Efficient Dataset Size Reduction by Finding Homogeneous Clusters

Stefanos Ougiaroglou*
stoug@uom.gr

Georgios Evangelidis
gevan@uom.gr

Department of Applied Informatics
University of Mecedonia
Thessaloniki, Greece

ABSTRACT

Although the k-Nearest Neighbor classifier is one of the most widely-used classification methods, it suffers from the high computational cost and storage requirements it involves. These major drawbacks have constituted an active research field over the last decades. This paper proposes an effective data reduction algorithm that has low preprocessing cost and reduces the storage requirements, and maintains classification accuracy at an acceptable high level. The proposed algorithm is based on a fast pre-processing clustering procedure that produces homogeneous clusters. The centroids of these clusters constitute the reduced training-set. Experimental results, based on real-life datasets, illustrate that the proposed algorithm is faster and achieves higher reduction rates than three known existing methods, while it does not significantly reduce the classification accuracy.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*; I.5.3 [**Pattern Recognition**]: Clustering—*Algorithms*

General Terms

Algorithms, Experimentation

Keywords

k-NN Classification, Clustering, Data reduction

1. INTRODUCTION

The k-Nearest Neighbor (k-NN) Classifier [4] is an extensively used lazy classification algorithm. It is a simple and easy to implement classifier and can be exploited in many domains. It classifies a new item x by searching in the training set (TS) for the k nearest items (neighbors) to x according to a distance metric (e.g., Euclidean distance). Then, x is assigned to the most common class determined via a majority vote of the retrieved k nearest neighbors. Ties are resolved either randomly or by the single nearest neighbor.

Although the k-NN classifier is considered to be an effective method, it has two major weaknesses that render its use irrelevant for large datasets: (i) it involves high computational cost, since all distances between the new, unclassified item and the training data items must be estimated, and, (ii) storage requirements are high since it has to maintain the TS. Multi-attribute indexes [17] can deal with the first weakness for datasets with moderate dimensionality (e.g. 2-10). In higher dimensions, the curse of dimensionality degrades their performance to the degree that sequential scans are more effective.

On the other hand, Data Reduction Techniques (DRTs) [20, 7, 13, 19, 23, 11, 8] can cope with both weaknesses since they build a small representative set of the available training data, which is usually called condensing set (CS).[1] Applying the k-NN classifier using this small set, we have the benefit of much lower computational cost and storage requirements. A DRT is effective when classification accuracy does not degrade significantly. DRTs are distinguished into selection [7] and abstraction [20] algorithms. Selection algorithms choose some items of the TS as representative items and put them into CS. On the other hand, abstraction approaches generate representatives by summarizing similar TS items.

Although many DRTs can achieve extremely high reduction rates, their execution constitutes a costly preprocessing step. In addition, many of the DRTs are parametric, i.e., the user has to define the number of representative items in advance (this usually involves an iterative execution of a trial-and-error procedure). The aforementioned procedure is inappropriate in application domains that periodically accept new training items, and thus, the CS must be rebuilt.

These observations and the need for fast k-NN classification algorithms in large and high-dimensional datasets constitute the motivation of our work. The contribution is the devel-

*Stefanos Ougiaroglou is supported by a scholarship from the Greek Scholarships Foundation (I.K.Y.)

[1] DRTs have two points of view: (i) item reduction, and, (ii) dimensionality reduction. We consider them from the first point of view.

opment of a fast, non-parametric, and easy to implement algorithm that achieves high reduction rates. The algorithm, which we call Reduction through Homogeneous Clusters (or RHC), is based on the well-known k-Means clustering algorithm[2] [14] and, thus, it can be easily integrated in many existing environments.

The rest of this paper is organized as follows. Section 2 briefly presents the related work. Section 3 considers in detail the RHC algorithm. In Section 4, RHC is experimentally compared to three known DRTs on eight real life datasets. Finally, Section 5 concludes the paper.

2. RELATED WORK

A great number of DRTs have been proposed in the literature. Space restrictions force us to present in details only the methods we compare our method with in Section 4. In addition, we mention some methods that also adopt clustering in order to speed-up the k-NN classifier. For the interested reader, abstraction and selection algorithms are reviewed, evaluated, and compared to each other in [20] and [7] respectively. Other relevant reviews can be found in [11, 8, 13, 19, 23].

