which all data is available.

Furthermore, note that these conclusions have been drawn from a simulated scenario, with a small number of partitions at maximum. Thus, extrapolating to a real framework, sharper tendencies are expected to be observed.

A possible research line for further developing and improving the present work would be to consider the use of more complex merging policies. The proposed naïve gathering approach may produce the inclusion of redundant information coming from the different subsets, thus unnecessarily increasing the set size. More complex proposals, as for instance performing PS over the gathered set, could be considered.

More complex hierarchies for the information distribution should also be considered. As an example, including more levels for progressively gathering the information, which might also be tested with different merging policies as well, could lead to better results.

Incremental learning in this context also represents a relevant topic to tackle. Given an initial basic model, it seems interesting to study how to integrate new knowledge drawn from the incoming data.

Finally, as aforementioned, addressing the design of new PS paradigms for such distributed approaches and thus avoiding classic schemes would be a promising line to pursue.

Acknowledgements

This work was partially supported by the Spanish Ministerio de Educación, Cultura y Deporte through a FPU Fellowship (AP2012–0939), Vicerrectorado de Investigación, Desarrollo e Innovación de la Universidad de Alicante through

FPU program (UAFPU2014–5883) and the Spanish Ministerio de Economía y Competitividad through Project TIMuL (No. TIN2013-48152-C2-1-R, supported by UE FEDER funds).

[1] X. Wu, X. Zhu, G.-Q. Wu, W. Ding, Data mining with big data, IEEE Trans. on Knowl. and Data Eng. 26 (1) (2014) 97–107. `doi:10.1109/TKDE.2013.109`.

[2] W. Fan, A. Bifet, Mining big data: Current status, and forecast to the future, SIGKDD Explor. Newsl. 14 (2) (2013) 1–5. `doi:10.1145/2481244.2481246`.

[3] M. Buhrmester, T. Kwang, S. D. Gosling, Amazon's mechanical turk a new source of inexpensive, yet high-quality, data?, Perspectives on psychological science 6 (1) (2011) 3–5.

[4] E. Dumbill, Making Sense of Big Data, Big Data 1 (1) (2013) 1–2. `doi:10.1089/big.2012.1503`.

[5] S. García, J. Luengo, F. Herrera, Data Preprocessing in Data Mining, Vol. 72 of Intelligent Systems Reference Library, Springer, 2015. `doi:10.1007/978-3-319-10247-4`.

[6] I. H. Witten, E. Frank, M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd Edition, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[7] T. M. Mitchell, Machine Learning, 1st Edition, McGraw-Hill, Inc., New York, NY, USA, 1997.

[8] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, 2nd Edition, John Wiley & Sons, New York, NY, 2001.

[9] D. Wilson, T. Martinez, Reduction techniques for instance-based learning algorithms, Machine Learning 38 (3) (2000) 257–286. `doi:10.1023/A:1007626913721`.

[10] L. Nanni, A. Lumini, Prototype reduction techniques: A comparison among different approaches, Expert Syst. Appl. 38 (9) (2011) 11820–11828. doi:10.1016/j.eswa.2011.03.070.

[11] I. Triguero, J. Derrac, S. Garcia, F. Herrera, A taxonomy and experimental study on prototype generation for nearest neighbor classification, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 42 (1) (2012) 86–100. doi:10.1109/TSMCC.2010.2103939.

[12] S. Garcia, J. Derrac, J. R. Cano, F. Herrera, Prototype selection for nearest neighbor classification: Taxonomy and empirical study, Pattern Analysis and Machine Intelligence, IEEE Transactions on 34 (3) (2012) 417–435. doi:10.1109/TPAMI.2011.142.

[13] N. García-Pedrajas, A. de Haro-García, Scaling up data mining algorithms: review and taxonomy, Progress in Artificial Intelligence 1 (1) (2012) 71–87. doi:10.1007/s13748-011-0004-4.

[14] F. Provost, V. Kolluri, A survey of methods for scaling up inductive algorithms, Data Mining and Knowledge Discovery 3 (2) (1999) 131–169. doi:10.1023/A:1009876119989.

[15] J. R. Cano, F. Herrera, M. Lozano, Stratification for scaling up evolutionary prototype selection, Pattern Recognition Letters 26 (7) (2005) 953 – 963. doi:10.1016/j.patrec.2004.09.043.

[16] J. R. Cano, F. Herrera, M. Lozano, On the combination of evolutionary algorithms and stratified strategies for training set selection in data mining, Applied Soft Computing 6 (3) (2006) 323 – 332. doi:10.1016/j.asoc.2005.02.006.

[17] J. Derrac, S. García, F. Herrera, Stratified prototype selection based on a steady-state memetic algorithm: a study of scalability, Memetic Computing (2010) 183–199.

