Very fast variations of training set size reduction algorithms for instance-based classification

STEFANOS OUGIAROGLOU*, International Hellenic University, Greece
GEORGIOS EVANGELIDIS, University of Macedonia, Greece

Reduction through Homogeneous Clustering (RHC) and its editing variant (ERHC) are effective data reduction techniques for the k-NN classifier. They are based on an iterative k-means clustering task that discovers homogeneous clusters. The centers of the resulting homogeneous clusters constitute the instances of the reduced training set. Although RHC and ERHC are quite fast compared to several well-known data reduction techniques, the iterative execution of k-means clustering renders both of them inappropriate for data reduction tasks that need to be performed quickly, especially, when run over large training datasets. The present paper proposes simple and very fast variations of the algorithms, which are appropriate for such environments. The variations are called RHC2 and ERHC2 and replace the complete execution of k-means clustering with a fast task that assigns instances to the class centers. The experimental study based on fourteen datasets, and, the corresponding statistical tests, show that the proposed RHC2 and ERHC2 variations are very fast and, at the cost of a small penalty on classification accuracy, they achieve higher reduction rates than their predecessors and other two well-known data reduction techniques. They are good candidates when fast reduction on large datasets is required.

CCS Concepts: • **Information systems → Nearest-neighbor search**; • **Computing methodologies → Instance-based learning**.

Additional Key Words and Phrases: data reduction, prototype generation, RHC, homogeneous clusters, k-NN Classification

ACM Reference Format:
Stefanos Ougiaroglou and Georgios Evangelidis. 2023. Very fast variations of training set size reduction algorithms for instance-based classification. In . ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3589462.3589493

1 INTRODUCTION

Handling large volumes of training data in k-Nearest Neighbors (k-NN) classification [8] is a CPU intensive task. Contrary to eager classifiers, which use the training data only to build a classification model, lazy classifiers like k-NN use the available training data as the classification model that is examined whenever an unclassified instance needs to be classified. In particular, for each unclassified instance x, k-NN classifier retrieves the k nearest to x neighbours by computing all distances between x and the available training data. Then, x is classified to the most common class of its nearest neighbors. Therefore, using exhaustive search for finding the nearest neighbours is inappropriate for large training datasets.

Indexing methods, such as k-dimensional trees [5], ball-trees [18, 27], etc., can avoid exhaustive searches. Nevertheless, their use becomes problematic when they are applied on high-dimensional training data [28]. A dimensionality reduction method can remedy this problem, however, it may result in information loss and the classification performance may deteriorate. Data (or instance) reduction techniques (DRTs) [13, 26] do not alter the dimensional space.

*Both authors contributed equally to this research.

Manuscript submitted to ACM

They pre-process the original training dataset and build a small set, which is called condensing set (CS) that represents as much as possible the original one.

DRTs belong to two major categories: (a) Prototype Selection (PS) [13] algorithms and (b) Prototype Generation (PG) [26] algorithms. PS algorithms retain some representative instances (prototypes) in CS. In contrast, PG algorithms create groups with similar instances and generate prototypes for each group. Both categories consider that training instances that are far from the class decision boundaries are useless and can be removed without penalty in accuracy. Thus, PS and PG algorithms try to retain or generate prototypes for the close to the decision boundaries data areas.

Reduction through Homogeneous Clustering (RHC) [19] is an effective PG algorithm. It is based on an iterative k-means clustering [29] task that discovers homogeneous clusters. The centers of the discovered homogeneous clusters play the role of prototypes and are stored in CS. The Editing and Reduction through Homogeneous Clustering (ERHC) algorithm [20] is a simple variation of RHC that can handle noisy training data. It works similar to RHC but does not generate prototypes for single instance clusters. The idea is that single instance clusters constitute noise and are removed. RHC and ERHC are considered fast PG algorithms that achieve high reduction rates either with gains or, at least, without loss in classification accuracy [19, 20].

Although RHC and ERHC are quite fast compared to other well-known PS and PG algorithms [19, 20], the iterative k-means clustering task renders both inappropriate for fast data reduction tasks, especially when they run over large training datasets. This is the motive behind the present work. More specifically, the motivation is to examine if already efficient DRTs can be modified in order to be able to be applied in environments where very fast data reduction tasks are needed. The contribution of this paper is the development of new very fast RHC and ERHC variations. They are called RHC2 and ERHC2 and are appropriate for fast data reduction tasks since they replace the costly execution of the complete k-means clustering by a simpler and faster procedure for assigning the training instances to the class centers. The experimental study shows that RHC2 and ERHC2 are much faster than their ancestors and other two well-known PS algorithms [2, 14] with small penalty in classification accuracy.