The earliest and best known selection algorithm is the Condensing Nearest Neighbor (CNN) Rule [9]. It tries to put only the close-class-border items in CS. The basic idea behind CNN is that, since the non-close-class-border (or central) items do not define the decision boundaries, they can be removed without affecting the classification accuracy. Contrary to many other DRTs, CNN automatically determines the CS size based on the level of noise that exists in the TS as well as the number of classes (or, in other words, the number of boundaries). The algorithm uses two bins, S and T. Initially, a TS item is placed in S while the remaining items are placed in T. CNN classifies the content of T using that of S by employing the 1-NN classifier. Whenever an item of T is misclassified, it is moved to S. When there is no move during a complete pass of T, the procedure terminates. The items that have been placed in S constitute the CS. Many other selection approaches either extend the CNN-rule or are based on the same idea (see [7] for details). However, CNN continues to be the reference algorithm and it is used for comparison purposes in many research papers.

Prototype Selection by Clustering (PSC) [15] is a recently proposed selection algorithm whose main goal is fast execution of the reduction procedure rather than high reduction rate. PSC executes k-Means clustering [14] in order to find c clusters in TS. For the homogeneous clusters (i.e., clusters that contain only items of a specific class), it keeps only the nearest to the centroid TS item. For each non-homogeneous cluster, it keeps only the items that define the decision boundaries between different classes in the cluster. A disadvantage of PSC is that the user has to determine the value of parameter c through a trial-and-error procedure.

Other DRTs based on clustering include the Self-Generating Prototypes (SGP) algorithms [6] and the Symbolic Nearest Mean Classifier (SNMC) [5]. Contrary to PSC, both belong to the abstraction category and, like PSC, they are parametric. Additionally, there are some methods based on clustering that although they do not reduce the TS size, they speed-up the k-NN classifier [10, 24, 16, 21, 12]. For each new item, they dynamically determine which subset of the initial TS should be searched. All of them are parametric.

The family of Reduction by Space Partitioning (RSP) algorithms [18] constitutes a popular set of three abstraction algorithms known as RSP1, RSP2 and RSP3. They can be characterized as extensions of the Chen and Jozwik algorithm (CJA) [3]. The latter, initially finds the most distant items, A, B in TS. Then, it divides the TS into two subsets. One subset includes the items nearest to A, while the other the TS items nearest to B. CJA continues by dividing the subset with the greatest diameter. This procedure is executed repetitively until the number of subsets is equal to a user-predefined threshold. Finally, it places in CS a mean item (centroid) for each produced subset. The class label of each centroid is the most common class in the corresponding subset. In contrast, RSP1 computes as many centroids as the number of different classes in the subset. RSP1 and RSP2 differ to each other on how they choose the next subset that will be split. RSP3 continues splitting the non-homogeneous subsets and terminates when all of them become homogeneous. RSP3 is non-parametric since it automatically determines the CS size.

