Exploring the effect of Data Reduction on Neural Network and Support Vector Machine Classification

Stefanos Ougiaroglou[a,b,*], Konstantinos I. Diamantaras[b], Georgios Evangelidis[a]

[a] *Dept. of Applied Informatics, School of Information Sciences, University of Macedonia, 54006 Thessaloniki, Greece*
[b] *Dept. of Information Technology, Alexander TEI of Thessaloniki, 57400 Sindos, Greece*

Abstract

Neural Networks and Support Vector Machines (SVMs) are two of the most popular and efficient supervised classification models. However, in the context of large datasets many complexity issues arise due to high memory requirements and high computational cost. In the context of the application of Data Mining algorithms, data reduction techniques attempt to reduce the size of training datasets in terms of the number of instances by selecting some of the existing instances or by generating new training instances. The idea is to speed up the application of the data mining algorithm with minimum or no sacrifice in performance. Data reduction techniques have been extensively used in the context of k-Nearest Neighbor classification, a lazy classifier that works by directly using a training dataset rather than building a model. This paper explores the application of data reduction techniques as a preprocessing step before the training step of Neural Networks and SVMs. Furthermore, the paper proposes a new data reduction technique that is based on k-median clustering algorithm. Our experimental results illustrate that, in the case of SVMs, data reduction techniques can effectively reduce the dataset size incurring small performance degradation. In the case of Neural Networks, the performance loss is somewhat

[☆] This is an extended version of the work presented in [1]

[*] Corresponding author
Email addresses: stoug@uom.gr (Stefanos Ougiaroglou), kdiamant@it.teithe.gr (Konstantinos I. Diamantaras), gevan@uom.gr (Georgios Evangelidis)
URL: http://users.uom.gr/~stoug (Stefanos Ougiaroglou)

greater, for the same data reduction rate, but both SVM and Neural Network models outperform the k-NN approach that is typically used in Data Mining applications.

Keywords: `Neural Networks, Support Vector Machines, k-NN classification, Data Reduction, Prototype Selection, Prototype Generation, Condensing`

2010 MSC: 00-01, 99-00

---

1. Introduction

In the recent years, problems involving high volumes of data challenge the effectiveness, efficiency and scalability of machine learning and data mining algorithms. Most popular proposed algorithms cannot be applied in such big data analysis scenarios, thus, new research issues have attracted the attention of both the industry and academia. For those algorithms data reduction is a necessary preprocessing step. Data Reduction can be thought of as item reduction or dimensionality reduction. In this paper, we deal with item reduction. More specifically, we consider classification tasks where data reduction processes are guided by the class labels.

Most of the Data Reduction Techniques (DRTs) were proposed in the context of dealing with the drawbacks of k-Nearest Neighbors (k-NN) classifier [2]. This classifier has high computational cost during classification, high memory requirements and is noise sensitive. This is because, the k-NN classifier works by using the training set as a model to classify new instances. Usually, the larger or more detailed the training set, the more accurate the classification.

A DRT can be either a Prototype Selection algorithm (PS) [3] that selects representative instances from the training set, or a Prototype Generation algorithm (PG) [4] that generates representatives by summarizing similar training instances. These chosen/generated representatives are called Prototypes. PS algorithms can be either editing or condensing. Editing attempts to improve accuracy by removing noise, outliers and mislabeled instances and by smooth-

ing the decision boundaries between classes. PG and PS-condensing algorithms build a small condensing dataset that represents the initial training data, thus, allowing the classification step to achieve low cost while accuracy remains almost as high as that achieved by using the original data. Some PG and PS-condensing algorithms are called hybrid because they integrate the concept of editing.

Dimensionality reduction and traditional sampling techniques have been applied to speed up the training times of eager classifiers. However, to the best of our knowledge, item reduction techniques have been used in the context of k-NN classification, but not in the context of large datasets in order to render the usage of Neural Networks and SVMs applicable on them. This is the key observation behind the motivation of this paper. Another motive is to check whether PG algorithms we proposed in the past can aid the development of fast and accurate Neural Networks and/or SVM based classifiers.

This paper contributes an experimental study on several datasets where Neural Networks and SVM based classifiers, which are trained by the original training data and the corresponding condensing sets built by state-of-the-art DRTs, are compared to each other and against the corresponding k-NN classifiers. Our study reviews in detail the algorithms that are used in the experimental study and reveals that the usage of DRTs leads to fast and accurate SVM-based classifiers.

Moreover, this paper presents a new variation of a PG algorithm we proposed in the past. The previously proposed algorithm is called Editing and Reduction through Homogeneous Clusters (ERHC) [5] and utilizes k-means clustering [6, 7] in order to build its condensing set. The main motivation behind the development of the new variation is to examine whether the use of k-medians [8] instead of k-means is a better choice in the case of datasets with outliers and noise. The new variation is called ERHC-MD and is expected to be more tolerant to the existence of outliers.

Although dimensionality reduction can be combined with PS-condensing and PG algorithms to obtain even faster k-NN classifiers or to speed-up the training of eager classifiers, this is not the objective of the present paper.

The rest of this paper is organized as follows. Section 2 briefly reviews Neural Networks, SVMs and the k-NN classifier. Section 3 presents in detail the PG and PS-condensing algorithms that we use in our experimental setup. The ERHC-MD algorithm is presented in Subsection 3.5.4. Section 4 presents the experimental study and the obtained results. Finally, Section 5 concludes the paper and gives direction for feature work.

2. Neural Networks, Support Vector Machines and the k-NN classifier

2.1. Feed-forward Neural Networks

Multilayer feed-forward neural networks [9] are some of the most popular learning models used in both classification and regression applications. The Multilayer Perceptron (MLP) with a single sigmoid hidden layer is known to have the *Universal Approximator* property [10], ie. it is capable of approximating any continuous function in the unit hypercube with arbitrary degree of accuracy, provided that there is no limit on the number of available hidden units. Different variations of the classical Back-Propagation algorithm [11] are typically used to train an MLP.

One of the most popular and efficient BP variations is the Conjugate Gradient method [12, 13]. Although networks with one hidden layer have been criticized for their relative inefficiency in capturing complex data representations compared to deep networks [14] they are still considered as yardsticks against which other learning methods are compared. One advantage of shallow networks is the fact that they have fewer tunable parameters to adjust, typically only the number of hidden units. Also the learning algorithm can have lower computational complexity compared to a deep learning model. Since our work focuses on the usefulness of data reduction as a preprocessing step for pattern classification and 2-layer neural networks are very popular classifiers we use them as one of the benchmark methods for our study.

2.2. Support Vector Machines

SVMs are supervised learning models introduced in 1995 by Cortes and Vapnik [15], but the original idea lies in the theory of statistical learning introduced by Vapnik [16] almost two decades earlier. They are suitable for pattern classification but can be easily extended to handle nonlinear regression problems in which case they are known as Support Vector Regressors (SVRs). The separating surface offered by an SVM classifier maximizes the *margin*, i.e., the distance of the closest patterns to it. This helps the generalization performance of the model and in fact it is related to the idea of Structural Risk Minimization [17, 18] which avoids over-fitting. With the use of nonlinear kernel functions such as Gaussian (RBF) or n-th order polynomials, SVM models can produce nonlinear separating surfaces achieving very good performance in complex problems.