The rest of the paper is organized as follows: Section 2 presents the recent related work in the field of prototype generation. Section 3 reviews the RHC and ERHC algorithms. Section 4 presents the new RHC2 and ERHC2 variations. Section 5 presents the experimental study and the results of the statistical tests. Section 6 concludes the paper and gives directions for future work.

2 RELATED WORK

Prototype Generation is motivated by the explosion of Big Data and, therefore, it constitutes an active research field. This section reviews the recent PG algorithms published after 2012. Triguero et al. [26] review PG algorithms introduced prior to 2012 and present a taxonomy and an experimental study.

Impedovo et al. [16] propose a handwriting digit recognition PG algorithm that comprises of two stages. The first stage uses the Adaptive Resonance Theory [7] to determine the number of prototypes and generate the initial prototypes. The second stage uses a naive evolution strategy in order to generate the final condensing set. The algorithm is an incremental approach and modifies the previously generated prototypes or adds new prototypes in the condensing set. In practice, the proposed algorithm is adaptive to writing style changes.

A swarm-based metaheuristic search algorithm is presented by Rezaei and Nezamabadi-pour in [22]. The algorithm is inspired by motion and gravity Newtonian laws [21] and is adapted for prototype generation.

Hu and Tan [15] present two methods for evolutionary computation based prototype generation. The methods improve the performance of k-NN classification. The first method upgrades the k-NN classifier's generation ability

by considering the misclassified instances. The second method avoids over-fitting by pursuing the performance on multiple data subsets. The authors claim that particle swarm optimization that uses the proposed methods achieves better performance.

Elkano et al. in [10] present an one-pass Map-Reduce Prototype Generation algorithm. It is called CHI-PG and exploits Map-Reduce. It uses fuzzy rules to generate prototypes. The prototypes are the same, regardless of the number of Mappers/Reducers used. The algorithm improves the distributed prototype reduction execution time. At the same time, classification reduction rates and accuracy are not negatively affected.

Prototype Generation via Genetic Programming is presented by Escalante et al. in [11]. The proposed algorithm is based on genetic programming. It generates prototypes by maximizing an estimate of the k-NN classifier's performance and by combining training instances through arithmetic operators.

The usage of Dissimilarity space techniques for prototype generation is proposed by Calvo-Zaragoza et al. in [6]. The use of statistical PG algorithms on the original space is allowed by a mapping technique that maps the initial structural representation to a feature-based representation.

A Learning Vector Quantization method based on granular computing that uses Big Data incremental learning mechanisms is presented by Cruz-Vega and Escalante in [9]. This method clusters instances with similar characteristics very fast by utilizing a one-pass clustering task. Then, the method generates prototypes to cover the class distribution. The method has two phases. In the first phase, the number of prototypes is controlled using a usage-frequency indicator and the best prototype is kept using a life index. In the second phase, useless dimensions are pruned.

A prototype generation multi-objective evolutionary technique is proposed by Escalante et al. in [12]. This PG technique targets at improving reduction rate and accuracy and achieving a good trade-off between them. This is achieved by formulating the PG task as a multi-objective optimization problem. The amount of prototypes and the generalization performance estimation that generates prototypes are the key factors in the proposed technique.

A Learning Vector Quantization PG algorithm for time series classification is proposed by Brijnesh J. Jain and David Schultz in [17]. The paper is an extended version of Learning Vector Quantization. It applies a transformation from Euclidean spaces to Dynamic Time Wrapping spaces.

A paper published by Leonardo A. Silva et al. [24] focuses on the number of prototypes generated by PG algorithms. The paper proposes a model that estimates the ideal number of prototypes according to the characteristics of the training data.

The research work conducted by I. Sucholutsky and M. Schonlau and published in [25] focuses on PG algorithms for training data that has complex geometries.