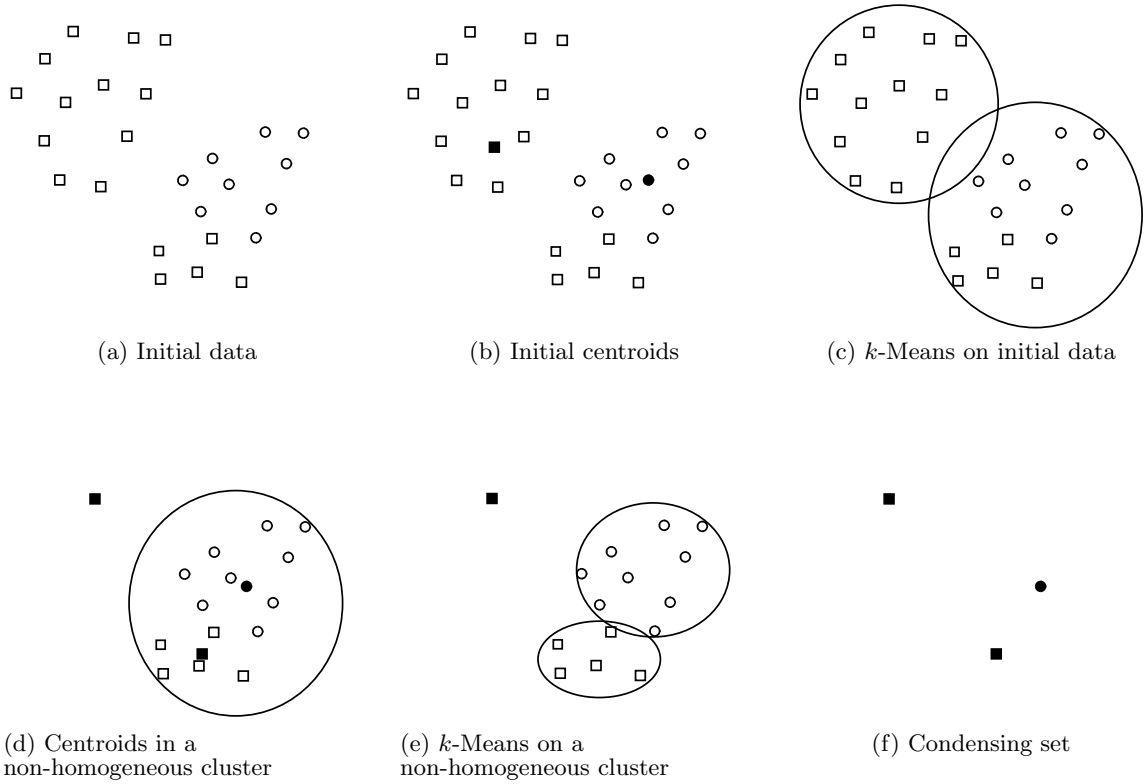
Some selection algorithms attempt to improve the classification accuracy rather than achieve high reduction rates. This is achieved by removing the noisy and the close-class-border items leaving smoother decision boundaries. These approaches are called editing algorithms. The reduction rates of the DRTs depend mainly on the level of noise in the TS. Thus, their success usually implies the execution of an editing routine before the application of the main reduction procedure [13]. Some selection approaches, such as DROP algorithms [23] and IB3 [1], integrate the idea of editing, and are called hybrid (see [7] for details). The reference editing algorithm is the ENN-rule [22]. It removes the irrelevant items by using the following rule: a TS item is removed, if its class does not agree with the majority of its k nearest neighbors. Thus, ENN-rule needs to compute all the distances among the TS items, i.e., $\frac{N*(N-1)}{2}$ distances.

3. PROPOSED METHOD

The Reduction through Homogeneous Clusters (RHC) algorithm is based on a simple idea that adopts the well-known k-Means clustering algorithm in a repetitive manner. Particularly, it continues constructing clusters until all of them are homogeneous, i.e., they contain items only of a specific class. RHC is summarized in Figure 1.

The RHC algorithm begins by finding the mean item (centroid) for each class by averaging the attribute values of the corresponding items in TS (Figure 1(b)). Thus, for a dataset with c classes, it computes c centroids. Then, RHC executes the k-Means clustering algorithm using the c aforementioned centroids and builds c clusters (Figure 1(c)). For each homogeneous cluster, it places the cluster centroid in CS. On the other hand, for each non-homogeneous cluster x, RHC counts the number of distinct classes in x and computes their initial centroids (Figure 1(d)). Afterwards, it executes the k-Means algorithm for the items of cluster x

[2]One should not confuse k of k-NN with k of k-Means.



(a) Initial data (b) Initial centroids (c) k-Means on initial data

(d) Centroids in a non-homogeneous cluster (e) k-Means on a non-homogeneous cluster (f) Condensing set

Figure 1: Reduction through Homogeneous Clusters

(Figure 1(e)). This procedure stops when all clusters are homogeneous. In the end, CS contains centroids of homogeneous clusters (Figure 1(f)).

Considering the proposed algorithm, it is obvious that RHC generates many representative items for close-class-border data areas and few representative items for the "central" class data areas. Thus, the more the classes in the training data and/or the higher the noise, the lower the reduction rates achieved.

Algorithm 1 is a non-recursive implementation of RHC. It uses a queue data structure, *QueueClust*, to hold unprocessed clusters. Initially, the whole TS constitutes an unprocessed cluster and is inserted in *QueueClust* (line 1). At each repeat-until iteration, the algorithm dequeues the cluster C from the head of *QueueClust* (line 3) and checks whether C is homogeneous or not. If it is a homogeneous cluster (line 4), its centroid is placed in CS (line 5). Otherwise, that is, if C is a non-homogeneous cluster (line 6), RHC computes a list of centroids (*InitCentroids*), one for each of the different classes that exist in C (lines 8-11). Then, RHC calls the function K-MEANS, which implements the k-Means clustering algorithm, with parameters the current non-homogeneous cluster C and the list of the initial centroids *InitCentroids*. The function returns a list of clusters (*ClList*) produced (line 12). The clusters of *ClList* are inserted into *QueueClust* (lines 13-15). The loop continues until *QueueClust* becomes empty (line 16), i.e., there are no more non-homogeneous clusters.

