Computer Science and Information Systems 17(1):71–92 https://doi.org/10.2298/CSIS190518044O

# Instance-based classification using prototypes generated from large noisy and streaming datasets

Stefanos Ougiaroglou[1,2], Dimitris A. Dervos[1], and Georgios Evangelidis[2]

[1] Department of Information and Electronic Engineering, International Hellenic University, GR-57400, Sindos, Thessaloniki, Greece
stoug@uom.edu.gr,dad@it.teithe.gr

[2] Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia Str., GR-54006, Thessaloniki, Greece
gevan@uom.gr

**Abstract.** Nowadays, large volumes of training data are available from various data sources and streaming environments. Instance-based classifiers perform adequately when they use only a small subset of such datasets. Larger data volumes introduce high computational cost that prohibits the timely execution of the classification process. Conventional prototype selection and generation algorithms are also inappropriate for data streams and large datasets. In the past, we proposed prototype generation algorithms that maintain a dynamic set of prototypes and are appropriate for such types of data. Dynamic because existing prototypes may be updated, or new prototypes may be appended to the set of prototypes in the course of processing. Still, repetitive generation of new prototypes may result to forming unpredictably large sets of prototypes. In this paper, we propose a new variation of our algorithm that maintains the prototypes in a convenient and manageable way. This is achieved by removing the weakest prototype when a new prototype is generated. The new algorithm has been tested on several datasets. The experimental results reveal that it is as accurate as its predecessor, yet it is more efficient and noise tolerant.

**Keywords:** $k$-NN classification, Data reduction, Prototype generation, Data streams, Large datasets, Noisy data.

## 1. Introduction

Classification is a fundamental concept in data mining [10]. Many classification algorithms have been proposed in the last half century [12]. They aim at accurate class prediction of the unclassified instances on the basis of a set of already classified instances (training set). The quality of the training set as well as its size determine the efficiency and the effectiveness of the algorithm and consequently they are vital for all classifiers.

Handling training data-streams [1] and large training sets in classification systems has attracted the interest of the data mining research community. The goal is the reduction of the high computational cost involved. The problem is more intense in cases of instance-based classifiers because the whole training set should be examined for each unclassified instance in order to classify it. In addition, executing classification algorithms on devices with limited memory (e.g., sensors) is also an important issue, since otherwise, transferring data to powerful servers for processing is inevitable. In both cases, a simple and obvious approach is the use of a subset of the available data. However, this subset probably cannot represent the whole training set and may harm the classification accuracy.

Instance-based classification is characterized to comprise a lazy learner algorithm because instead of involving a discriminative function it directly processes unclassified instances against the training dataset. The k-Nearest Neighbors ($k$-NN) [6] classifier is a typical example of a lazy learner. When an unclassified instance is to be classified, the $k$-NN classifier identifies and retrieves the k nearest instances from the available training set on the basis of a pre-specified distance metric; most often, the Euclidean distance. The nearest instances are called neighbors. The unclassified instance is assigned to the class that dominates in its k-nearest neighbor set. This task is very simple. However, since all the training set members need to always be available in memory and all distances between the unclassified instance and the training data have to be computed, the approach comprises a memory and CPU intensive task. Yet another drawback is that the $k$-NN classifier is not noise-tolerant. Noise affects the classification process and reduces accuracy.

Data Reduction Techniques (DRTs) [9,24] can remedy the stated drawbacks by introducing a preprocessing stage to the training dataset. A set of representative prototypes is created (usually called a condensing set) from the initial training data. Although there are few exceptions [21], by definition, DRTs cannot handle data of an incremental nature (data streams), as well as large dataset that do not fit in the main memory. To overcome these drawbacks, we recently proposed the dynamic RHC (dRHC) [19], and the abstraction IB2 (AIB2) [17,18] algorithms. Both construct/maintain a dynamic condensing set from the data stream or large data set. Dynamic because the algorithms update continuously the condensing set by incrementally considering new instances as the latter emerge.

Both dRHC and AIB2 operate on a small set of prototypes without seriously degrading the accuracy achieved by the $k$-NN classifier that operates on the whole of the original training set. Yet, they both introduce an unpleasant phenomenon that constitutes the major motivation behind this work: as new prototypes are generated and they get appended to the condensing set, the size of the latter may exceed the size of the available memory. A second motive is that both algorithms are not noise tolerant. False new prototypes may be generated as a result of the noise present in the initial training set. Consequently, there is a clear need to filter out the noise present in the training set, and to maintain the size of the condensing set under control, up to a user specified threshold value.

In [16], we made a first attempt to address the aforementioned issues. In particular, we proposed dRHC-V2, which is a variation of dRHC. Contrary to dRHC, dRHC-V2 keeps the size of the condensing set in a convenient, manageable by the classifier, level by ranking the prototypes and removing the least important ones. As soon as the condensing set size exceeds a user-specified threshold value, the condensing set prototypes are ranked on a calculated weight value, and the necessary number of low ranked prototypes are removed from the condensing set in order to maintain the set threshold. The experimental measurements presented in [16] show that dRHC-V2 filters the noisy data and keeps the classification accuracy at high levels. However, dRHC-V2 is not a one pass algorithm. It utilizes a ranking procedure that may not cope well with fast data streams. Even a quick sort algorithm may be inappropriate for very fast data streams, where new instances arrive at fast rates. This is another motive of the present work.

We propose a new variation of AIB2 called AIB2-V2. Like dRHC-V2, AIB2-V2 incorporates a mechanism for prototype weighting and removal. However, AIB2-V2 is of a more proactive nature. Weight values are calculated again as in the case of dRHC-V2 and as a new prototype enters the condensing set, the weakest (i.e., the one with the lowest weight) of the existing prototypes is removed from the set. In other words, the newly generated prototype replaces the weakest one. Contrary to dRHC-V2, AIB2-V2 does not rank the prototypes. The prototype with the minimum weight is removed. Noisy prototypes tend to involve low weight values. Hence, they are the first to be removed as new prototypes get appended to the (dynamically updated) condensing set. The experimental results show the new algorithm to be faster and more noise-tolerant than AIB2, with no sacrifice in accuracy and computational complexity.