The Condensed Nearest Neighbor (CNN) [14] is the first PS algorithm for condensing data in the literature and is used in many relevant papers for comparison purposes. Thus, we decided to follow the same approach and include CNN in our experimental study. CNN is based on a very simple idea. Training instances whose nearest neighbor belongs to a different class are probably close to decision boundaries. Therefore, they must be included in the condensing set as prototypes. CNN works as follows: Initially, a random training instance moves from the training set (TS) to the condensing set (CS). Then, for each instance x in TS, the algorithm finds and retrieves its nearest neighbor in CS. If the two instances have different class labels, x moves from TS to CS, otherwise, x remains in TS. In this way, the size of CS increases while the size of TS decreases. CNN performs multiple passes over TS trying to move more instances from TS to CS by examining the new content of CS. The algorithm stops when no move is performed during a complete pass of the instances in TS. IB2 [1] is one-pass PS algorithm, practically, the first iteration of CNN. Therefore, it is a quite fast algorithm and it is appropriate to be compared with the proposed RHC2 and ERHC2 algorithms.

Contrary to the proposed RHC2 and ERHC2 algorithms and their ancestors, CNN and IB2 build a different condensing set when examining the same training data in different order. Similarly to the proposed algorithms and their ancestors, CNN and IB2 are parameter-free. They determine the size of the condensing set automatically without any input parameter.

3 THE RHC AND ERHC ALGORITHMS

As mentioned above, RHC [19] belongs to PG algorithms and it is based on k-means clustering. Contrary to many other DRTs, the size of its condensing set is determined without any input parameter. Thus, RHC is a parameter-free algorithm.

Initially, RHC considers the whole training set as a non homogeneous cluster C, i.e., a set of instances that belong to different classes. RHC computes a class center for each class present in C by finding the average value of the attributes of the instances that belong to each class.

For example, if the C instances belong to five classes, the algorithm will compute five class centers. Then, k-means clustering is executed using the computed class representatives as initial means, hence, it discovers five clusters. If a discovered cluster contains instances of only one class (i.e., it is a homogeneous cluster), the cluster center plays the role of prototype and is stored in CS. For all non-homogeneous discovered clusters, the aforementioned procedure is applied recursively. The RHC algorithm stops when all non-homogeneous discovered clusters are split into homogeneous clusters.

RHC produces many and small clusters for the close class border data areas. Consequently, more prototypes are generated for those areas. On the other hand, it discovers large clusters for the "internal" class data areas. Thus, fewer prototypes are generated. Furthermore, the use of the class centers as initial means is a very efficient approach because the probability of discovering large homogeneous clusters is higher and that causes high reduction rates. Another benefit of RHC is that it constructs the same CS regardless the order of the training data. In addition, RHC is quite easy to implement.

The ERHC algorithm is a simple variation of RHC for datasets containing noise. ERHC deals with noise in the training data by discarding single instance (homogeneous) clusters. These instances are probably noise since they are surrounded by instances of other classes.

Since ERHC removes the noise from the training data, it achieves higher reduction rates than RHC. Moreover, ERHC has good performance even when applied on noise-free training datasets.

4 THE PROPOSED RHC2 AND ERHC2 ALGORITHMS

The RHC2 and ERHC2 algorithms work similar to RHC and ERHC, respectively. However, they are much faster because they avoid the complete execution of k-means clustering in each algorithm iteration.

Similarly to RHC, RHC2 initially considers the whole training dataset as a non-homogeneous cluster C and computes a class representative for each class label cl present in C by averaging the cluster's instances that belong to each label. Hence, RHC2 creates as many class representatives, suppose $|n|$, as the number of distinct classes labels in C. Then, RHC2 splits C into $|n|$ clusters by assigning its instances to their closest class representative without taking into account the label of the instances. Whenever a homogeneous cluster is formed, its center is stored in CS as a prototype, labeled by the unique class label present in the cluster. On the other hand, whenever a non-homogeneous cluster is formed, the aforementioned procedure is repeated recursively. A non-recursive implementation of RHC2 is shown in Algorithm 1.