Algorithm 1 RHC Algorithm

Input: TS **Output:** CS

```
1: Enqueue(QueueClust, TS)
2: repeat
3:    C ← Dequeue(QueueClust)
4:    if C is homogeneous then
5:       Put the mean vector (centroid) of C into CS
6:    else
7:       InitCentroids ← ∅
8:       for each Class L in C do
9:          Centroid_L ← Compute the mean vector of items
            that belong to L
10:         InitCentroids ← InitCentroids ∪ Centroid_L
11:      end for
12:      ClList ← K-MEANS(C, InitCentroids)
13:      for each new cluster X in ClList do
14:         Enqueue(QueueClust, X)
15:      end for
16:   end if
17: until IsEmpty(QueueClust)
18: return CS
```

Actually, RHC combines the idea of RSP3 [18] with that of PSC [15]. It retains their advantages and avoids their weaknesses. Let's recall that PSC is a fast and parametric algorithm, while RSP3 is non-parametric but involves high computational cost (procedure for finding the most distant items in each subset). Thus, RSP3 is inappropriate for large datasets. Contrary to PSC, RHC is a non-parametric algo-

rithm. Contrary to RSP3, RHC is a fast approach since it is based on the k-Means algorithm. Note that we have adopted the full cluster consolidation (no item re-assignment during a complete pass of data) as the k-Means stopping condition. The proposed method could become even faster, had we used a more efficient stopping condition.

4. PERFORMANCE EVALUATION

4.1 Experimental setup

The proposed algorithm was evaluated by using eight real life datasets distributed by KEEL Repository[3] [2]. These datasets are summarized in Table 1. None of them has missing values. For comparison purposes, we implemented two selection algorithms, CNN-rule [9] and PSC [15] as well as an abstraction approach, RSP3 [18]. We selected these methods because: (i) CNN-rule and RSP3 are popular algorithms that have been used in many research papers for comparison purposes, (ii) PSC and RHC have the same goal, that is, the fast execution of the reduction procedure (or, low preprocessing cost), and, (iii) RHC is based on the idea of RSP3 and PSC.

In addition, we wanted to evaluate how the proposed method works on noise-free data. Thus, we run our tests twice using the original and an edited version of the TS. For editing purposes, we implemented the ENN-rule [22] and used it by defining k=3 [23]. All implementations were written in C. Moreover, the Euclidean distance was adopted as the distance metric. The datasets were used without data normalization or any other data transformation.

For each dataset and algorithm, we report three measurements: (i) Accuracy (Acc), (ii) Reduction Rate (RR), and, (iii) Preprocessing Cost (PC) in terms of distance computations. These measurements were obtained by a five-cross-fold validation schema. Thus, we run five training/testing set k-NN experiments for each dataset and each algorithm and we report the averages. We used the five already constructed pairs of sets hosted by the KEEL repository (it distributes all datasets in three forms: original, 5-folds and 10-folds). Of course, only the training set was preprocessed by the reduction algorithms (ENN-rule was included to reveal the level of noise in the datasets). For each one of the five pairs of training/testing sets, we used the k parameter that achieved the highest accuracy. Ties were resolved by the 1-NN rule.

Apart from PSC, all algorithms are non-parametric. For determining the value of c for PSC (number of clusters built), we run experiments by building $c = r \times j$, $j = 2, 4, \cdots, 10$, clusters, where r is the number of discrete classes in the data, as Lopez et al. did in their experiments [15]. Thus, we built five PSC classifiers for each dataset.

4.2 Comparisons

The results of our experiments can be found in Table 2 and Table 3, for the original and the edited datasets, respectively. Each table column lists the measurements of a specific classifier. The PC measurements are in million distance computations. For reference, in both tables we report the accuracy values of the conventional k-NN classifier (the one applied on the original training set) and the ENN measurements. We note that, in Table 3, the PC measurements of CNN, RSP3, PSC and RHC algorithms do not include the cost of editing. In these cases, the total preprocessing cost can be computed by adding the PC measurements of the ENN column.