Due to their good generalization performance these models have become very popular with a wide range of applications, including document classification, image classification, bio-informatics, handwritten character recognition, etc. One of the major drawbacks of these models is the memory and the computational complexity requirements for large datasets. The reason is that the separating surface is obtained by solving a quadratic programming problem involving an $N \times N$ matrix, where N is the number of items in the dataset. Although there are techniques that can reduce the complexity to $O(N^2)$ [19], the problem remains hard and the size of the problem can easily become prohibitively large calling for methods for data reduction such as the ones discussed in the following sections.

2.3. k-Nearest Neighbor classifier

The k-NN classifier [2] is an extensively used lazy (or instance-based) classification algorithm. Lazy classifiers do not build any classification model like eager classifiers do. The k-NN classifier is a quite simple and easy to implement algorithm, the predictions it makes are easy to understand/explain, it is analytically tractable, and, for $k = 1$ and unlimited instances the error rate

is asymptotically never worse than twice the minimum possible, which is the Bayes' rate [20].

A new instance is classified by retrieving the k nearest instances or neighbors to it from the training set. Then, the new instance is assigned to the most common class among the classes of the k nearest neighbors. This class is called the major class. The Euclidean distance is the commonly used distance metric, although any distance metric can be used. Despite not spending any time to train a model, the classification step is time consuming because in the worst case the algorithm must compute all distances between the new instance and all the training instances.

Another issue is that the selection of the value of k affects the accuracy of the classifier. The value of k that gives the highest accuracy depends on the dataset at hand and needs to be tuned in advance. Usually, large k values are appropriate for noisy datasets since they examine larger neighborhoods. In binary problems, an odd value for k should be used so that possible ties during nearest neighbors voting are avoided. In problems with more than two classes, ties are resolved by choosing a random "most common" class or the class voted by the nearest neighbor. The later is the approach adopted in the experimental study of this paper.

3. Prototype Generation and Condensing algorithms

Although, there are numerous PG and PS-condensing algorithms available in the literature, here, we review only the ones used in our experimental study. For the interested reader, abstraction and selection algorithms are reviewed, categorized and compared to each other in [4] and [3]. Interesting reviews are presented in [21, 22].

3.1. Condensing Nearest Neighbor rule

The Condensing Nearest Neighbor (CNN) rule [23] is the earliest condensing algorithm and uses a simple idea to build its condensing set. Instances that are

far away from decision boundaries, or, in other words, lie in an "internal" data area of a class can be removed without loss of accuracy. Thus, CNN-rule tries to retain only the instances that lie in the close-border areas. The close-border instances are selected as follows. Initially, an instance of the training set (TS) is moved to the condensing set (CS). CNN-rule uses the 1-NN rule and classifies the instances of TS by examining the instances of CS. If an instance is wrongly classified, it is probably close to decision boundaries and is moved from TS to CS. This procedure is repeated and the algorithm terminates when there are no moves from TS to CS in a complete pass of TS.

CNN-rule is sensitive to noise and wrongly selects "noisy" instances with their neighborhood. Consequently, noise affects the reduction rate. CNN-rule determines the number of the prototypes automatically, without requiring any user-defined parameters. Also, the multiple passes over the data guarantee that the removed training instances are correctly classified by the 1-NN classifier in the context of the condensing set. A drawback is that CNN-rule builds different condensing sets depending on the order of examining the same training instances.

3.2. The IB2 algorithm

IB2 is one of the Instance-Based Learning (IBL) algorithms presented in [24, 25] and is a one-pass version of CNN-rule. Each training instance $x \in TS$ is classified by the 1-NN rule on the current CS. If x is classified correctly, x is discarded, otherwise x is moved to CS.

IB2 also determines the size of the condensing set automatically, however, the condensing set highly depends on the order of training instances. As a one-pass algorithm, it is very fast, but, it does not guarantee that the removed instances can be correctly classified using the condensing set. In addition, it builds its condensing set in an incremental manner, i.e., new training instances can update an existing condensing set without considering the "old" instances that had already been used. Hence, IB2 can be applied in streaming environments.

3.3. The AIB2 algorithm

In [26], we presented a PG variation of IB2, called Abstraction IB2 (AIB2), that inherits all the properties of IB2. AIB2 considers that the prototypes should be close to the center of the data area they represent. Contrary to IB2, AIB2 does not ignore the instances that were correctly classified, but, these instances contribute to the condensing set by repositioning the nearest prototype. This is achieved by associating a weight with each prototype that denotes the number of instances it represents.

In an early step, a random training instance is placed in the condensing set and its weight becomes one. For each training instance x, AIB2 fetches its nearest prototype P from the current condensing set. If x has a class label different than the one of P, it is moved to the condensing set and plays the role of a new prototype with its weight set to one. Otherwise, the attributes of P are updated by taking into account the attributes of x and its weight. More formally, each attribute $attr(i)$ of P becomes $P_{attr(i)} \leftarrow \frac{P_{attr(i)} \times P_{weight} + x_{attr(i)}}{NN_{weight}+1}$. Thus, P slightly moves towards x. The weight of P is incremented by one and x is discarded.

3.4. The RSP3 algorithm

The ancestor of the Reduction by Space Partitioning (RSP) algorithms is the PG algorithm proposed by Chen and Jozwik (Chen's algorithm) [27]. Chen's algorithm retrieves the instances that define the diameter of the training data, i.e., the two most distant instances, a and b. Then, it splits the training data into two subsets by moving all the instances closer to a to C_a and the rest to C_b. Subsequently, Chen's algorithm selects to split the non-homogeneous subset with the largest diameter. Non-homogeneous are the subsets with instances of more than one class. If there are no non-homogeneous subsets, the algorithm proceeds by spitting the largest diameter homogeneous subsets. The splitting stops when the number of subsets becomes equal to a value specified by the user. The final step is the generation of prototypes, with each subset C is replaced

by its mean instance. The class label of the mean instance is the major class in C. The set of mean instances constitutes the condensing set.

Chen's algorithm generates the same condensing set regardless of the ordering of the instances. A drawback is that the user has to specify the size of the condensing set. Chen and Jozwik claim that this allows the user to define the trade-off between reduction rate and accuracy, but in practice it can be a costly trial-and-error procedure. Another weak point is that the instances that do not belong to the major class of a final subset are not represented in the condensing set.

RSP1 [28] is similar to Chen's algorithm, but it does not ignore instances. The algorithm computes as many means as the number of distinct classes in the non-homogeneous subsets, thus, it builds larger condensing sets than Chen's algorithm. However, the quality of the condensing set is improved by taking into account all training instances.

RSP2 selects the subset that will be split first by examining its overlapping degree. The overlapping degree of a subset is the ratio of the average distance between instances belonging to different classes and the average distance between instances that belong to the same class. This splitting criterion assumes that instances that belong to a given class are close to each other, whereas, instances that belong to different classes lie far apart. In [28], it is claimed that it is better to split the subset with the highest overlapping degree than that with the largest diameter.

RSP3 [28] is the only RSP algorithm that builds its condensing set without any user specified parameter. RSP3 eliminates both weaknesses of Chen's algorithm. It splits all the non-homogeneous subsets, i.e., it terminates when all subsets become homogeneous. RSP3 can use either the diameter or the overlapping degree as splitting criterion. In effect, the selection of splitting criterion is an issue of secondary importance because all non-homogeneous subsets are eventually split. Also, the order of the training instances is irrelevant.