Algorithm 1 RHC2

Input: TS
Output: CS

1: $Q \leftarrow \emptyset$ {Initialize the queue that will hold the clusters to be processed}
2: $CS \leftarrow \emptyset$ {Initialize CS}
3: Enqueue(Q, TS) {the whole TS becomes an unprocessed cluster}
4: **repeat**
5: $C \leftarrow$ Dequeue(Q)
6: **if** homogeneous(C) **then**
7: $rep \leftarrow$ representative of C {average of its instances}
8: $rep.label \leftarrow$ class label of instances in C
9: $CS \leftarrow CS \cup \{rep\}$
10: **else**
11: $numlabels \leftarrow$ number of existing labels in C
12: $reps \leftarrow \{rep_i | rep_i$ a representative of existing label in C, $i = 1 \ldots numlabels\}$
13: $clusters \leftarrow \{cluster_i | cluster_i$ initially empty cluster corresponding to $rep_i\}$
14: **for** each instance $x \in C$ **do**
15: $rep_i \leftarrow$ nearest representative to x
16: $cluster_i \leftarrow cluster_i \cup x$
17: **end for**
18: **for** each $cl \in clusters$ **do**
19: Enqueue(Q, cl)
20: **end for**
21: **end if**
22: **until** IsEmpty(Q)
23: **return** CS

The algorithm takes a training set and returns a condensing set CS. It utilizes a queue, Q, to hold the non homogeneous clusters of training instances. Initially, the complete training set is considered to be a non-homogeneous cluster and is placed in Q (line 1). At each repeat-until iteration, RHC2 dequeues a cluster C from Q (line 5). If it is homogeneous (line 6), its center constitutes a prototype and is placed in CS (line 9). If C is non-homogeneous (line 10), for each class label in C, RHC2 computes a class center or representative (lines 11–12) by averaging the instances of C that belong to that label. If *numlabels* classes are present in C, *numlabels* class centers are computed. The class centers are placed in *reps*. Then, the algorithm splits C into *numlabels* clusters. An initially empty cluster for each class center is created (line 13). Each training instance x, regardless its class label, is assigned to the cluster of the closest class center in *reps* (lines 14–17). Each formed cluster is placed in Q (line 18–20). The repeat-until loop continues until there is no cluster in Q (line 22).

The ERHC2 algorithm splits each non-homogeneous cluster in a similar manner to RHC2. Like ERHC, ERHC2 considers single instance clusters to be noise and discards them. Therefore, in the case of ERHC2, line 19 of Algorithm 1 is applied only to non-single instance clusters, so that, the prototypes created in line 7 from homogeneous clusters represent more than one training instances.

Obviously, RHC2 and ERHC2 replace the computationally costly k-means clustering with the fast procedure that assigns instances to the class centers. In reality, the fast procedure constitutes the first step of k-means clustering. It is worth mentioning that RHC2 and ERHC2 inherit all the RHC and ERHC properties. Therefore, they are parameter-free PG algorithms that produce the same CS regardless the order of the training data.

Table 1. dataset characteristics

Dataset	Instances	Attributes	Classes
Banana (BN)	5300	2	2
Eye State (EEG)	14980	14	2
KDD Cup (KDD)	141481	40	23
Letter Image Recognition (LIR)	20000	16	26
Landsat Satellite (LS)	6435	36	7
Magic Gamma Telescope (MGT)	19020	11	2
Pen Digits (PD)	10992	16	10
Phoneme (PH)	5404	5	2
Pima (PM)	768	8	2
Ring (RNG)	7400	20	2
Shuttle (SH)	58000	9	7
Twonorm (TN)	7400	20	2
Textrue (TXR)	5500	40	11
Yeast (YST)	1484	8	10

5 PERFORMANCE EVALUATION

5.1 Datasets

For the experimental study, fourteen datasets distributed by the KEEL [3] and UCI machine learning [4] repositories were used. The datasets come from different domains and have different characteristics. Their main characteristics are summarized in Table 1. The Euclidean distance was used as distance measure. The attribute values of each dataset were normalized to the range [0,1].

5.2 Experimental Setup

The proposed RHC2 and ERHC2 algorithms as well as their ancestors RHC and ERHC were coded in C++. In addition, for comparison purposes, we conducted experiments for the CNN and IB2 PS algorithms as well as for NOP approach. NOP stands for "No Operation", hence, indicates the fact that no data reduction is performed on the training dataset. The NOP approach is the conventional k-NN classifier that runs over the original training set. We measured the accuracy by running the 1-NN classifier over the original training set (case of NOP) and over the condensing sets generated by the DRTs that took part in the experimental study.

We used a five-fold cross validation schema to measure the metrics of Accuracy (ACC), Reduction Rate (RR), and Distance Computations (DST in thousands) needed for CS construction. The higher the RR, the faster the classification is. Also, the fewer the distances computed for CS construction, the lower the pre-processing computational cost required.