The paper is organized as follows: In section 2, the fundamental issues about DRTs and their limitations are briefly presented. The section also includes a recap of the AIB2, dRHC and dRHC-V2 algorithms. Section 3 considers in detail the AIB2-V2 algorithm. In Section 4, the latter is experimentally compared to dRHC-V2 and to their predecessors on sixteen datasets. The experimental study is complemented with a statistical validation using the Wilcoxon signed rank test [7,23]. Section 5 suggests new directions for future work and concludes the paper.

## 2. Background knowledge

### 2.1. Data Reduction Techniques

DRTs [9,24] pre-process the training data and construct a set of prototypes that is then used by the $k$-NN classifier. DRTs can be grouped into two categories:

- Prototype Selection (PS) algorithms [9] select prototypes from the initial training set. PS algorithms can be also grouped into two subcategories:
  - PS-editing algorithms aim to remove noise from the training set and to “clean” the borders between classes. This way, the classification accuracy is improved.
  - PS-condensing algorithms aim for data condensation, i.e., the construction of a small set of prototypes (condensing set) that represents the initial training data.
- Prototype Generation (PG) algorithms [24] generate prototypes by summarizing on instances. As in the case of PS-condensing algorithms, the goal is data condensation.

Most PS-condensing and PG algorithms construct condensing sets by removing “internal” instances. These instances do not determine the borders between the classes and can be removed without accuracy loss. Thus, PS-condensing algorithms try to select only the instances that are close to the borders. These instances are called close-border instances and are essential to classification. PG algorithms generate many prototypes for the close-border areas and few for the “internal” areas. Unfortunately, most of PS and PG algorithms cannot handle noise and this leads to lower data reduction rates. In the case of PS-condensing algorithms, an instance with wrong class label is considered as close-border instance and it is erroneously included in the condensing set, along with its neighboring instances. In the case of PG algorithms, more prototypes are generated for noisy data areas because neighboring instances of different classes cannot be summarized. Consequently, for training sets with noise, editing should be applied beforehand.

Although there are some exceptions (e.g., the IBL algorithms [3,5]), DRTs are memory-based. All training data are assumed to reside in main memory. Hence, DRTs are unsuitable for large datasets that cannot fit into memory and they cannot be used on devices with limited memory. In addition, DRTs are appropriate only for static datasets. Except for few exceptions [21], they cannot process new incoming data, after the construction of the condensing set. Equivalently, they cannot dynamically update the condensing set. Suppose that a DRT constructs a condensing set by considering a training set $TS$. Also, suppose that new training data $D$ arrive. For the construction of an updated version of the condensing set, the DRT needs to operate from scratch on the complete training set $TS \cup D$. This means that the entire training instances set need be considered. Hence, DRTs are inappropriate for data streams where new training instances become gradually available. The dRHC [19] and AIB2 [17,18] algorithms are PG algorithms that can be used in such environments.

## 2.2. Review of dRHC and dRHC-V2

Dynamic RHC (dRHC) constitutes a descendant of the Reduction through Homogeneous Clusters (RHC) algorithm [19]. Inherent to the latter is the concept of cluster homogeneity. RHC utilizes $k$-means clustering. Suppose that a training set $C$ contains $D$ classes. Initially, the whole training set is considered as an unprocessed non-homogeneous cluster. RHC averages out over the instances of each class in $C$ and calculates a mean (centroid) for each class. Then, $k$-means runs over $C$ by using these class-centroids as initial seeds. The result is $D$ clusters. Each instance in $C$ is assigned to one of the $D$ clusters. Subsequently, the $D$ clusters are considered. For each one homogeneous cluster (i.e., involving instances of only one class), the cluster-mean comprises a representative prototype placed in the condensing set. On the other hand, for each non-homogeneous cluster, the algorithm proceeds recursively. When no non-homogeneous clusters are left, RHC terminates. This way, each homogeneous cluster contributes a (representative) prototype to the condensing set. Like most PG algorithms, RHC generates few prototypes for the "internal" data areas and more prototypes for close-border data areas. Like most DRTs, RHC is memory-based and as such it cannot manage data that cannot fit in memory and data streams.

The dRHC algorithm can manage large and / or streaming datasets. This is accomplished by considering the training data in the form of data segments. The size of the data segment is fixed and it can be adjusted according to the available memory. In the case of data streams, dRHC utilizes a buffer the size of the data segment that is set to accept incoming instances. When the buffer gets full, its content is moved forward for processing. Analogously, large datasets that cannot fit into memory are divided into equally sized data segments appropriate to the device's memory, and the latter are processed sequentially. The algorithm includes two main stages: stage 1 is the *initial condensing set construction*. It is executed only once utilizing the first data segment. It is identical to RHC with one difference: each one prototype is stored alongside with a weight attribute value equal to the number of instances it represents. Stage 2 is the *condensing set update*. It is executed following the arrival of data segment number two and onwards. It processes the prototypes of the current condensing set against the instances of a new data segment and constructs a new set of clusters. It then proceeds in a way similar to that of RHC.

Figure 1 illustrates an example of the execution of the *condensing set update* stage. Sub-figure (a) presents an existing condensing set that has three prototypes with the corresponding weights. Suppose that a new segment with seven new instances arrives (Sub-figure (b)). Each new instance carries a weight value equal to one. At first, each new instance is assigned to the cluster of the nearest prototype (Sub-figure (c)). No instance has been assigned to cluster B. Hence, the corresponding prototype remains unchanged. The instances assigned to cluster A are of the same class as the class of the prototype in A. Thus, A remains homogeneous. Therefore, the weighted mean in A is computed and it is placed in the condensing set alongside with its new weight value. In effect, the prototype "moves" towards the new instances to represent better this data area (Sub-figure (d)). Cluster C becomes non-homogeneous after the assignment of the new instances. For each discrete class in C, the algorithm computes a weighted class mean and executes $k$-means. Two new homogeneous clusters emerge (Sub-figures (d) and (e)). Eventually, dRHC computes a weighted cluster mean for each cluster as well as the corresponding weights. They constitute new prototypes and they are placed in the condensing set (Sub-figure (f)).