RSP3 generates many prototypes for close-border areas and few prototypes for "internal" areas. The size of the condensing set depends on the level of noise

in the data. The higher the level of noise, the smaller subsets constructed and the lower reduction achieved. Note that the discovery of the most distant instances is a time-consuming procedure since all distances between the instances of the subset should be estimated. Thus, the usage of RSP3 may be prohibitive in the case of large datasets. In our experimental study, we used only RSP3 since we considered only non-parametric algorithms.

3.5. Reduction through Homogeneous Clusters

3.5.1. The RHC algorithm

We have recently proposed Reduction through Homogeneous Clusters (RHC) [29, 30], a PG algorithm. Like RSP3, RHC is based on the concept of homogeneity but employs k-means clustering [6, 7] to avoid the costly discovery of distant instances. Initially, the training data is considered as a non-homogeneous cluster. The algorithm computes a mean instance for each class in the cluster. Subsequently, RHC applies k-means clustering on the non-homegeneous cluster at hand by adopting the class-means as initial means. The result is the creation of as many clusters as the number of discrete classes in the cluster. This clustering process is applied on each non-homogeneous cluster, and in the end, all clusters become homogeneous. Each cluster contributes a prototype in the condensing set that is constructed by averaging the instances of the cluster.

RHC generates many prototypes for close-border areas and fewer for the "internal" areas. RHC uses the class-means as initial means for the k-means clustering in order to quickly find large homogeneous clusters. This has the advantage of achieving high reduction rates, since, the larger clusters discovered, the higher reduction rates achieved. Obviously, noise can affect the reduction rate. RHC is fast because it uses k-means clustering, and, its condensing set does not depend on the ordering of the training data. The experimental study presented in [29, 30] shows that RHC achieves higher reduction rates and is faster than RSP3 and CNN-rule, whereas accuracy remains high.

3.5.2. The dRHC algorithm

Although the experimental results presented in [29, 30] shows that RHC is a fast PG algorithm and achieves high reduction rates, it is memory based. With other words, it cannot handle big datasets which are not fit in the main memory or streaming training data which are gradually available.

dRHC [29] is a variation of RHC that retains all its properties and can also cope with large datasets that cannot fit in the available main memory or streaming datasets where new training data are gradually available. Hence, dRHC considers the available data in the form of data segments. The application of dRHC has two phases. When the first data segment arrives, the *initial CS construction* phase takes place, which is quite similar to RHC with the addition of a weight value for each generated prototype. The weight value is the number of items that are represented by the corresponding prototype. For the following data segments, dRHC applies the *CS update*.

The input of *CS update* phase is an already constructed condensing set (OLD) and the new data segment (DS) of training items. The algorithm returns an updated condensing set. The *CS update* phase begins by initializing as many clusters as the number of prototypes in OLD. Initially, each cluster has only the corresponding prototype. Then, for each training item x of DS, the algorithm retrieves its nearest prototype and x is assigned to the corresponding cluster. Thus, each cluster contains an old prototype and training items (from DS) with weight equal to one. The clusters are enqueued to a queue (Q) structure for further processing. The updated condensing set is generated taking into account the weight values. For each homogeneous cluster, the *CS update* phase estimates the weighted mean (centroid) of the cluster. For each non-homogeneous cluster C, the *CS update* phase computes a weighted mean for each class in C. Similar to RHC, the weighted class means are utilized as initial seeds in the call to k-means clustering for that cluster. Obviously, dRHC uses a variation of k-means clustering that estimates the cluster means by taking into account the weights. Consequently, a vector attribute a_j, $j = 1, 2, \ldots, n$ of a

class or cluster mean m_C is estimated as follows:

$$m_C.a_j = \frac{\sum_{x_i \in C} x_i.a_j \times x_i.weight}{\sum_{x_i \in C} x_i.weight}$$

The result of k-means clustering is the construction of as many clusters as the number of different classes in C. These clusters are enqueued to Q for further processing. The *CS update* phase ends when all clusters become homogeneous.

Old prototypes usually have weights greater than one and have higher influence in the computation of a new weighted class or cluster mean than any item of a new data segment, whose weight is one. dRHC creates different condensing sets by examining the data segments in different order. The order of data into the data segments is irrelevant. Note that although dRHC, IB2 and AIB2 can deal with data streams, they do not take into account the phenomenon of concept drift [31] that may exist in streaming environments.

3.5.3. The ERHC algorithm

The Editing and Reduction through Homogeneous Clusters (ERHC) [5] algorithm is a simple variation of RHC that can deal with noisy data. In ERHC, homogeneous clusters with only one instance are discarded. They probably represent noise. Thus, the final condensing set contains the means of the homogeneous clusters that have more than one instance. ERHC simultaneously removes noise and reduces the size of the training set. Therefore, it can be characterized as hybrid PG algorithm. The experimental study in [5] shows that this simple editing mechanism can improve classification performance when data contains noise.

3.5.4. The ERHC-MD algorithm

In this paper, we propose a new variation of the ERHC algorithm that utilizes k-median clustering [8] and Manhattan distance instead of k-means and Euclidean distance. We call this variation ERHC-MD. The k-median clustering works similar to k-means but instead of calculating the mean for each cluster, it estimates the median. Hence, ERHC-MD begins by computing the median

item of each class. Then, the algorithm executes k-median clustering using these class-medians as initial seeds. Subsequently, the algorithm continues executing similar to ERHC.

The advantage of using medians is that the clustering is not affected by outliers. To compute the median of a cluster, a median for each attribute is estimated. Contrary to the mean of an attribute that is a computed value, the median of each individual attribute comes from the dataset. The median of a cluster may not be a member of the dataset. Notice that median calculation is more time-consuming than mean calculation. Although ERHC-MD may not compute more distances than conventional ERHC, it involves the execution of quick sort (or the Median of Medians algorithm) for each attribute in each iteration. Therefore, the preprocessing cost of ERHC-MD is higher than that of ERHC.

4. Performance evaluation

4.1. Experimental setup

We conducted several experiments on fourteen datasets distributed by the KEEL repository[1] [32]. Their profiles are presented in Table 1. Six datasets do not contain noise and the rest datasets have variable levels of noise (see column "Noise" in Table 1). We do not use any editing algorithm for noise removal. For each dataset, we built eight condensing sets by applying the algorithms presented in Section 3. More specifically, we used CNN-rule, IB2, RSP3, RHC, dRHC, ERHC, ERHC-MD and AIB2. We do not test Chen's algorithm, RSP1 and RSP2 in the experimental study because they are parametric (the size of the condensing set must be supplied by the user) and we are interested in non-parametric approaches. The last column of Table 1 depicts the memory segment size that we used in dRHC execution[2] (e.g., We make the assumption that the device's memory can reside the specific number of items). Some datasets

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

[2] The work in [29] experimentally proves that the segment size is irrelevant

are small (e.g., MN2, BL). We include them in our study because we want to examine whether data reduction affects the accuracy on such datasets.