The major goal of the proposed algorithms is to reduce the computational cost required for the condensing set construction at a minimum level. The goals of high reduction rates and keeping the classification accuracy at high levels are also significant.

5.3 Experimental results

Table 2 presents the results of the experimental study. As expected, RHC2 and ERHC2 compute the fewest distances. In most cases, RHC2 and ERHC2 are 10 times faster or more than the fastest competitor.

Table 2. Comparison in terms of ACC(%), RR(%) and DST(in Thousands of distance computations)

Dataset		NOP	RHC	ERHC	RHC2	ERHC2	CNN	IB2
BN	ACC	86.92	83.17	88.09	82.92	87.60	86.38	84.15
	RR	-	79.39	90.36	80.61	91.73	77.36	83.36
	DST	-	564.18	564.18	65.92	65.92	12095.48	1545.14
EEG	ACC	45.62	47.54	46.45	48.31	46.62	47.24	45.11
	RR	-	77.22	85.06	76.89	86.71	66.11	85.94
	DST	-	3870.43	3870.43	258.61	258.61	146134.02	8159.40
KDD	ACC	99.71	99.39	99.45	98.38	98.34	99.66	99.55
	RR	-	99.19	99.46	99.28	99.56	99.12	99.26
	DST	-	81593.69	81593.69	3404.88	3404.88	475607.46	58950.34
LIR	ACC	95.75	93.49	92.74	91.18	89.67	92.66	91.41
	RR	-	88.01	91.95	85.77	91.59	83.12	85.46
	DST	-	29606.11	29606.11	750.10	750.10	166004.67	23375.86
LS	ACC	89.94	88.93	88.69	86.59	87.07	87.75	86.71
	RR	-	90.04	93.10	90.45	93.92	79.92	84.31
	DST	-	1644.57	1644.57	89.63	89.63	19989.68	2294.72
MGT	ACC	80.50	74.79	79.99	71.53	76.90	76.78	74.54
	RR	-	79.38	86.72	80.61	89.10	64.09	73.58
	DST	-	3990.16	3990.16	320.48	320.48	266736.99	31166.01
PD	ACC	99.33	98.50	98.64	97.27	97.20	98.44	97.88
	RR	-	96.48	97.45	96.93	97.92	95.35	96.07
	DST	-	3007.78	3007.78	173.19	173.19	12026.86	1969.99
PH	ACC	89.64	85.06	85.99	82.62	83.40	87.69	84.55
	RR	-	80.74	87.90	80.94	89.29	75.53	80.86
	DST	-	711.31	711.31	71.24	71.24	14306.66	1970.61
PM	ACC	70.54	63.23	67.92	62.45	67.14	65.97	65.07
	RR	-	68.86	81.63	69.67	84.53	49.64	62.31
	DST	-	59.03	59.03	8.50	8.50	387.07	74.09
RNG	ACC	74.55	82.20	86.51	76.51	79.93	82.65	81.68
	RR	-	90.14	92.68	91.05	93.55	73.26	80.79
	DST	-	2184.73	2184.73	95.27	95.27	32015.24	3446.38
SH	ACC	99.93	99.79	99.71	98.12	97.82	99.92	99.91
	RR	-	99.63	99.70	99.66	99.72	99.65	99.66
	DST	-	9672.88	9672.88	745.84	745.84	20206.74	5296.20
TN	ACC	94.70	89.76	91.45	82.53	84.73	89.78	87.43
	RR	-	96.65	97.49	97.91	98.52	82.12	88.28
	DST	-	1534.66	1534.66	47.07	47.07	23191.84	2049.06
TXR	ACC	98.91	97.11	97.02	96.11	95.82	97.13	96.56
	RR	-	94.51	95.94	95.25	96.62	91.96	93.33
	DST	-	2724.36	2724.36	79.84	79.84	5017.76	826.96
YS	ACC	51.58	47.47	52.60	45.51	50.64	49.36	47.33
	RR	-	50.41	78.92	49.89	80.66	33.08	45.63
	DST	-	554.96	554.96	28.23	28.23	1464.26	378.53

The proposed algorithms achieved high reduction rates. Reduction rate measurements are critical in data reduction tasks, since the higher the reduction rate is, the faster the execution of the k-NN classifier is on the produced condensing

Table 3. Results of Wilcoxon signed rank test on ACC measurements

Methods	Accuracy	
	wins/losses	Wilcoxon value
RHC vs ERHC2	10/4	0.551
ERHC vs ERHC2	13/1	0.001
CNN vs ERHC2	10/4	0.026
IB2 vs ERHC2	8/6	0.925
RHC vs RHC2	13/1	0.002
ERHC vs RHC2	13/1	0.004
CNN vs RHC2	13/1	0.002
IB2 vs RHC2	13/1	0.011
ERHC2 vs RHC2	8/6	0.084

set. It is obvious that RHC2 achieves higher reduction rates than RHC and ERHC2 achieves higher reduction rates than ERHC. Also, both proposed algorithms achieve higher reduction rates than IB2 and CNN.