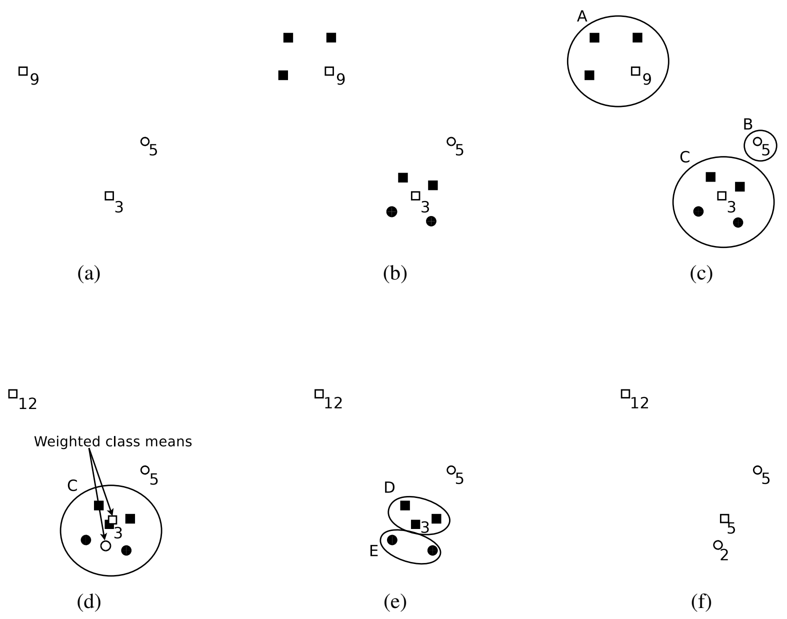


**Fig. 1.** Example of execution of the *condensing set update* stage of dRHC

RHC and dRHC are compared to each other and against state-of-the-art PS [11,3,14,26] and PG [22] algorithms in [19]. The results obtained reveal that dRHC involves the lowest preprocessing cost (i.e., it is the fastest to execute) and constructs the most compact condensing set without sacrificing accuracy. A week point is that noise in the training data results into having a larger number of non-homogeneous clusters which, consequently, leads to a lower reduction rate and to a higher preprocessing cost during the prototypes construction stage.

The dRHC-V2 algorithm constitutes a dRHC variation that retains the size of the condensing set within acceptable levels. The desirable maximum size of the condensing set comprises a user-specified input parameter ($T$). dRHC-V2 executes in a way similar to dRHC with one difference: as soon as the size of the condensing set exceeds the set number of $T$ prototypes, the least important of the latter are removed from the condensing set, to save space. In effect, the mechanism for prototypes removal comprises a post-processing step of the *condensing set update* stage. The input parameter $T$ is set by considering the available memory, the noise, and the desirable trade-off between computational cost and accuracy.

The stated functionality is implemented by having dRHC-V2 rank the prototypes according to their importance, following the execution of each *condensing set update* stage. Next, only the top $T$ prototypes are retained. The prototype weights are vital for the ranking procedure. It is reminded that the prototype weight depicts the number of instances represented by the prototype in question. A straightforward approach could be to remove the necessary number of prototypes with the lowest weights. However, this would not comprise a fair criterion. An old prototype probably has a higher weight value than that of a prototype generated at a subsequent *condensing set update* stage. Consequently, prototypes generated at later stages would be prone for removal as compared to older ones. Also, if the weight is adopted to comprise the sole ranking criterion, an old prototype of high weight that has survived many executions of the *condensing set updates* will probably tend to be favored to survive against all recently generated prototypes and will thus remain permantly in the condensing set. Therefore, prototype weight should not by itself comprise the sole contributor to the calculation of the prototype's rank measure.

To achieve fairness in prototype ranking, dRHC-V2 takes into consideration the weight as well as the age of the prototype. Thus, dRHC-V2 holds a counter of the data segments that have been processed at any one given instance in time. Each prototype has an extra attribute that denotes the number ($r$) of the data segment corresponding to its generation. In this respect, $r$ depicts the instance in time when the prototype in question got appended to the condensing set. Following the completion of each *condensing set update* stage, dRHC-V2 re-calculates the rank measure for all the condensing set prototypes. The measure is called Average number of Arrivals ($AnA$). This measure incorporates the prototype's weight ($w$) and age ($r$) contributors. More specifically, the prototype's $AnA$ rank measure is calculated as follows:

$$AnA = \frac{w}{ds - r + 1}$$

where $ds$ is the (sequence) number of the current data segment, $r$ is the number of the data segment corresponding to the prototype's generation, and $w$ is the prototype's weight value, i.e. the number of instances it represents. In other words, the denominator reflects the age of the prototype. A prototype is weak when it has low $AnA$ value.

Considering the above, prototypes generated by the latest *condensing set update* stage have $AnA$ values equal to their weight ($w$). For existing prototypes, their $Ana$ values are (re-)calculated by dividing their weights $w$ by their (updated) age values. Prototypes updated to represent new instances during the *condensing set update* stage have their $AnA$ value (re-)calculated by diving their new weight $w$ value by their (updated) age value.

The post-processing step of dRHC-V2 executes following the completion of the *initial condensing set construction* stage, as well as the execution of each one subsequent *condensing set update* stage. It operates on an already constructed condensing set, with given a maximum condensing set size $T$. When the size of the updated condensing set is found to exceed the set maximum $T$, the training set is trimmed to only contain the top $T$ prototypes with the highest $AnA$ values.

Noisy prototype instances present in the dRHC generated condensing sets usually relate to low $AnA$ values in dRHC-V2. As such, they are weak and comprise the first candidates to be removed when the $T$ threshold value is reached in dRHC-V2. Thus, contrary to dRHC, dRHC-V2 can deal with datasets that involve noise and, for such datasets, it achieves higher classification accuracy compared to dRHC.

Despite its fairness in treating the newly generated protoyes, dRHC-V2 is not tuned for handling the concept drift phenomenon [25] that may be present in data streams. Newly generated prototypes or prototypes updated during a *condensing set update* stage are in no way favored to survive and remain in the condensing set by removing older ones.

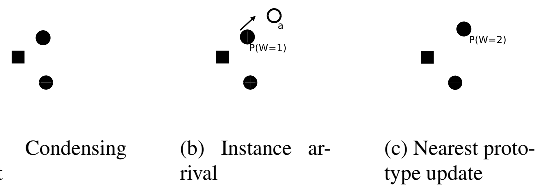
## 2.3. Review of AIB2

The Abstraction IB2 (AIB2) algorithm is a prototype generation version of the well-known IB2 condensing algorithm [2,3]. Contrary to most DRTs, the latter is a one-pass algorithm and constructs its condensing set in an incremental manner[3]. This means that IB2 can add new prototypes to an already constructed condensing set without needing to retain the instances that had been used for the initial construction of the condensing set. From this point of view, IB2 is suitable for data streams and for large datasets that cannot fit into the device's memory. IB2 starts by moving the first training instance to the condensing set. For each following training instance $inst$, IB2 examines the current condensing set and retrieves the nearest prototype $p$ to $inst$. If $inst$ has a different class from $p$, it is moved to the condensing set. Otherwise, it is ignored.