We trained several Neural Networks and SVMs on the original training set (without data reduction) and for each condensing set by using several parameter values. Finally, we kept the most accurate Neural Networks and SVMs. In Subsection 4.2, we report only that accuracy measurements. The Neural Network architecture used in our experiments is a two layer architecture with sigmoid hidden unit activations. Training is performed with the stochastic conjugate gradient method using the cross-entropy error. In the case of data sets with more than two classes the number of output units equals the number of classes and the target vectors are one-hot encoded. The number of hidden units of the NN model is determined by grid search. The SVM hyper-parameters *gamma* and C for the RBF kernel are also obtained through grid search [3].

For the six "noise-free" datasets, the k-NN classifier was run over the original training data and over the eight condensing sets by setting $k = 1$. Most of the time $k = 1$ is the best choice for noise-free data. For the other eight datasets, we adopted four k values, namely, 1, 5, 9 and 13.

All measurements presented in Subsection 4.2 are average values obtained via a five-fold cross-validation. We used the Euclidean distance as distance metric for the classification step. Since CNN-rule, IB2 and AIB2 depend on the order of class labels in the training set, we randomized all the datasets. Excluding CAR and KDD, we did not perform any other transformations. The CAR dataset has ordinal attributes. We transformed the attribute values into numerical values. Furthermore, we normalized to the interval [0-1] all attribute values of CAR. The KDD dataset contains 494020 items. However, only 141481 are unique. The duplicate items were removed. We also randomized the attribute values of KDD because their ranges vary extremely. It is worth mentioning that KDD contains rare classes. ERHC and ERHC-MD adopt an editing mechanism for noise removal. This mechanism eliminates the items that belong to the rare

[3] They can be found in `http://users.uom.gr/~stoug/neurocomputing2017.pdf`

Table 1: Dataset details

Dataset	**Size**	**Attr.**	**Classes**	**Noise**	**Seg.Size**
Magic Gamma Telescope (MGT)	19020	10	2	True	1902
Pen-Digits (PD)	10992	16	10	False	1000
Landsat Satellite (LS)	6435	36	6	True	572
Banana (BN)	5300	2	2	True	530
Shuttle (SH)	58000	9	7	False	1856
Texture (TXR)	5500	40	11	False	440
Yeast (YS)	1484	8	10	True	396
Phoneme (PH)	5404	5	2	False	500
Letter Image Recognition (LIR)	20000	16	26	False	2000
Balance (BL)	625	4	3	True	100
MONK2 (MN2)	432	6	2	True	115
Twonorm (TN)	7400	20	2	True	592
Car (CAR)	1728	6	4	True	200
KddCup (KDD)	141481	36	23	False	1000

classes. Thus, we do not execute ERHC and ERHC-MD on KDD.

4.2. Experimental results

We compared the eight DRTs to each other by estimating the Preprocessing Cost (PC) and the Reduction Rate (RR) that they achieved. Since the larger the training set used, the higher the cost for k-NN classifier to classify a new item and the higher the cost of the training procedure of SVMs and Neural Networks, the RR measurements can reflect the computational cost (the higher the RR, the lower the computational cost of k-NN classification and SVM and Neural Networks training). However, we computed the training times of SVMs and Neural Networks.

4.2.1. Reduction Rates and Preprocessing cost of Data Reduction Techniques

Table 2 presents the RR and PC measurements. Best measurements are in bold. The last row shows the average values. We observe that ERHC and ERHC-MD achieved the highest RR. This means that the Neural Network and SVM that uses the condensing set built by ERHC and ERHC-MD requires the least time for its training. AIB2 is the fastest DRT. It builds its condensing set by computing the fewest distances. On the other hand, RSP3 has the highest computational cost in order to build its condensing set. In addition, RSP3 seems to build the largest condensing sets. As expected, ERHC achieves higher RR than RHC and AIB2 is better in terms of RR and PC than IB2. Note that PC measurements of ERHC-MD do not include the cost of median computation. In fact, the real preprocessing cost of ERHC-MD is higher.

For the LIR and BL datasets, the k-medians clustering process involved in ERHC-MD did not converge. Therefore, in Table 2 and 8 the corresponding measurements are omitted. For the same reason, the average performance of ERHC-MD is not included in Figure 1. For the ERHC-MD to converge, we could adopt a stopping criterion other than the full clusters consolidation that we used (i.e., no move between clusters in an algorithm iteration). However,

this would not be fair for RHC, dRHC and the conventional ERHC since they adopt the full clusters consolidation.

4.2.2. Training times of SVMs and Neural Networks

Tables 3 - 5 present the execution times needed to train SVMs and Neural Networks. The k-NN classifier is a lazy classifier and thus, it does not train any model. The training times on the small MN2 and BL datasets were negligible and are omitted. The experiments were conducted on a personal computer with an Intel Core-i5 CPU and 6GB of RAM. The execution time measurements are in seconds. Each time measurement is the average time of the five folds. The tables include two columns for each type of classifier. The first column reports the time measurements needed to build to classification model that achieves the best accuracy (i.e., the accuracy measurements presented in Tables 6 - 9). The second column reports the average time measurements of all experiments conducted on the specific dataset (i.e., experiments with different parameter values). Obviously, data reduction reduces the training time at a minimum level. In all cases, the classifiers built on the condensing sets generated by RSP3 require more time than the other DRTs. For the large KDD, SH, LIR and MGT datasets, the usage of condensing sets instead of the original data is necessary. The difference between the training times is remarkable.

4.2.3. Accuracy measurements

Tables 6 - 9 show the accuracy measurements achieved by the Neural Networks, SVM and k-NN classifiers (Tables 7, 8, 9 are the continuations of Table 6). All tables contain nine rows for each dataset. Each row represents the different versions of the same dataset. The first one concerns the original data (i.e., without data reduction). The other eight rows concern the condensing set constructed by the DRTs. Each column of the table corresponds to a classifier. In particular, the third column corresponds to Neural Network models, the fourth column corresponds to SVM classifiers while the other four columns correspond to the k-NN classifiers. The best accuracy measurements of the dif-

Table 2: Comparison of DRT algorithms in terms of Reduction Rate (RR(%)) and Preprocessing Cost (PC (millions of distance computations))