[3]http://sci2s.ugr.es/keel/datasets.php

Table 1: Dataset description

dataset	Size	Attr.	Classes
Letter Recognition (LR)	20000	16	26
Magic G. Telescope (MGT)	19020	10	2
Pen-Digits (PD)	10992	16	10
Landsat Satellite (LS)	6435	36	6
Shuttle (SH)	58000	9	7
Texture (TXR)	5500	40	11
Phoneme (PH)	5404	5	2
Ring (RNG)	7400	20	2

At a glance, we observe that RHC has the lowest preprocessing cost. In almost all cases, it is lower than PSC, whose major goal is to reduce the preprocessing cost. To be fair, we should mention that the k-Means part of RHC includes a small extra preprocessing overhead for computing the initial mean items. However, this cost is almost insignificant compared to the cost of distance computations that assign items to clusters, and we do not include it in our measurements.

In all cases, RHC achieves the highest reduction rates. This means that the k-NN classifier executes faster when using the condensing set produced by RHC. Our measurements confirm that RSP3 is a time-consuming approach that achieves low reduction rates. However, RSP3 has the highest accuracy, very close to the one measured for the conventional k-NN classifier. RHC is more accurate than PSC and as accurate as CNN. Although the accuracy measurements of RHC do not reach that of RSP3, in most cases, they are close enough. Thus, they can be characterized as acceptable.

Concerning the difference between the two tables (original vs edited data), we observe that the reported accuracies are similar. On the other hand, all DRTs executed faster and produced smaller condensing sets. In the cases of MGT, LS, PH and RNG, the ENN rule removed many irrelevant items ($> 9\%$) and so, the differences in Acc and PC measurements between the two tables are obvious. It is worth noting that, in the case of edited data, RHC produces its CS by calculating an extremely low number of distances.

We can make a final comment concerning the average measurements (bottom row of both tables): the proposed algorithm builds the smallest condensing sets with the lowest preprocessing cost, and in all cases, it has an acceptable classification accuracy similar to that of CNN-rule.

5. CONCLUSIONS

Data reduction is very important when using the k-NN classifier on large datasets. In this paper, we presented a new fast non-parametric algorithm for data reduction. It works by using the k-Means algorithm to recursively cluster the training dataset into homogeneous clusters. The condensed set consists of the centroids of the final clusters. The pro-

Table 2: Comparisons on original datasets: Accuracy (Acc(%)), Reduction Rate (RR(%)) and Preprocessing Cost (PC(in million of distance computations))

Dataset		k-NN	ENN	CNN	RSP3	PSC j=2	PSC j=4	PSC j=6	PSC j=8	PSC j=10	RHC
	Acc:	96.01	94.98	92.84	95.51	82.73	85.65	87.14	87.73	89.43	93.59
LR	RR:	-	4.33	83.54	61.98	81.40	79.76	79.46	79.88	79.90	88.09
	PC:	-	127.99	163.03	326.52	66.32	110.06	129.16	165.32	169.92	41.85
	Acc:	81.32	80.95	80.64	81.08	70.60	71.11	73.13	73.24	73.36	80.09
MGT	RR:	-	20.08	59.94	53.70	70.72	71.13	71.46	71.62	71.72	73.76
	PC:	-	115.76	277.18	511.67	24.19	19.60	24.05	24.56	24.99	4.09
	Acc:	99.37	99.30	98.68	99.18	96.83	96.80	96.29	96.89	96.97	98.30
PD	RR:	-	0.67	95.36	89.04	91.44	92.86	93.72	94.41	94.83	96.52
	PC:	-	38.65	11.76	94.80	6.55	15.37	28.37	34.53	37.14	2.88
	Acc:	91.22	90.41	90.15	91.06	84.16	84.51	84.97	84.97	85.47	90.16
LS	RR:	-	9.07	79.83	73.14	85.05	84.60	84.64	84.99	85.09	89.84
	PC:	-	13.25	18.59	37.70	3.50	5.05	7.90	12.14	10.26	1.69
	Acc:	99.82	99.79	99.77	99.75	99.67	98.24	97.93	98.82	95.96	98.09
SH	RR:	-	0.18	99.37	98.59	96.88	97.68	97.87	98.33	98.54	99.55
	PC:	-	1076.46	45.40	17597.68	123.96	54.33	150.69	224.34	256.99	16.83
	Acc:	99.02	98.64	97.38	98.29	92.84	95.22	95.96	96.80	96.27	97.04
TXR	RR:	-	1.24	91.96	83.31	42.13	40.94	46.44	54.86	58.96	94.70
	PC:	-	9.68	5.57	27.63	5.18	8.92	11.24	13.99	17.57	3.63
	Acc:	90.10	88.14	87.83	87.77	78.82	79.56	79.41	79.24	79.04	86.36
PH	RR:	-	11.25	76.05	69.91	23.45	35.43	32.70	32.13	32.61	80.71
	PC:	-	9.35	13.47	20.32	7.79	3.46	4.02	4.37	4.38	0.65
	Acc:	74.69	62.20	84.61	81.84	74.96	72.43	73.19	73.57	73.59	84.55
RNG	RR:	-	28.81	72.95	56.48	21.79	22.23	19.58	18.86	18.57	90.35
	PC:	-	17.52	29.63	43.42	16.41	11.64	10.82	12.25	13.79	2.00
	Acc:	91.44	89.3	91.49	91.81	85.07	85.44	86.00	86.41	86.26	91.02
Average	RR:	-	9.45	82.38	73.27	64.11	65.58	65.73	66.89	67.53	89.19
	PC:	-	176.08	70.58	2332.47	31.74	28.55	45.78	61.44	66.88	9.20