AIB2 inherits all the properties of IB2. In addition, AIB2 not only appends new prototypes to an already constructed condensing set; it may also update existing prototypes. The motive behind the development of AIB2 is that prototypes should be the centroids of the instances they represent. Thus, if the examined instance has the same class label with its nearest prototype in the current condensing set, the examined instance is not ignored as in the case of IB2. Instead, it contributes to the shaping of the condensing set by updating the nearest prototype. To accomplish this, AIB2 assigns a weight value to each prototype. The weight value denotes the number of instances that the prototype represents and it is used to move the prototype in the data space. In effect, the nearest prototype moves towards the examined instance.

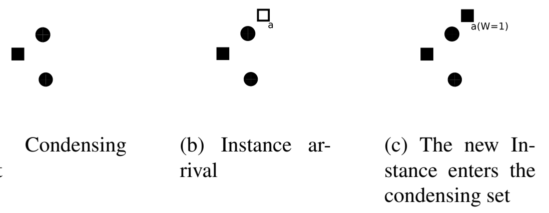
Considering the above, AIB2 improves on the representation effectiveness of the condensing set prototypes in comparison with IB2. Consequently, higher classification accuracy is achieved. Moreover, by having prototypes act as centroids for the instances they represent, AIB2 reduces the number of prototypes in the condensing set. In this respect, higher reduction rates and lower computational cost are achieved when compared to IB2. Contrary to dRHC and dRHC-V2, AIB2 does not consider the dataset in the form of data segments. Each instance is processed individually. Like dRHC, AIB2 is also subject to the noise effect: a noisy instance nearest to a prototype is likely to be of a different class. As such, its inclusion in the condensing set increases the number of prototypes in the latter.

[3] According to the reviews [9,24], DRTs can be either incremental or decremental. This depends on how they construct the condensing set. Here, the term incremental refers to the algorithm's ability to update an already constructed condensing set



(a) Condensing Set (b) Instance arrival (c) Nearest prototype update

**Fig. 2.** AIB2 example: repositioning an existing prototype. CS prototypes are colored black. The new instance is colored white. Shapes indicate classes.



(a) Condensing Set (b) Instance arrival (c) The new Instance enters the condensing set

**Fig. 3.** AIB2 example: new prototype enters the condensing set. CS prototypes are colored black. The new instance is colored white. Shapes indicate classes.

AIB2 execution examples are depicted in Figures 2 and 3. Initially, the condensing set includes three prototypes. Suppose that a new circle instance $a$ arrives (Figure 2(b)). Since $a$ is closer to a prototype $P$ of the same class, $P$ moves towards $a$ and its weight is increased by one (Figure 2(c)). On the other hand, suppose that a new square instance, $a$, arrives (Figure 3(b)). Since $a$ is closer to a prototype of a different class, $a$ enters the condensing set forming a new prototype the weight of which is set to one (Figure 3(c)).

## 3. The AIB2-V2 Algorithm

One may claim that dRHC and AIB2 are also appropriate for data streams and large datasets. This is not true since the repetitive generation of new prototypes is likely to lead to a very large condensing set, one that is inappropriate for instance-based classification. This fact renders dRHC and AIB2 inapplicable, especially on infinite data streams [13]. Therefore, there is a need of a mechanism for the size of the condensing set to remain compact. The dRHC-V2 algorithm is seen to handle large datasets and data streams, yet its prototype ranking stage may render its use problematic especially in cases of very fast (i.e., high velocity) data streams. Even a quick sort procedure may not cope with very fast rates of instance arrivals. Therefore, dRHC-V2 may involve a large queue of data segments that await their turn for processing. In case of high speed data streaming, prototypes ranking should be avoided.

Yet another weakness of dRHC-V2 is that it first appends a set of new prototypes to an already existing condensing set and it then removes the prototypes with the lowest

$AnA$ values (a.k.a. the weakest prototypes). Since the number of new prototypes into the condensing set during each *condensing set update* stage is not known beforehand, the reverse cannot be implemented. Thus, the size of the condensing set may temporarily exceed the size of available memory, an issue to be taken into consideration when setting the $T$ parameter for dRHC-V2.

AIB2-V2 is a simple variation of AIB2 that successfully addresses the stated dRHC-V2 drawbacks. Like AIB2, AIB2-V2 does not consider the new data as data segments. Each new instance is considered individually. During each iteration of the algorithm, an existing prototype either gets updated or a new prototype is generated. Like in dRHC-V2, AIB2-V2 is given a $T$ threshold value. When the generation of a new prototype results into a condensing set whose size exceeds $T$, the weakest prototype gets discarded. Since, only one prototype is discarded, there is no need for the algorithm to rank the prototypes. When a prototype need be removed to retain the size of the condensing set within the set $T$ value, AIB2-V2 calculates the $AnA$ value of each prototype and removes the weakest one. One pass over the prototypes set suffices. In case of a tie involving two or more prototypes having the same (lowest) $AnA$ value, the oldest one gets removed.

At each one point in time (clock tick), one instance arrives and it is examined against the existing condensing set. There exist no data segments in the logic of the AIB2-V2 algorithm. A clock tick corresponds to incrementing the time value by 1. The newly generated prototype is assigned the current time value. A prototype created at "time" $r$ is of age $t - r$ when the current time is $t$. Hence, the value of $AnA$ is calculated as follows:

$$AnA = \frac{w}{t - r + 1}$$

with $w$ being the given prototype's weight.