[18] I. Triguero, D. Peralta, J. Bacardit, S. García, F. Herrera, Mrpr: A mapreduce solution for prototype reduction in big data classification, Neurocomputing 150, Part A (0) (2015) 331 – 345. doi:10.1016/j.neucom.2014.04.078.

[19] C. García-Osorio, A. de Haro-García, N. García-Pedrajas, Democratic instance selection: A linear complexity instance selection algorithm based on classifier ensemble concepts, Artificial Intelligence 174 (5–6) (2010) 410 – 441. doi:10.1016/j.artint.2010.01.001.

[20] N. García-Pedrajas, A. de Haro-García, J. Pérez-Rodríguez, A scalable approach to simultaneous evolutionary instance and feature selection, Information Sciences 228 (0) (2013) 150 – 174. doi:10.1016/j.ins.2012.10.006.

[21] A. Haro-García, N. García-Pedrajas, A divide-and-conquer recursive approach for scaling up instance selection algorithms, Data Min. Knowl. Discov. 18 (3) (2009) 392–418. doi:10.1007/s10618-008-0121-2.

[22] X. Wu, X. Zhu, G.-Q. Wu, W. Ding, Data mining with big data, Knowledge and Data Engineering, IEEE Transactions on 26 (1) (2014) 97–107. doi:10.1109/TKDE.2013.109.

[23] N. Natarajan, I. Dhillon, P. Ravikumar, A. Tewari, Learning with noisy labels, in: Advances in Neural Information Processing Systems, 2013, pp. 1196–1204.

[24] P. Hart, The condensed nearest neighbor rule (corresp.), Information Theory, IEEE Transactions on 14 (3) (1968) 515–516. doi:10.1109/TIT.1968.1054155.

[25] D. L. Wilson, Asymptotic Properties of Nearest Neighbor Rules Using Edited Data, Systems, Man and Cybernetics, IEEE Transactions on SMC-2 (3) (1972) 408–421. doi:10.1109/TSMC.1972.4309137.

[26] P. A. Devijver, J. Kittler, Pattern recognition: A statistical approach, Prentice Hall, 1982.

[27] B. V. Dasarathy, J. S. Sánchez, S. Townsend, Nearest neighbour editing and condensing tools-synergy exploitation., Pattern Anal. Appl. (2000) 19–30.

[28] F. Angiulli, Fast Nearest Neighbor Condensation for Large Data Sets Classification, Knowledge and Data Engineering, IEEE Transactions on 19 (11) (2007) 1450–1464.

[29] J. R. Rico-Juan, J. M. Iñesta, New rank methods for reducing the size of the training set using the nearest neighbor rule, Pattern Recognition Letters 33 (5) (2012) 654–660.

[30] 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.

[31] H. Brighton, C. Mellish, On the Consistency of Information Filters for Lazy Learning Algorithms, in: J. Żytkow, J. Rauch (Eds.), Principles of Data Mining and Knowledge Discovery, Vol. 1704 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1999, pp. 283–288.

[32] 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, Indiana, USA, July 15-18 1990., 1990, pp. 265–283.

[33] 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. doi:10.1016/j.asoc.2005.02.006.

[34] J. Hull, A database for handwritten text recognition research, Pattern Analysis and Machine Intelligence, IEEE Transactions on 16 (5) (1994) 550–554. `doi:10.1109/34.291440`.

[35] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-Based Learning Applied to Document Recognition, in: Intelligent Signal Processing, IEEE Press, 2001, pp. 306–351.

[36] L. J. Latecki, R. Lakämper, U. Eckhardt, Shape descriptors for non-rigid shapes with a single closed contour, in: Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2000, pp. 424–429.

[37] 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.

[38] A. Asuncion, D. Newman, UCI Machine Learning Repository (2007). URL `http://www.ics.uci.edu/$\sim$mlearn;http://epository.html`

[39] H. Freeman, On the encoding of arbitrary geometric configurations, Electronic Computers, IRE Transactions on EC-10 (2) (1961) 260–268. `doi:10.1109/TEC.1961.5219197`.

[40] R. A. Wagner, M. J. Fischer, The string-to-string correction problem, J. ACM 21 (1) (1974) 168–173. `doi:10.1145/321796.321811`.

[41] H. Sakoe, S. Chiba, Readings in Speech Recognition, in: A. Waibel, K.-F. Lee (Eds.), Readings in Speech Recognition, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990, Ch. Dynamic Programming Algorithm Optimization for Spoken Word Recognition, pp. 159–165.

[42] J. Demsar, Statistical Comparisons of Classifiers over Multiple Data Sets, Journal of Machine Learning Research 7 (2006) 1–30.