Dataset		CNN	IB2	RSP3	RHC	ERHC	ERHC-MD	dRHC	AIB2
MGT	RR	60.08	70.60	53.70	73.76	84.46	**86.11**	74.62	71.90
	PC	281.49	34.61	511.67	**4.08**	**4.08**	4.75	26.03	33.05
PD	RR	95.36	96.23	89.22	96.52	**97.45**	97.10	97.23	97.19
	PC	11.75	1.78	86.66	2.88	2.88	2.62	1.44	**1.38**
LS	RR	80.22	84.62	73.19	89.84	92.95	**93.64**	88.35	86.72
	PC	17.99	2.22	37.70	1.69	1.69	**0.98**	**1.53**	1.92
BN	RR	77.44	83.27	75.21	79.68	90.33	**91.05**	82.41	83.40
	PC	11.49	1.58	18.76	**0.56**	0.56	**0.53**	1.53	1.53
SH	RR	99.37	99.44	98.59	99.55	99.69	**99.84**	99.50	99.46
	PC	45.30	8.26	1,7410.18	16.83	16.83	**6.03**	7.68	7.89
TXR	RR	91.90	93.33	83.31	94.71	**95.94**	95.65	94.95	94.95
	PC	5.65	0.84	27.63	3.63	3.63	2.37	0.68	**0.66**
YS	RR	32.68	44.82	27.36	49.83	79.34	**81.67**	51.23	46.94
	PC	1.41	0.39	2.12	0.84	0.84	**0.31**	**0.31**	0.37
PH	RR	76.04	80.85	69.94	80.71	88.05	**88.56**	82.34	81.75
	PC	13.45	1.96	20.31	**0.66**	0.66	**0.56**	1.64	1.84
LIR	RR	83.54	85.66	61.98	88.08	**92.03**	-	88.18	88.12
	PC	163.03	23.37	326.52	41.85	41.85	-	**19.57**	20.10
BL	RR	65.72	69.36	64.64	78.00	**86.68**	-	78.12	70.36
	PC	0.21	0.04	0.3	0.05	0.05	-	**0.03**	0.04
MN2	RR	87.23	91.68	61.33	96.47	**96.76**	95.67	96.88	92.54
	PC	0.04	0.006	0.13	0.007	0.007	0.007	**0.004**	0.005
TN	RR	82.09	88.25	84.56	96.63	**97.58**	97.13	95.37	93.44
	PC	22.13	2.07	37.13	1.64	1.64	1.66	**0.70**	1.10
CAR	RR	75.82	81.61	68.65	85.87	**90.31**	89.75	84.06	83.63
	PC	1.50	0.19	1.94	0.18	0.18	**0.10**	0.15	0.17
KDD	RR	99.12	**99.26**	98.54	99.19	-	-	99.22	99.21
	PC	384.90	**55.58**	20278.87	81.59	-	-	57.40	58.78
AVG	RR	79.44	83.50	72.16	86.35	91.66	92.38	86.60	84.98
	PC	68.59	9.50	1531.36	11.18	5.76	1.81	8.47	9.18

Table 3: Training Times in seconds - Datasets MGT through BN

Dataset	DRT	NNet	NNet avg	SVM	SVM avg
MGT	None	60.994	36.334	12.017	21.264
	CNN	24.111	17.053	3.086	3.219
	IB2	38.872	13.200	1.296	1.657
	RSP3	53.626	18.762	3.501	3.863
	RHC	7.153	12.642	0.911	1.260
	ERHC	4.226	8.397	0.268	0.333
	ERHC-MD	4.175	8.003	0.211	0.276
	dRHC	6.468	11.840	0.832	1.173
	AIB2	7.162	12.776	1.085	1.429
PD	None	8.137	10.663	0.583	1.955
	CNN	0.955	1.117	0.022	0.026
	IB2	0.981	0.935	0.016	0.020
	RSP3	1.391	1.551	0.045	0.085
	RHC	0.563	0.728	0.014	0.017
	ERHC	0.408	0.481	0.009	0.011
	ERHC-MD	0.435	0.572	0.011	0.013
	dRHC	0.775	0.707	0.012	0.012
	AIB2	0.753	0.696	0.010	0.012
LS	None	20.468	17.589	1.271	1.104
	CNN	7.771	4.209	0.122	0.114
	IB2	5.395	3.144	0.091	0.077
	RSP3	8.899	5.001	0.191	0.172
	RHC	2.819	2.174	0.040	0.039
	ERHC	2.163	1.664	0.014	0.019
	ERHC-MD	2.125	1.447	0.013	0.017
	dRHC	3.945	2.509	0.052	0.049
	AIB2	4.480	2.796	0.070	0.059
BN	None	25.224	11.149	0.213	0.690
	CNN	3.618	4.573	0.056	0.041
	IB2	3.168	4.002	0.031	0.023
	RSP3	3.623	4.821	0.047	0.049
	RHC	3.424	4.364	0.030	0.033
	ERHC	2.525	3.527	0.007	0.008
	ERHC-MD	2.726	3.610	0.006	0.007
	dRHC	3.226	4.199	0.024	0.025
	AIB2	3.568	4.166	0.019	0.023

Table 4: Training Times in seconds - Datasets SH through PH

Dataset	DRT	NNet	NNet avg	SVM	SVM avg
SH	None	136.229	154.061	5.034	8.317
	CNN	2.754	3.857	0.017	0.015
	IB2	3.571	3.701	0.014	0.012
	RSP3	3.872	5.215	0.033	0.037
	RHC	2.807	3.555	0.011	0.010
	ERHC	2.731	3.151	0.005	0.005
	ERHC-MD	2.705	3.173	0.004	0.003
	dRHC	2.665	3.626	0.012	0.011
	AIB2	3.230	3.698	0.012	0.011
TXR	None	3.001	5.183	0.184	1.705
	CNN	0.690	0.877	0.015	0.023
	IB2	0.560	0.783	0.013	0.018
	RSP3	0.889	1.024	0.033	0.072
	RHC	0.760	0.584	0.011	0.013
	ERHC	0.377	0.457	0.008	0.009
	ERHC-MD	0.325	0.452	0.008	0.010
	dRHC	0.446	0.535	0.010	0.012
	AIB2	0.436	0.486	0.009	0.012
YS	None	14.852	6.764	0.121	0.093
	CNN	8.536	5.785	0.068	0.053
	IB2	10.295	4.939	0.041	0.040
	RSP3	12.019	5.585	0.072	0.060
	RHC	10.326	4.688	0.047	0.035
	ERHC	4.477	2.741	0.008	0.008
	ERHC-MD	3.579	2.614	0.006	0.007
	dRHC	10.608	4.918	0.035	0.034
	AIB2	11.059	5.077	0.038	0.037
PH	None	29.204	11.981	0.606	0.670
	CNN	11.210	5.070	0.087	0.119
	IB2	9.290	4.422	0.054	0.083
	RSP3	9.032	5.437	0.102	0.159
	RHC	9.313	4.439	0.033	0.076
	ERHC	8.323	4.627	0.016	0.024
	ERHC-MD	7.101	3.878	0.016	0.023
	dRHC	13.328	5.232	0.038	0.065
	AIB2	7.009	4.450	0.039	0.070

Table 5: Training Times in seconds - Datasets LIR through CAR

Dataset	DRT	NNet	NNet avg	SVM	SVM avg
LIR	None	79.446	108.226	26.787	25.813
	CNN	11.321	17.657	0.515	1.039
	IB2	9.770	14.553	0.442	0.809
	RSP3	26.967	39.955	1.414	4.473
	RHC	7.153	10.242	0.323	0.564
	ERHC	3.850	5.136	0.186	0.278
	dRHC	7.074	10.107	0.323	0.568
	AIB2	6.924	10.468	0.322	0.567
TN	None	3.020	4.578	0.346	0.895
	CNN	2.061	1.228	0.068	0.062
	IB2	1.052	0.788	0.021	0.030
	RSP3	1.060	0.871	0.051	0.046
	RHC	0.293	0.384	0.004	0.004
	ERHC	0.213	0.230	0.003	0.002
	ERHC-MD	0.249	0.342	0.003	0.003
	dRHC	0.257	0.293	0.006	0.006
	AIB2	0.527	0.423	0.013	0.010
CAR	None	1.081	1.792	0.045	0.057
	CNN	0.701	0.987	0.008	0.008
	IB2	0.522	0.788	0.006	0.005
	RSP3	0.462	0.904	0.013	0.012
	RHC	0.407	0.581	0.005	0.003
	ERHC	0.256	0.330	0.003	0.002
	ERHC-MD	0.334	0.491	0.003	0.002
	dRHC	0.379	0.557	0.004	0.004
	AIB2	0.415	0.546	0.006	0.005
KDD	None	1734.170	1343.838	28.511	191.159
	CNN	23.179	16.936	0.200	0.129
	IB2	20.373	14.933	0.161	0.100
	RSP3	34.183	24.406	0.367	0.280
	RHC	15.275	15.718	0.099	0.121
	dRHC	20.761	15.229	0.171	0.117
	AIB2	21.131	15.578	0.187	0.119

ferent classifiers are in bold. The best accuracy among the different condensing sets is in italics.