The pseudocode in Algorithm 1 presents the AIB2-V2 algorithm. Like AIB2, when an instance $inst$ from the training set $TS$ has the same class as its nearest prototype $nn$ in the current condensing set $CS$, $nn$'s attributes are updated by taking into consideration its current weight and the attributes of $inst$. Of course, since $inst$ is from now on to be represented by $nn$, the weight of $nn$ is increased by 1 (lines 32–35). The difference between AIB2-V2 and AIB2 is depicted in lines 16–22 and 27–29. The algorithm starts to remove prototypes when $|CS| > T$. The for-loop in lines 12–23 performs the single pass over the prototypes and finds the nearest prototype $nn$. Simultaneously, it computes the corresponding $AnA$ values and marks the prototype $p\_min$ with the lowest $AnA$ value. Next, if $inst$ enters $CS$, $p\_min$ is removed from $CS$ (line 28). $p\_min$ is retrieved only when the size of the condensing set exceeds the set $T$ threshold value. Contrary to the ranking procedure executed by dRHC-V2 for each data segment, here, there is no extra cost for finding the prototype with the lowest $AnA$. It is retrieved by the single pass over the prototypes. It is worth noting that the new prototype enters at the end of $CS$. Thus, the prototypes in $CS$ are stored sorted by their age values, from the oldest to the newest. Therefore, in case of ties, $p\_min$ is the oldest prototype with the lowest $AnA$ value.

As already stated, both dRHC and AIB2 are not noise-tolerant. In the case of their -V2 variations, it is noted that noisy prototypes usually involve low $AnA$ values and they are amongst the first candidates for removal from the condensing set. In this respect, it may safely be assumed that both dRHC-V2 and AIB2-V2 are noise tolerant algorithms, provided that in the course of condensing set construction the pre-specified $T$ value is exceeded for the algorithms to start removing prototypes from the condensing set. This

**Algorithm 1** AIB2-V2

```
Input: $TS, T$ Output: $CS$
1: $time \leftarrow 1$
2: $CS \leftarrow \varnothing$
3: move first instance $inst$ of $TS$ to $CS$
4: $inst_{weight} \leftarrow 1$
5: $inst_r \leftarrow time$
6: while there exist instances in $TS$ do
7:     $time \leftarrow time + 1$
8:     $inst \leftarrow$ next instance in $TS$
9:     $minimum\_AnA \leftarrow \infty$
10:    $p\_min \leftarrow NULL$
11:    $nn \leftarrow NULL$
12:    for each prototype $p \in CS$ do
13:        if $p$ is the nearest prototype of $inst$ then
14:            $nn \leftarrow p$
15:        end if
16:        if $|CS| > T$ then
17:            $p_{AnA} \leftarrow \frac{p_{weight}}{time - p_r + 1}$
18:            if $p_{AnA} < minimum\_AnA$ then
19:                $minimum\_AnA \leftarrow p_{AnA}$
20:                $p\_min \leftarrow p$
21:            end if
22:        end if
23:    end for
24:    if $nn_{class} \neq inst_{class}$ then
25:        $inst_{weight} \leftarrow 1$
26:        $inst_r \leftarrow time$
27:        if $|CS| > T$ then
28:            $CS \leftarrow CS - \{p\_min\}$
29:        end if
30:        $CS \leftarrow CS \cup \{inst\}$
31:    else
32:        for each attribute $attr(i)$ of $nn$ do
33:            $nn_{attr(i)} \leftarrow \frac{nn_{attr(i)} \times nn_{weight} + inst_{attr(i)}}{nn_{weight} + 1}$
34:        end for
35:        $nn_{weight} \leftarrow nn_{weight} + 1$
36:    end if
37:    $TS \leftarrow TS - \{inst\}$
38: end while
39: return $CS$
```

is yet one more issue to be taken into consideration when setting the value of the $T$ parameter. Setting $T$ to a high value effectively disables the noise tolerant character of the algorithm, and setting it too low ends up in having the algorithm remove useful prototypes from the condensing set, especially in cases when the size of the latter is relatively small.

## 4. Performance Evaluation

### 4.1. Experimental Setup

The performance of AIB2-V2 was tested against dRHC-V2, dRHC and AIB2 using fourteen well-known and widely-used datasets distributed by the KEEL repository[4] [4]. Their profile is summarized in Table 1. It is worth mentioning that in [19] and [18], dRHC and AIB2 were evaluated against five state-of-the-art DRTs by utilizing the same fourteen datasets. More specifically, dRHC and AIB2 were experimentally evaluated against the CNN-rule [11], IB2 [3,2], PSC [14,15], ENN-rule [26] and RSP3 [22]. In this respect, when dRHC-V2 and AIB2-V2 are compared to dRHC and AIB2, they are "indirectly" compared to these five data reduction techniques. We did not include the PS and PG algorithms reviewed in [21] because they either focus on concept drift detection or introduce high computational cost. Consequently, they are inappropriate to be compared against AIB2-V2 and dRHC-V2. It is worth mentioning that the reader may execute all dRHC-V2, AIB2-V2, their predecessors and the aforementioned algorithms using the stated datasets at the publicly accessible WebDR[5]. environment [20].

**Table 1.** Dataset description

| Dataset | Size | Attributes | Classes |
|---|---|---|---|
| Letter Image Recognition (LIR) | 20,000 | 16 | 26 |
| Magic G. Telescope (MGT) | 19,020 | 10 | 2 |
| Pen-Digits (PD) | 10,992 | 16 | 10 |
| Landsat Satellite (LS) | 6,435 | 36 | 6 |
| Shuttle (SH) | 58,000 | 9 | 7 |
| Texture (TXR) | 5,500 | 40 | 11 |
| Phoneme (PH) | 5,404 | 5 | 2 |
| Balance (BL) | 625 | 4 | 3 |
| Pima (PM) | 768 | 8 | 2 |
| Ecoli (ECL) | 336 | 7 | 8 |
| Yeast (YS) | 1,484 | 8 | 10 |
| Twonorm (TN) | 7,400 | 20 | 2 |
| MONK 2 (MN2) | 432 | 6 | 2 |
| KddCup (KDD) | 141,481 | 36 | 23 |

[4] `http://sci2s.ugr.es/keel/datasets.php`
[5] `https://atropos.uom.gr/webdr`

The LIR, PD, SH, PH, TXR datasets are all noise-free. It is expected that both dRHC-V2 and AIB2-V2 will perform better on noisy datasets by achieving higher classification accuracy than dRHC and AIB2, respectively. Thus, we built two additional datasets with noise by introducing a 10% of random noise to the PD and LS datasets. The noise was introduced by setting 10% of the training instances to a randomly chosen (different) class label. We refer to these datasets as PDN and LSN, respectively.