In the BN, PM, MGT and YS datasets, ERHC2 achieves higher classification accuracy than RHC and CNN. Similarly, ERHC2 is more accurate than IB2 when the BN, EEG, LS, MGT, PM and YS datasets are used. Moreover, ERHC2 is more accurate than RHC2 especially on datasets that contain noise.

RHC2 has a greater penalty in classification accuracy. We observe that only in the case of the EEG dataset it achieves higher classification accuracy than RHC, IB2 and CNN. However, in most cases, its classification accuracy is close enough to that of its competitors. ERHC seems to be the most accurate DRT. In many cases, it achieves even higher classification accuracy than NOP.

A last comment is that the experimental results show that RHC2 and ERHC2 are very fast, thus, both are appropriate for the purpose they were designed for.

Furthermore, both RHC2 and ERHC2 algorithms considerably reduce the training dataset. RHC2 has small penalty in classification accuracy while ERHC2 can be as accurate as IB2, RHC and CNN.

5.4 Statistical tests

We conducted a Wilcoxon signed rank test [23] and a Friedman test to complement the experimental study. Both tests are common in the field of PS and PG algorithms.

5.4.1 Wilcoxon Signed Rank Test results. We used the Wilcoxon signed rank test, a non-parametric test, to compare the DRTs in pairs, examining their measurements on each dataset. The test statistically confirms the validity of the descriptive measurements shown in Table 5.

Table 5 clearly demonstrates that RHC2 and ERHC2 computed the fewest distances and achieved the highest RR measurements of all competitors. Therefore, we did not run the Wilcoxon signed rank test to confirm the validity of these comparisons. We used the statistical test only for the ACC measurements.

The results of the Wilcoxon signed rank test are presented in Table 3. The column with header "Wilcoxon" lists the Wilcoxon value. The latter quantifies the significance of the statistical difference between the two DRTs. When it is lower than 0.05, the difference between the two algorithms is considered statistically significant.

The results of the Wilcoxon signed rank test show that the difference between the RHC-ERHC2 and IB2-ERHC2 pairs is not significant. Although ERHC2 beats CNN in four datasets, there is statistically significant difference between

Table 4. Results of Friedman test on ACC, RR and DC measurements

Algorithm	Mean Rank		
	ACC	RR	DST
RHC	4.00	2.79	3.07
ERHC	5.00	4.93	3.07
RHC2	1.71	3.68	5.50
ERHC2	2.64	5.93	5.50
CNN	4.71	1.07	1.00
IB2	2.93	2.61	2.86

them. Therefore, we conclude that the proposed ERHC2 algorithm achieves higher reduction rates by computing an extremely small number of distances compared to IB2 and RHC and, at the same time, classification accuracy is retained as high as that of IB2 and RHC.

5.4.2 Friedman Test results. The Friedman test is also non parametric and is used to rank the DRTs. The best DRT has the highest mean rank, the second best DRT has the second highest mean rank, etc. The Friedman test was run three times, one for the ACC measurements, one for the RR measurements and one for the DST measurements. Table 4 presents the results of the test. As expected, the test confirms that RHC2 and ERHC2 are the fastest approaches. Also, the test confirms that ERHC2 achieves the highest RR measurements and that RHC2 achieves higher RR measurements than RHC, CNN and IB2. Last but not least, the test shows that in terms of accuracy, ERHC2 has almost the same mean rank with IB2.