The results depicted in these tables demonstrate that, in almost in all cases, Neural Networks and SVM classifiers are more accurate than the k-NN classifiers. In all datasets the Neural Network models show better accuracy compared to the other classifiers. The use of DRT methods leads to the decrease of Neural Network classification accuracy anywhere between 0.2% and 5% even for the best DRT method.

For most datasets trained with Neural Networks, CNN or RSP3 are the best DRTs with respect to accuracy. Interestingly, most DRTs seem to only slightly affect the accuracy achieved by SVMs. In most cases, a SVM trained by any condensing set is as accurate as the SVM trained by the initial training set. In eight datasets, the SVMs trained by the condensing set of RSP3 are the most accurate classifier. However, RSP3 has the highest PC and the lowest RR measurements. In the cases of the rest five datasets, the most accurate classifier is the SVM built by the condensing set of CNN-rule. The accuracy achieved by IB2 is close enough to that of CNN-rule, but IB2 is faster and achieved higher RR.

We conclude that the PG and PS-condensing algorithms can effectively be used for speeding-up the training process of SVMs without sacrificing accuracy. The same is partially true for Neural Networks: although training can be accelerated the accuracy can have more serious degradation depending on the dataset.

Furthermore, we observe that, in the case of SVMs and Neural Networks, the editing mechanism of ERHC and ERHC-MD is not as effective as it is when the k-NN classifier is used. In addition, although AIB2 achieves higher accuracy than IB2 in the case of k-NN classification, this is not true in the cases of SVMs and Neural Networks. Consequently, for SVMs, ERHC (and ERHC-MD) and AIB2 are not efficient extensions of RHC and IB2 respectively. This could be attributed to the fact that those DRTs are quite aggressive in terms of data reduction and were developed with k-NN classification in mind.

Considering the accuracy measurements achieved by ERHC and ERHC-MD, one cannot conclude if the one algorithm is better than the other. However, we can claim that ERHC-MD should be used only if the user knows that the dataset used includes outliers and extreme values.

Figure 1 summarizes the performance of the three types of classifiers tested in our experiments using the seven data reduction techniques (ERHC and KDD were omitted). Since ERHC-MD was not run over LIR, KDD and BL, it was omitted. Moreover, since ERHC was not run over KDD, KDD was omitted. Each point in any subplot represents the average classification accuracy vs. the average reduction rate over all datasets. Better DRT methods are those that approach the upper right corner of the plot. We can see that the CNN and RSP3 rules are the best data reduction techniques for both Neural Network and SVM learning perhaps because they are also the ones with the smallest reduction rate. Still, the reduction rate they offer is above 70% which makes their use quite appealing, especially for large datasets.

5. Conclusions

This paper demonstrated that the DRTs proposed for the k-NN classifier can also be applied for speeding-up classical machine learning models such as Neural Networks and SVMs. More specifically, the experimental measurements of our study showed that the usage of a DRT can reduce the time needed for the training process of both Neural Networks and SVMs with small impact on the classification accuracy. Especially in the case of SVMs the accuracy penalty is minimal. Although the particular DRTs have been proposed for speeding up the k-NN classifier, our study illustrated that the classification performance is better when Neural Networks or SVMs are used.

In the future, we plan to develop new PS-condensing or PG algorithms that deal with fast streaming data with concept drift. Our goal is to build fixed size condensing sets that can be incrementally updated. Moreover, we plan to explore the use of parallelization for the generation of the condensing set.

Table 6: Comparison of DRT in terms of accuracy (%) - Datasets MGT through BN

Dataset	DRT	NNets	SVM	1-NN	5-NN	9-NN	13-NN
MGT	None	**87.99**	83.79	78.14	80.48	80.84	81.12
	CNN	***86.92***	83.90	74.54	78.63	79.65	80.24
	IB2	**85.40**	83.38	71.97	76.84	78.50	79.11
	RSP3	**85.93**	83.71	74.96	78.90	80.15	80.52
	RHC	80.93	**83.08**	71.97	76.67	77.83	78.94
	ERHC	**83.68**	83.12	77.01	79.64	79.86	79.86
	ERHC-MD	**84.76**	83.04	75.50	79.20	79.64	79.60
	dRHC	82.05	**82.85**	72.96	76.55	78.23	78.74
	AIB2	82.83	**83.13**	73.36	77.40	78.36	78.89
PD	None	**99.90**	99.65	99.35	-	-	-
	CNN	98.02	**99.21**	98.68	-	-	-
	IB2	97.73	**99.02**	98.04	-	-	-
	RSP3	99.21	***99.50***	99.05	-	-	-
	RHC	97.11	**98.81**	98.30	-	-	-
	ERHC	97.76	**98.84**	98.63	-	-	-
	ERHC-MD	97.81	**98.65**	98.21	-	-	-
	dRHC	96.70	**98.82**	98.49	-	-	-
	AIB2	96.73	**98.74**	98.33	-	-	-
LS	None	98.07	**92.40**	90.60	90.69	90.62	90.21
	CNN	**92.60**	90.83	88.21	89.99	89.39	88.00
	IB2	**89.91**	88.73	86.87	88.45	87.55	86.23
	RSP3	***93.38***	91.14	90.57	90.16	89.64	89.50
	RHC	85.13	88.95	88.95	**89.54**	88.21	86.05
	ERHC	84.24	88.37	**89.01**	88.90	86.81	84.26
	ERHC-MD	83.87	**88.39**	**88.39**	88.35	86.01	84.17
	dRHC	86.93	88.08	**88.50**	87.72	85.75	84.33
	AIB2	88.89	88.19	**89.40**	87.69	86.03	84.34
BN	None	**91.47**	90.57	86.91	89.02	89.85	89.87
	CNN	**90.58**	90.53	85.62	88.15	89.09	88.77
	IB2	**90.40**	**90.06**	83.81	87.57	88.08	88.00
	RSP3	***90.91***	90.30	84.00	87.83	89.11	88.91
	RHC	**90.49**	90.25	83.28	87.23	88.19	88.38
	ERHC	89.79	**90.23**	88.00	89.09	88.64	88.00
	ERHC-MD	89.62	**90.17**	87.17	88.60	88.49	87.15
	dRHC	**90.57**	90.06	82.79	87.53	88.30	89.15
	AIB2	89.91	**90.49**	82.96	87.89	88.57	89.26