The algorithms were coded in C, adopting the Euclidean distance as the distance metric. We randomized the datasets that were distributed sorted on the class label. The KDD dataset involves a large number of duplicates, their ranges varying widely. We removed duplicates and normalized the attributes to the $[0, 1]$ range. Furthermore, we removed the nominal and the fixed-value attributes that exist in KDD. No other transformation was applied to all other datasets. For each one dataset and algorithm, Accuracy (ACC), Reduction Rate (RR), and Preprocessing Cost (PC) in terms of distance computations were calculated. The average values of the three reported measurements were obtained via five-fold-cross-validation. As expected, the higher the reduction rates, the fewer distances are computed during the classification step and, as a consequence, the lower is the computational cost introduced by $k$-NN classifier. Therefore, we did not measure the cost of the classification process.

Since the concept drift phenomenon is not present in the datasets used, we did not implement any special evaluation method for data streams, like test-then-train [8]. Classification accuracy was estimated by running $k$-NN classification with $k = 1$. The PC cost measurements conducted do not include the cost overhead introduced by the prototypes ranking in the case of dRHC2-V2. The cost of ranking is $\mathcal{O}(n \log n)$ on average when a quicksort approach is used. When the threshold is reached, dRHC-V2 ranks the prototypes after the arrival of each data segment. AIB2-V2 avoids the cost of ranking.

Both dRHC and dRHC-V2 consider data in data segments. Datasets are split into data segments of a specific size. Table 2 lists the segment size and the number of segments used for each dataset. It is worth noting that the experiments conducted empirically in [19] reveal that the size of the data segment does not influence the performance of dRHC. Hence, the experimental measurements reported do not involve different segment sizes.

Both AIB2-V2 and dRHC-V2 utilize the $T$ threshold value, that is a ceiling value for the condensing set size. If the condensing set size exceeds $T$, both algorithms remove prototypes from the condensing set in order to maintain its size equal to $T$. A number of runs were conducted, with $T$ assuming a value from a set of percentages of the size of the condensing set constructed by dRHC, namely 85%, 70%, 55%, and 40%. This is done in order to be fair when comparing the two algorithms, since dRHC and and AIB2 achieve similar data reduction rates. Table 2 lists the $T$ values used.

## 4.2. Experimental Results

Two types of experiments were conducted. The first type focuses on measuring the performance, following the arrival of all data. The second type measures the data reduction rate (i.e., the size of the condensing set) and the classification accuracy achieved following the arrival and the processing of each one data segment.

Table 3 lists the performance measurements conducted following the arrival of all data segments. Table 3 lists the accuracy achieved by the 1-NN classifier operating on

**Table 2.** Segment size and $T$ parameter values

| Dataset | Segment size | Segments | CS Size $T = 85\%$ | CS Size $T = 70\%$ | CS Size $T = 55\%$ | CS Size $T = 40\%$ |
|---|---|---|---|---|---|---|
| LIR | 2,000 | 8 | 1,608 | 1,324 | 1.040 | 757 |
| MGT | 1.902 | 8 | 3,283 | 2,703 | 2,124 | 1,545 |
| PD | 1,000 | 9 | 207 | 171 | 134 | 97 |
| LS | 572 | 9 | 510 | 420 | 330 | 240 |
| SH | 1,856 | 25 | 197 | 162 | 128 | 93 |
| TXR | 440 | 10 | 189 | 156 | 122 | 89 |
| PH | 500 | 9 | 649 | 534 | 420 | 305 |
| BL | 100 | 5 | 93 | 77 | 60 | 44 |
| PM | 100 | 7 | 182 | 150 | 118 | 86 |
| ECL | 100 | 3 | 71 | 58 | 46 | 33 |
| YS | 396 | 3 | 492 | 405 | 318 | 232 |
| TN | 592 | 10 | 233 | 192 | 151 | 110 |
| MN2 | 115 | 3 | 9 | 8 | 6 | 4 |
| KDD | 4,000 | 29 | 752 | 620 | 487 | 354 |
| PDN | 1,000 | 9 | 2,072 | 1,706 | 1,341 | 975 |
| LSN | 572 | 9 | 1,249 | 1,029 | 808 | 588 |

the condensing set of each algorithm. The best results are highlighted in boldface. Moreover, Table 3 lists the reduction rates achieved by the algorithms and the corresponding preprocessing costs in terms of millions of distance computations. Although, DRTs are adopted when the conventional $k$-NN classifier cannot be applied due to its limitations, for reference, Table 3 presents the accuracy measurements achieved by applying $k$-NN on the original complete training set.

The accuracy measurements for AIB2-V2 are quite promising (see in Table 3). In cases of noise-free datasets, the $k$-NN classifier that operates on condensing sets built by AIB2-V2 achieves accuracy values comparable to those of the dRHC and AIB2 algorithms at a lower (classification stage) computational cost, and at a lower preprocessing cost. In cases of datasets that contain noise, the gain is higher. AIB2-V2 and dRHC-V2 are measured to achieve higher classification accuracy than dRHC and AIB2. This happens because noisy data originating prototypes relate to low importance (i.e. $AnA$) values and as such they are removed from the condensing set. Hence, in cases of datasets with a high level of noise (e.g., MGT, PDN, LSN, BL), AIB2-V2 and dRHC-V2 with the lowest $T$ value were found to achieve even higher accuracy than the conventional $k$-NN classifier. It is noted that the lower is the $T$ value used, the higher the reduction rate and the lower the preprocessing cost achieved. Since AIB2-V2 and dRHC-V2 adopt a ceiling value for the maximum condensing set size and maintain it throughout the whole execution, the reduction rate (RR) and processing cost (PC) results obtained were as expected; the size of the condensing set and the processing cost increase until the set maximum value $T$ is reached and then they remain constant, for all the $T$ values used.

It is not clear which of AIB2-V2 and dRHC-V2 are more accurate. In Table 3, dRHC-V2 is seen to construct its condensing set by computing fewer distances than AIB2-V2.

However, it ranks the prototypes and it may be problematic in cases of very fast data streams. Also, the average measurements (AVG) depicted in the last row of Table 3 suggest that dRHC-V2 and AIB2-V2 can achieve even better accuracy than their predecessors by avoiding the arbitrary growth in size of the condensing set, and by reducing the preprocessing cost.

Tables 3 lists performance rates measured after the arrival of the last data segment, i.e., when all data are processed. An additional set of experiments conducted involved the measuring of accuracy and condensing set sizes following the processing of each one data segment by executing the algorithms on ten datasets. With the exception of the Shuttle (SH) dataset, the largest datasets were used for this set of experimental runs. In the case of the Shuttle (SH) dataset accuracy does not present an essential variance from one data segment to another and for this reason the dataset was excluded. For AIB2-V2, whereby processing does not involve the use of data segments, accuracy values were measured utilizing condensing set snapshots as they were following the processing of $s$ prototypes, where $s$ is the "Segment size" value listed in Table 2.