Table 5. Descriptive statistics of experimental measurements (Average (AVG), Standard Deviation (STDEV), Coefficient of Variation (CV))

Dataset		NOP	RHC	ERHC	RHC2	ERHC2	CNN	IB2
AVG	ACC	84.12	82.17	83.94	80.00	81.63	82.96	81.56
	RR	-	85.05	91.31	85.35	92.39	76.45	82.77
	DST	-	10122.77	10122.77	438.49	438.49	85370.34	10107.38
STD	ACC	17.74	17.88	16.92	17.54	16.52	17.40	17.865
	RR	-	13.68	6.56	13.83	5.87	18.83	14.64
	DST	-	21928.85	21928.85	888.21	888.21	137875.16	16846.86
CV	ACC	21.09	21.75	20.15	21.92	20.24	20.98	21.90
	RR	-	16.09	7.19	16.20	6.36	24.63	17.69
	DST	-	216.63	216.63	202.56	202.56	161.50	166.68

6 CONCLUSIONS AND FUTURE WORK

Computationally costly data reduction tasks are inappropriate in many domains where instance-based classification is applied. This paper presented a preliminary work on the development of very fast data reduction algorithms for instance-based classification.

More specifically, the paper presented the RHC2 and ERHC2 algorithms. Both are very fast variations of the known RHC and ERHC algorithms, respectively.

Contrary to RHC and ERHC, RHC2 and ERHC2 avoid the computationally costly complete k-means clustering procedure that runs over the instances of each non-homogeneous cluster. RHC2 and ERHC2 form as many clusters of

instances as the number of classes in each non-homogeneous cluster by assigning the training instances of the non-homogeneous cluster to their closest class center in the cluster. When a homogeneous cluster is discovered, its center constitutes a prototype and is stored in CS.

The experimental study and the corresponding statistical tests showed that both proposed variations are very fast and achieve higher reduction rates than RHC, ERHC, CNN and IB2, and, at the same time, in many cases their classification accuracy is not negatively affected.

In the future, we plan to design new fast PS and PG algorithms that will be as accurate as state-of-the-art DRTs. Moreover, we plan to apply RHC2 and ERHC2 in fast training data streams and to develop RHC2 and ERHC2 variations that will be able to handle training data streams with concept drift.

REFERENCES

[1] David W. Aha. 1992. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies* 36, 2 (1992), 267–287. https://doi.org/10.1016/0020-7373(92)90018-G Symbolic problem solving in noisy and novel task environments.

[2] David W. Aha, Dennis Kibler, and Marc K. Albert. 1991. Instance-based learning algorithms. *Machine Learning* 6, 1 (Jan. 1991), 37–66. https://doi.org/10.1007/bf00153759

[3] Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, and Salvador García. 2011. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *Multiple-Valued Logic and Soft Computing* 17, 2-3 (2011), 255–287.

[4] K. Bache and M. Lichman. 2013. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[5] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (sep 1975), 509–517. https://doi.org/10.1145/361002.361007

[6] Jorge Calvo-Zaragoza, Jose J. Valero-Mas, and Juan R. Rico-Juan. 2017. Prototype Generation on Structural Data Using Dissimilarity Space Representation. *Neural Comput. Appl.* 28, 9 (9 2017), 2415–2424. https://doi.org/10.1007/s00521-016-2278-8

[7] Gail A. Carpenter and Stephen Grossberg. 1998. *Adaptive Resonance Theory (ART).* MIT Press, Cambridge, MA, USA, 79–82.

[8] T. Cover and P. Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13, 1 (1967), 21–27. https://doi.org/10.1109/TIT.1967.1053964

[9] Israel Cruz-Vega and Hugo Jair Escalante. 2017. An Online and Incremental GRLVQ Algorithm for Prototype Generation Based on Granular Computing. *Soft Comput.* 21, 14 (7 2017), 3931–3944. https://doi.org/10.1007/s00500-016-2042-0

[10] Mikel Elkano, Mikel Galar, Jose Sanz, and Humberto Bustince. 2018. CHI-PG: A fast prototype generation algorithm for Big Data classification problems. *Neurocomputing* 287 (2018), 22 – 33. https://doi.org/10.1016/j.neucom.2018.01.056

[11] Hugo Jair Escalante, Mario Graff, and Alicia Morales-Reyes. 2016. PGGP: Prototype Generation via Genetic Programming. *Applied Soft Computing* 40 (2016), 569 – 580. https://doi.org/10.1016/j.asoc.2015.12.015

[12] Hugo Jair Escalante, Maribel Marin-Castro, Alicia Morales-Reyes, Mario Graff, Alejandro Rosales-Pérez, Manuel Montes-Y-Gómez, Carlos A. Reyes, and Jesus A. Gonzalez. 2017. MOPG: A Multi-Objective Evolutionary Algorithm for Prototype Generation. *Pattern Anal. Appl.* 20, 1 (2 2017), 33–47. https://doi.org/10.1007/s10044-015-0454-6

[13] Salvador Garcia, Joaquin Derrac, Jose Cano, and Francisco Herrera. 2012. Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study. *IEEE Trans. Pattern Anal. Mach. Intell.* 34, 3 (March 2012), 417–435. https://doi.org/10.1109/TPAMI.2011.142

[14] P E Hart. 1968. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory* 14, 3 (1968), 515–516.