Table 7: Comparison of DRT in terms of accuracy (%) - Datasets SH through PH

Dataset	DRT	NNets	SVM	1-NN	5-NN	9-NN	13-NN
SH	None	**99.95**	99.84	99.82	-	-	-
	CNN	98.71	99.66	**99.76**	-	-	-
	IB2	99.30	99.62	**99.73**	-	-	-
	RSP3	99.48	***99.81***	99.75	-	-	-
	RHC	95.24	**99.64**	98.01	-	-	-
	ERHC	90.72	**99.64**	98.04	-	-	-
	ERHC-MD	80.60	98.26	**98.36**	-	-	-
	dRHC	98.21	**99.73**	99.70		-	-
	AIB2	98.01	**99.74**	99.72	-	-	-
TXR	None	**99.98**	99.84	99.02	-	-	-
	CNN	99.53	**99.58**	97.16	-	-	-
	IB2	99.13	**99.56**	96.35	-	-	-
	RSP3	99.64	***99.69***	98.29	-	-	-
	RHC	98.89	**99.24**	97.04	-	-	-
	ERHC	98.51	**99.10**	97.36	-	-	-
	ERHC-MD	98.56	**98.91**	96.98	-	-	-
	dRHC	99.13	**99.56**	97.60	-	-	-
	AIB2	98.96	**99.45**	97.69	-	-	-
YS	None	**79.80**	60.11	52.02	57.01	58.15	59.43
	CNN	**75.35**	59.84	49.06	52.97	55.66	56.94
	IB2	**72.12**	59.03	46.02	52.29	55.53	55.66
	RSP3	***76.70***	59.84	50.47	54.99	57.08	57.75
	RHC	**64.44**	58.89	48.85	52.29	54.65	54.92
	ERHC	**62.15**	59.09	53.17	56.00	55.86	56.80
	ERHC-MD	**60.00**	57.95	54.11	57.88	57.21	56.74
	dRHC	**65.59**	59.10	48.18	53.71	54.99	55.73
	AIB2	**68.96**	58.22	48.25	51.48	56.06	56.81
PH	None	**93.14**	89.19	90.10	-	-	-
	CNN	***89.25***	87.56	87.82	-	-	-
	IB2	85.27	**86.47**	85.57	-	-	-
	RSP3	86.29	**87.10**	86.94	-	-	-
	RHC	82.44	**85.88**	85.59	-	-	-
	ERHC	85.33	86.08	**86.57**	-	-	-
	ERHC-MD	83.92	**85.31**	83.73	-	-	-
	dRHC	83.96	**86.33**	85.33	-	-	-
	AIB2	83.57	**85.83**	84.92	-	-	-

Table 8: Comparison of DRT in terms of accuracy (%) - Datasets LIR through TN

Dataset	DRT	NNets	SVM	1-NN	5-NN	9-NN	13-NN
LIR	None	**99.14**	97.58	95.83	-	-	-
	CNN	93.13	**95.10**	92.84	-	-	-
	IB2	92.28	**94.75**	91.98	-	-	-
	RSP3	96.44	***96.51***	95.43	-	-	-
	RHC	88.57	**93.80**	93.59	-	-	-
	ERHC	87.16	92.60	**92.69**	-	-	-
	dRHC	89.04	**94.02**	93.90			
	AIB2	88.18	93.71	**94.12**	-	-	-
BL	None	**99.52**	90.96	78.40	84.16	87.84	88.32
	CNN	***99.20***	95.36	70.88	76.32	82.72	84.16
	IB2	**98.88**	95.36	70.72	77.28	81.60	83.20
	RSP3	**97.60**	94.72	73.28	76.64	82.88	82.88
	RHC	**97.28**	93.44	68.04	75.36	82.56	84.32
	ERHC	**94.88**	89.60	76.00	83.52	83.68	83.68
	dRHC	**96.32**	91.20	69.60	78.72	81.60	81.76
	AIB2	**96.80**	94.72	68.64	77.12	83.20	81.92
MN2	None	**100.00**	**100.00**	90.51	99.31	98.84	99.07
	CNN	95.86	**97.21**	95.84	90.97	84.03	84.26
	IB2	88.74	**94.66**	93.75	81.47	80.33	78.00
	RSP3	***99.77***	99.08	91.22	98.38	97.69	96.76
	RHC	73.10	91.43	**94.68**	80.09	67.12	47.12
	ERHC	57.47	90.26	**95.14**	77.53	63.65	47.12
	ERHC-MD	76.09	88.17	**88.20**	79.82	62.23	65.27
	dRHC	73.79	91.22	**97.68**	78.48	63.73	52.34
	AIB2	86.21	**93.04**	91.43	85.65	78.23	66.70
TN	None	**99.54**	97.89	94.88	96.91	97.31	97.38
	CNN	***98.74***	97.84	92.00	95.47	96.50	96.82
	IB2	**98.15**	97.81	89.15	94.95	95.87	96.45
	RSP3	**97.95**	97.70	92.68	96.30	96.88	97.31
	RHC	94.57	**97.39**	88.69	95.74	96.69	97.10
	ERHC	93.97	**97.45**	91.53	96.50	96.92	97.07
	ERHC-MD	87.70	**97.73**	87.95	95.41	96.23	96.97
	dRHC	96.45	**97.55**	93.08	69.62	97.31	97.31
	AIB2	97.12	**97.73**	93.47	96.69	97.28	97.28

Table 9: Comparison of DRT in terms of accuracy (%) - Datasets CAR and KDD

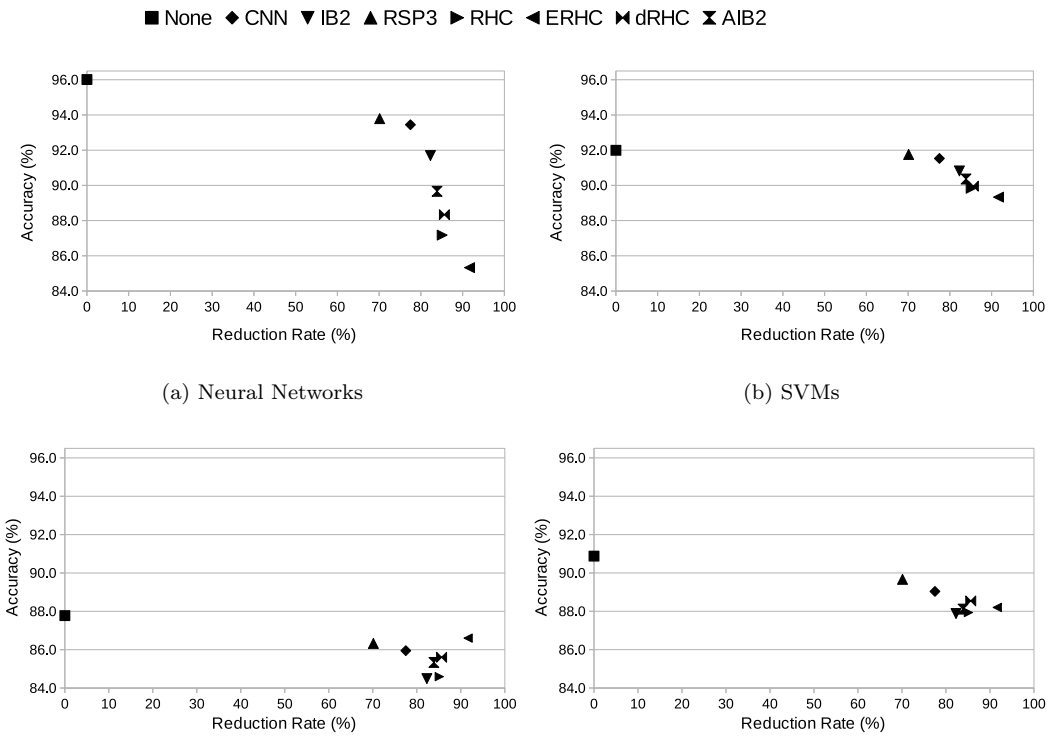
Dataset	DRT	NNets	SVM	1-NN	5-NN	9-NN	13-NN
CAR	None	**99.65**	94.10	85.53	89.75	90.39	89.76
	CNN	***96.88***	93.28	84.95	88.14	84.72	81.48
	IB2	**94.74**	92.24	84.43	86.11	84.20	82.41
	RSP3	**96.01**	93.75	85.59	89.70	89.12	88.14
	RHC	85.09	**87.10**	82.75	77.66	75.29	71.81
	ERHC	83.58	**86.98**	82.69	79.92	78.00	74.76
	ERHC-MD	85.43	**88.89**	78.53	78.13	76.74	72.64
	dRHC	89.65	**90.91**	85.07	87.15	85.88	84.49
	AIB2	89.48	**91.72**	87.09	87.56	85.30	82.40
KDD	None	**99.78**	99.76	99.71	-	-	-
	CNN	99.47	99.61	***99.66***	-	-	-
	IB2	99.31	**99.55**	99.48	-	-	-
	RSP3	99.47	***99.66***	99.60	-	-	-
	RHC	98.63	**99.46**	99.39	-	-	-
	dRHC	99.02	**99.62**	99.42	-	-	-
	AIB2	99.05	**99.55**	99.42	-	-	-