Figures 4– 13 present the results obtained from this new set of experiments. Each figure involves two diagrams. Diagram (a) plots the size of the condensing set following the arrival of each one data segment (numbered 1,2,...) on the $x$ axis. The size of the condensing set is seen to rise and start to level off when $T$ is reached. Diagram (b) plots the accuracy achieved following the processing of each (subsequent) data segment. It is worth noting that for the MGT, PDN and LSN datasets which involve a relatively high level of noise, the classification accuracy increases considerably when the prototype removing mechanism is enabled. Thus algorithms that adopt low $T$ values perform better in cases of noisy datasets. In general, we observe accuracy tends to rise relatively fast for the first few data segments that are being processed and it then tends to level off either gradually (e.g. for LS, PH) or a bit more abruptly (LIR, PD, TXR and KDD).

## 4.3. Wilcoxon Signed Rank Test results

The experimental results obtained are herewith complemented by the Wilcoxon signed rank test results [7,23] in order to statistically confirm the validity of the ACC measurements presented in Table 3. The Wilcoxon signed rank test compares all the algorithms in pairs, considering the accuracy achieved against each one dataset. All four versions of dRHC(-V2) and AIB2(-V2) were considered, using a number of $T$ values. Since both dRHC-V2 and AIB2-V2 dominate in terms of the RR and PC results obtained, there was no need to include the corresponding measures in the test.

Table 4 presents the Wilcoxon signed rank test results obtained. The column labeled "w/l/t" lists the number of wins, losses and ties for each one comparison test. The column labeled "Wilcoxon" value (last column) lists a figure that quantifies the significance of the measured difference between the two algorithms compared. When it is lower than 0.05, the difference is statistically significant.

Despite the fact that AIB2-V2 is seen to involve more wins over dRHC-V2, the results in Table 3 indicate that there exists no statistically significant difference between the two algorithms. Furthermore, there is no statistical difference between AIB2-V2 and AIB2. In the case of dRHC-V2 and dRHC, the former is seen to outperform the latter when $T$=85%, and when $T$=70%. There is no statistical difference between the two when $T$=55%, and when $T$=40%. The results obtained reveal that both dRHC-V2 and AIB2-V2 can be used

**Table 3.** Comparisons in terms of Accuracy (ACC(%)), Reduction Rates (RR(%)) and Preprocessing Cost(M)

| Data | *T*% | 1NN | dRHC | dRHC-V2 | AIB2 | AIB2-V2 |
|---|---|---|---|---|---|---|
| LIR | 85: | 95.83 | 93.920 | 93.530 | **94.145** | 93.775 |
| | 70: | | | 92.835 | | 92.750 |
| | 55: | | | 91.660 | | 91.440 |
| | 40: | | | 88.845 | | 89.075 |
| MGT | 85: | 78.14 | 72.965 | 74.606 | 73.286 | 75.268 |
| | 70: | | | 75.447 | | 75.662 |
| | 55: | | | 75.920 | | 75.994 |
| | 40: | | | **76.393** | | 76.236 |
| PD | 85: | 99.35 | 98.490 | **98.508** | 98.326 | 98.226 |
| | 70: | | | 98.235 | | 98.062 |
| | 55: | | | 97.953 | | 97.316 |
| | 40: | | | 97.471 | | 96.543 |
| PDN | 85: | 89.52 | 75.864 | 81.832 | 77.265 | 84.125 |
| | 70: | | | 85.862 | | 86.672 |
| | 55: | | | 89.065 | | 88.719 |
| | 40: | | | **91.394** | | 90.175 |
| LS | 85: | 90.60 | 88.500 | 88.671 | 89.417 | **89.573** |
| | 70: | | | 88.920 | | 89.402 |
| | 55: | | | 88.687 | | 88.873 |
| | 40: | | | 88.314 | | 88.733 |
| LSN | 85: | 81.99 | 76.566 | 79.487 | 77.653 | 81.663 |
| | 70: | | | 81.974 | | 82.828 |
| | 55: | | | 83.357 | | 83.807 |
| | 40: | | | 84.646 | | **84.880** |
| SH | 85: | 99.82 | 99.695 | 99.667 | **99.726** | 99.671 |
| | 70: | | | 99.624 | | 99.621 |
| | 55: | | | 99.590 | | 99.593 |
| | 40: | | | 99.400 | | 99.243 |
| TXR | 85: | 99.02 | 97.600 | 97.291 | **97.691** | 97.400 |
| | 70: | | | 98.836 | | 96.909 |
| | 55: | | | 96.218 | | 96.255 |
| | 40: | | | 95.182 | | 94.982 |
| PH | 85: | 90.10 | 85.381 | **85.667** | 85.067 | 85.474 |
| | 70: | | | 85.104 | | 85.363 |
| | 55: | | | 84.955 | | 84.327 |
| | 40: | | | 83.845 | | 83.623 |
| BL | 85: | 78.40 | 70.560 | 74.080 | 68.480 | 77.600 |
| | 70: | | | 77.600 | | 78.080 |
| | 55: | | | 79.840 | | 79.520 |
| | 40: | | | **81.790** | | 77.760 |
| PM | 85: | 68.36 | 63.925 | 66.792 | 67.321 | 66.796 |
| | 70: | | | 67.572 | | 64.972 |
| | 55: | | | **69.913** | | 67.445 |
| | 40: | | | 67.184 | | 68.481 |
| ECL | 85: | 79.78 | 71.462 | 74.732 | 72.963 | 72.963 |
| | 70: | | | 76.216 | | 74.460 |
| | 55: | | | 75.316 | | 71.185 |
| | 40: | | | **77.094** | | 70.597 |
| YS | 85: | 52.02 | 48.379 | 49.256 | 48.247 | 49.258 |
| | 70: | | | 49.864 | | 49.932 |
| | 55: | | | 50.743 | | 49.594 |
| | 40: | | | **51.415** | | 49.662 |
| TN | 85: | 94.88 | 93.081 | 93.838 | 93.054 | 94.865 |
| | 70: | | | 94.514 | | 95.203 |
| | 55: | | | 94.986 | | 95.351 |
| | 40: | | | 95.459 | | **95.568** |
| MN2 | 85: | 90.51 | **97.680** | 96.517 | 93.293 | 91.890 |
| | 70: | | | 95.589 | | 86.792 |
| | 55: | | | 91.190 | | 81.489 |
| | 40: | | | 80.281 | | 81.251 |
| KDD | 85: | 99.71 | 99.424 | 99.449 | 99.414 | **99.469** |
| | 70: | | | 99.464 | | 99.444 |
| | 55: | | | 99.443 | | 99.443 |
| | 40: | | | 99.353 | | 99.309 |
| AVG | 85: | 86.75 | 83.343 | 84.620 | 83.459 | 84.876 |
| | 70: | | | 85.479 | | 84.760 |
| | 55: | | | **85.552** | | 84.397 |
| | 40: | | | 84.879 | | 84.132 |