posed method combines the advantages of PSC and RSP3 algorithms while avoiding their drawbacks.

Experimental measurements, derived by using the original and the edited versions of eight real life datasets, showed that the proposed method used the lowest preprocessing cost (in terms of distance computations) and achieved the highest reduction rates without significant loss of accuracy. We claim that these properties render the proposed method appropriate for environments where fast classification and/or low preprocessing cost are critical and slightly lower accuracy is acceptable.

6. REFERENCES

[1] D. W. Aha, D. F. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2011.

[3] C. H. Chen and A. Jóźwik. A sample set condensation algorithm for the class sensitive artificial neural network. *Pattern Recogn. Lett.*, 17:819–823, July 1996.

[4] B. V. Dasarathy. *Nearest neighbor (NN) norms : NN pattern classification techniques*. IEEE Computer Society Press, 1991.

[5] P. Datta and D. Kibler. Learning symbolic prototypes. In *In Proceedings of the Fourteenth ICML*, pages 158–166. Morgan Kaufmann, 1997.

[6] H. A. Fayed, S. R. Hashem, and A. F. Atiya. Self-generating prototypes for pattern classification. *Pattern Recogn.*, 40:1498–1509, May 2007.

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

[8] M. Grochowski and N. Jankowski. Comparison of instance selection algorithms ii. results and comments. In *Artificial Intelligence and Soft Computing - ICAISC 2004*, volume 3070 of *Lecture Notes in Computer Science*, pages 580–585. Springer Berlin / Heidelberg, 2004.

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

[10] S. Hwang and S. Cho. Clustering-based reference set reduction for k-nearest neighbor. In *4th international symposium on Neural Networks: Part II–Advances in Neural Networks*, ISNN '07, pages 880–888. Springer, 2007.

[11] N. Jankowski and M. Grochowski. Comparison of instances seletion algorithms i. algorithms survey. In *Artificial Intelligence and Soft Computing - ICAISC*

Table 3: Comparisons on edited data: Accuracy (Acc(%)), Reduction Rate (RR(%)) and Preprocessing Cost (PC(in million of distance computations))