References

[1] S. Ougiaroglou, K. I. Diamantaras, G. Evangelidis, Efficient support vector machine classification using prototype selection and generation, in: L. Iliadis, I. Maglogiannis (Eds.), Artificial Intelligence Applications and Innovations: 12th IFIP WG 12.5 International Conference and Workshops, AIAI 2016, Thessaloniki, Greece, September 16-18, 2016, Proceedings, Springer International Publishing, Cham, 2016, pp. 328–340. doi:10.1007/978-3-319-44944-9_28.
URL http://dx.doi.org/10.1007/978-3-319-44944-9_28

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

[3] S. Garcia, J. Derrac, J. Cano, F. Herrera, Prototype selection for nearest neighbor classification: Taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 417–435. doi:10.1109/TPAMI.2011.142.
URL http://dx.doi.org/10.1109/TPAMI.2011.142



(a) Neural Networks

(b) SVMs

(c) k-NN (k=1)

(d) k-NN (best k)

Figure 1: Average Accuracies vs average Reduction Rates using different classification models and DRTs.

[4] I. Triguero, J. Derrac, S. Garcia, F. Herrera, A taxonomy and experimental study on prototype generation for nearest neighbor classification, Trans. Sys. Man Cyber Part C 42 (1) (2012) 86–100. doi:10.1109/TSMCC.2010.2103939.
URL http://dx.doi.org/10.1109/TSMCC.2010.2103939

[5] S. Ougiaroglou, G. Evangelidis, Efficient editing and data abstraction by finding homogeneous clusters, Annals of Mathematics and Artificial Intelligence 76 (3) (2015) 327–349. doi:10.1007/s10472-015-9472-8.
URL http://dx.doi.org/10.1007/s10472-015-9472-8

[6] J. McQueen, Some methods for classification and analysis of multivariate

observations, in: Proc. of 5th Berkeley Symp. on Math. Statistics and Probability, Berkeley, CA : University of California Press, 1967, pp. 281–298.

[7] J. Wu, Advances in K-means Clustering: A Data Mining Thinking, Springer Publishing Company, Incorporated, 2012.

[8] P. S. Bradley, O. L. Mangasarian, W. N. Street, Clustering via concave minimization, in: Advances in Neural Information Processing Systems -9, MIT Press, 1997, pp. 368–374.

[9] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, 1994.

[10] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural networks 4 (2) (1991) 251–257.

[11] P. Werbos, Beyond regression: New tools for prediction and analysis in the behavioral sciences.

[12] R. Fletcher, Practical methods of optimization, 2nd Edition, John Wiley & Sons, 1987.

[13] M. F. Møller, A scaled conjugate gradient algorithm for fast supervised learning, Neural networks 6 (4) (1993) 525–533.

[14] Y. Bengio, et al., Learning deep architectures for ai, Foundations and trends® in Machine Learning 2 (1) (2009) 1–127.

[15] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297.

[16] V. Vapnik, Estimation of dependencies based on empirical data, Nauka, Moscow, 1979, english translation: Springer Verlag, New York, 1982.

[17] V. Vapnik, A. Chervonenkis, Theory of pattern recognition.

[18] V. Vapnik, Statistical learning theory, Wiley New York, 1998.

[19] O. Chapelle, Training a support vector machine in the primal, Neural Computation 19 (2007) 1155–1178.

[20] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theor. 13 (1) (2006) 21–27. doi:10.1109/TIT.1967.1053964.
URL http://dx.doi.org/10.1109/TIT.1967.1053964

[21] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, J. Kittler, A review of instance selection methods, Artif. Intell. Rev. 34 (2) (2010) 133–143. doi:10.1007/s10462-010-9165-y.
URL http://dx.doi.org/10.1007/s10462-010-9165-y

[22] S. Garca, J. Luengo, F. Herrera, Data Preprocessing in Data Mining, Springer Publishing Company, Incorporated, 2014.

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

[24] D. W. Aha, D. Kibler, M. K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1) (1991) 37–66. doi:10.1023/A:1022689900470.
URL http://dx.doi.org/10.1023/A:1022689900470

[25] D. W. Aha, Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms, Int. J. Man-Mach. Stud. 36 (2) (1992) 267–287. doi:10.1016/0020-7373(92)90018-G.
URL http://dx.doi.org/10.1016/0020-7373(92)90018-G

[26] S. Ougiaroglou, G. Evangelidis, Efficient data abstraction using weighted IB2 prototypes, Comput. Sci. Inf. Syst. 11 (2) (2014) 665–678. doi:10.2298/CSIS140212036O.
URL http://dx.doi.org/10.2298/CSIS140212036O

[27] C. H. Chen, A. Jóźwik, A sample set condensation algorithm for the class sensitive artificial neural network, Pattern Recogn. Lett. 17 (8) (1996) 819–823. doi:10.1016/0167-8655(96)00041-4.
URL http://dx.doi.org/10.1016/0167-8655(96)00041-4

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

[29] S. Ougiaroglou, G. Evangelidis, Rhc: a non-parametric cluster-based data reduction for efficient k-nn classification, Pattern Analysis and Applications 19 (1) (2014) 93–109. doi:10.1007/s10044-014-0393-7.
URL http://dx.doi.org/10.1007/s10044-014-0393-7

[30] S. Ougiaroglou, G. Evangelidis, Efficient dataset size reduction by finding homogeneous clusters, in: Proceedings of the Fifth Balkan Conference in Informatics, BCI '12, ACM, New York, NY, USA, 2012, pp. 168–173. doi:10.1145/2371316.2371349.
URL http://doi.acm.org/10.1145/2371316.2371349

[31] A. Tsymbal, The problem of concept drift: definitions and related work, Tech. Rep. TCD-CS-2004-15, The University of Dublin, Trinity College, Department of Computer Science, Dublin, Ireland (2004).

[32] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, 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) (2011) 255–287.