[15] W. Hu and Y. Tan. 2016. Prototype Generation Using Multiobjective Particle Swarm Optimization for Nearest Neighbor Classification. *IEEE Transactions on Cybernetics* 46, 12 (2016), 2719–2731. https://doi.org/10.1109/TCYB.2015.2487318

[16] S. Impedovo, F.M. Mangini, and D. Barbuzzi. 2014. A Novel Prototype Generation Technique for Handwriting Digit Recognition. *Pattern Recogn.* 47, 3 (3 2014), 1002–1010. https://doi.org/10.1016/j.patcog.2013.04.016

[17] Brijnesh J. Jain and David Schultz. 2018. Asymmetric learning vector quantization for efficient nearest neighbor classification in dynamic time warping spaces. *Pattern Recognition* 76 (2018), 349–366. https://doi.org/10.1016/j.patcog.2017.10.029

[18] Stephen M. Omohundro. 1989. *Five Balltree Construction Algorithms.* Technical Report TR-89-063. International Computer Science Institute.

[19] Stefanos Ougiaroglou and Georgios Evangelidis. 2014. RHC: non-parametric cluster-based data reduction for efficient k-NN classification. *Pattern Analysis and Applications* 19, 1 (2014), 93–109. https://doi.org/10.1007/s10044-014-0393-7

[20] Stefanos Ougiaroglou and Georgios Evangelidis. 2015. Efficient editing and data abstraction by finding homogeneous clusters. *Annals of Mathematics and Artificial Intelligence* 76, 3 (2015), 327–349. https://doi.org/10.1007/s10472-015-9472-8

[21] Esmat Rashedi, Hossein Nezamabadi-pour, and Saeid Saryazdi. 2009. GSA: A Gravitational Search Algorithm. *Information Sciences* 179, 13 (2009), 2232 – 2248. https://doi.org/10.1016/j.ins.2009.03.004 Special Section on High Order Fuzzy Sets.

[22] Mohadese Rezaei and Hossein Nezamabadi-pour. 2015. Using gravitational search algorithm in prototype generation for nearest neighbor classification. *Neurocomputing* 157 (2015), 256 – 263. https://doi.org/10.1016/j.neucom.2015.01.008

[23] D. Sheskin. 2011. *Handbook of Parametric and Nonparametric Statistical Procedures.* Chapman & Hall/CRC, New York, NY, USA.

[24] Leandro A. Silva, Bruno P. de Vasconcelos, and Emilio Del-Moral-Hernandez. 2021. A Model to Estimate the Self-Organizing Maps Grid Dimension for Prototype Generation. *Intell. Data Anal.* 25, 2 (jan 2021), 321–338. https://doi.org/10.3233/IDA-205123

[25] Ilia Sucholutsky and Matthias Schonlau. 2021. Optimal 1-NN prototypes for pathological geometries. *PeerJ Computer Science* 7 (Apr 2021), e464. https://doi.org/10.7717/peerj-cs.464

[26] Isaac Triguero, Joaquín Derrac, Salvador Garcia, and Francisco Herrera. 2012. A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification. *Trans. Sys. Man Cyber Part C* 42, 1 (Jan. 2012), 86–100. https://doi.org/10.1109/TSMCC.2010.2103939

[27] Jeffrey K. Uhlmann. 1991. Satisfying general proximity/similarity queries with metric trees. *Inform. Process. Lett.* 40, 4 (November 1991), 175–179.

[28] Roger Weber, Hans-Jörg Schek, and Stephen Blott. 1998. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB '98).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 194–205. http://dl.acm.org/citation.cfm?id=645924.671192

[29] Junjie Wu. 2012. *Advances in K-means Clustering: A Data Mining Thinking.* Springer Publishing Company, Incorporated, New York, NY, USA.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009