Dataset		k-NN	ENN	CNN	RSP3	PSC j=2	PSC j=4	PSC j=6	PSC j=8	PSC j=10	RHC
	Acc:	96.01	94.98	92.06	94.56	81.72	85.23	86.82	88.00	88.56	92.73
LR	RR:	-	4.33	87.74	65.94	82.96	81.28	81.13	81.72	82.06	90.34
	PC:	-	127.99	118.38	344.39	34.30	49.85	59.32	78.32	85.36	31.05
	Acc:	81.32	80.96	80.90	81.32	75.71	75.68	77.12	77.19	77.54	80.76
MGT	RR:	-	20.08	89.97	84.04	88.66	88.65	89.13	89.34	89.30	93.06
	PC:	-	115.76	92.97	540.66	7.19	4.94	6.71	8.39	10.66	2.83
	Acc:	99.37	99.30	98.60	99.00	97.46	96.90	97.16	97.19	97.17	98.45
PD	RR:	-	0.67	96.42	90.39	91.95	93.53	94.27	95.12	95.63	97.19
	PC:	-	38.65	9.35	106.23	2.82	8.28	13.13	18.25	17.60	2.83
	Acc:	91.22	90.41	89.56	89.95	83.70	84.74	85.28	85.77	85.72	89.14
LS	RR:	-	9.07	91.27	85.62	91.26	91.18	91.44	91.87	91.91	95.06
	PC:	-	13.25	7.97	45.94	1.38	2.53	4.81	4.29	4.96	1.74
	Acc:	99.82	99.79	99.75	99.72	99.56	98.35	97.90	98.89	98.20	99.60
SH	RR:	-	0.18	99.58	98.87	97.09	97.87	98.12	98.46	98.69	99.66
	PC:	-	1076.46	26.17	15829.73	54.37	31.01	74.90	98.14	105.57	22.41
	Acc:	99.02	98.64	97.15	97.91	92.16	94.85	95.69	95.75	95.33	97.11
TXR	RR:	-	1.24	93.37	84.96	40.76	44.52	49.55	59.52	65.20	95.58
	PC:	-	9.68	4.48	27.34	2.17	4.10	5.88	6.13	8.92	3.00
	Acc:	90.10	88.14	86.88	86.46	80.06	79.74	79.87	79.52	79.50	85.40
PH	RR:	-	11.25	90.44	85.09	30.45	40.49	38.82	43.05	46.35	92.10
	PC:	-	9.35	6.44	20.43	3.64	1.55	1.77	1.42	2.26	0.47
	Acc:	74.69	62.20	70.49	72.28	59.84	61.57	61.14	61.51	60.88	77.47
RNG	RR:	-	28.81	87.84	79.12	36.93	33.57	34.06	32.96	34.64	96.73
	PC:	-	17.52	12.12	74.06	6.37	4.46	4.03	4.60	5.05	1.04
	Acc:	91.44	89.30	89.42	90.15	83.78	84.63	85.12	85.48	85.36	90.08
Average	RR:	-	9.45	92.08	84.25	70.01	71.39	72.06	74.01	75.47	94.96
	PC:	-	176.08	34.74	2123.60	14.03	13.34	21.32	27.44	30.05	8.17

2004, volume 3070 of *Lecture Notes in Computer Science*, pages 598–603. Springer Berlin / Heidelberg, 2004.

[12] L. Karamitopoulos and G. Evangelidis. Cluster-based similarity search in time series. In *Proceedings of the 2009 Fourth Balkan Conference in Informatics*, BCI '09, pages 113–118, Washington, DC, USA, 2009. IEEE Computer Society.

[13] M. Lozano. *Data Reduction Techniques in Classification processes (Phd Thesis)*. Universitat Jaume I, 2007.

[14] J. McQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symp. on Math. Statistics and Probability*, pages 281– 298, Berkeley, CA : University of California Press, 1967.

[15] J. A. Olvera-Lopez, J. A. Carrasco-Ochoa, and J. F. M. Trinidad. A new fast prototype selection method based on clustering. *Pattern Anal. Appl.*, 13(2):131–141, 2010.

[16] S. Ougiaroglou, G. Evangelidis, and D. A. Dervos. An adaptive hybrid and cluster-based model for speeding up the k-nn classifier. In *Proceedings of 7th International Conference on Hybrid Artificial Intelligence Systems, HAIS 2012*, volume 7209 of *LNCS*, pages 163–175, Salamanca, Spain, 2012. Springer.

[17] H. Samet. *Foundations of multidimensional and metric data structures*. The Morgan Kaufmann series in computer graphics. Elsevier/Morgan Kaufmann, 2006.

[18] J. S. Sánchez. High training set size reduction by space partitioning and prototype abstraction. *Pattern Recognition*, 37(7):1561–1564, 2004.

[19] G. Toussaint. Proximity graphs for nearest neighbor decision rules: Recent progress. In *34th Symposium on the INTERFACE*, pages 17–20, 2002.

[20] I. Triguero, J. Derrac, S. García, and F. Herrera. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42(1):86–100, 2012.

[21] X. Wang. A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality. In *The 2011 International Joint Conference on Neural Networks (IJCNN)*, pages 1293–1299, August 2011.

[22] D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE trans. on systems, man, and cybernetics*, 2(3):408–421, July 1972.

[23] D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.

[24] B. Zhang and S. N. Srihari. Fast k-nearest neighbor classification using cluster-based trees. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(4):525–528, 2004.