| dRHC | | dRHC-V2 | | AIB2 | | AIB2-V2 | |
|---|---|---|---|---|---|---|---|
| RR | PC | RR | PC | RR | PC | RR | PC |
| 88.18 | 19.574 | 89.95 | 19.179 | 88.14 | 20.098 | 89.95 | 19.387 |
| | | 91.73 | 17.721 | | | 91.73 | 17.488 |
| | | 93.50 | 15.362 | | | 93.50 | 14.820 |
| | | 95.27 | 12.293 | | | 95.27 | 11.337 |
| 74.62 | 26.025 | 78.42 | 25.851 | 71.87 | 33.079 | 78.42 | 31.259 |
| | | 82.24 | 24.414 | | | 82.24 | 28.551 |
| | | 86.04 | 21.709 | | | 86.04 | 24.648 |
| | | 89.85 | 17.730 | | | 89.85 | 19.513 |
| 97.23 | 1.438 | 97.65 | 1.410 | 97.19 | 1.381 | 97.65 | 1.336 |
| | | 98.06 | 1.317 | | | 98.06 | 1.234 |
| | | 98.48 | 1.155 | | | 98.48 | 1.057 |
| | | 98.90 | 0.925 | | | 98.90 | 0.811 |
| 72.28 | 9.502 | 76.44 | 9.407 | 70.04 | 11.572 | 76.44 | 11.045 |
| | | 80.60 | 8.908 | | | 80.60 | 10.128 |
| | | 84.75 | 7.964 | | | 84.75 | 8.786 |
| | | 88.91 | 6.504 | | | 88.91 | 6.998 |
| 88.35 | 1.531 | 90.09 | 1.511 | 86.77 | 1.916 | 90.09 | 1.785 |
| | | 91.84 | 1.422 | | | 91.84 | 1.615 |
| | | 93.59 | 1.261 | | | 93.5 | 1.378 |
| | | 95.34 | 1.029 | | | 95.34 | 1.075 |
| 71.44 | 3.511 | 75.74 | 3.477 | 68.28 | 4.349 | 75.74 | 4.101 |
| | | 80.01 | 3.288 | | | 80.01 | 3.737 |
| | | 84.31 | 2.932 | | | 84.31 | 3.213 |
| | | 88.58 | 2.390 | | | 88.58 | 2.532 |
| 99.50 | 7.977 | 99.58 | 7.614 | 99.45 | 8.041 | 99.58 | 7.420 |
| | | 99.65 | 6.911 | | | 99.65 | 6.540 |
| | | 99.72 | 5.946 | | | 99.72 | 5.453 |
| | | 99.80 | 4.729 | | | 99.80 | 4.127 |
| 94.95 | 0.685 | 95.71 | 0.669 | 94.91 | 0.660 | 95.71 | 0.637 |
| | | 96.46 | 0.617 | | | 96.46 | 0.575 |
| | | 97.23 | 0.533 | | | 97.23 | 0.481 |
| | | 97.98 | 0.429 | | | 97.98 | 0.369 |
| 82.34 | 1.638 | 84.99 | 1.615 | 81.50 | 1.883 | 84.99 | 1.808 |
| | | 87.65 | 1.515 | | | 87.65 | 1.651 |
| | | 90.29 | 1.333 | | | 90.29 | 1.420 |
| | | 92.95 | 1.077 | | | 92.95 | 1.119 |
| 78.12 | 0.029 | 81.40 | 0.029 | 70.56 | 0.037 | 81.40 | 0.032 |
| | | 84.60 | 0.027 | | | 84.60 | 0.028 |
| | | 88.00 | 0.025 | | | 88.00 | 0.024 |
| | | 91.20 | 0.021 | | | 91.20 | 0.018 |
| 65.11 | 0.064 | 70.41 | 0.063 | 64.75 | 0.062 | 70.41 | 0.060 |
| | | 75.61 | 0.060 | | | 75.61 | 0.056 |
| | | 80.81 | 0.055 | | | 80.81 | 0.049 |
| | | 86.02 | 0.046 | | | 86.02 | 0.040 |
| 68.92 | 0.015 | 73.61 | 0.015 | 68.70 | 0.011 | 73.61 | 0.011 |
| | | 78.44 | 0.015 | | | 78.44 | 0.011 |
| | | 82.90 | 0.014 | | | 82.90 | 0.009 |
| | | 87.73 | 0.013 | | | 87.73 | 0.007 |
| 51.23 | 0.306 | 58.59 | 0.306 | 47.10 | 0.366 | 58.59 | 0.349 |
| | | 65.91 | 0.306 | | | 65.91 | 0.321 |
| | | 73.23 | 0.278 | | | 73.23 | 0.278 |
| | | 80.47 | 0.244 | | | 80.47 | 0.222 |
| 95.37 | 0.695 | 96.06 | 0.688 | 93.47 | 1.080 | 96.06 | 0.917 |
| | | 96.76 | 0.654 | | | 96.76 | 0.822 |
| | | 97.45 | 0.590 | | | 97.45 | 0.701 |
| | | 98.14 | 0.495 | | | 98.14 | 0.551 |
| 96.88 | 0.004 | 97.63 | 0.004 | 93.18 | 0.005 | 97.63 | 0.003 |
| | | 97.80 | 0.004 | | | 97.80 | 0.003 |
| | | 98.27 | 0.004 | | | 98.27 | 0.002 |
| | | 98.84 | 0.004 | | | 98.84 | 0.001 |
| 99.22 | 54.70 | 99.34 | 53.555 | 99.21 | 58.705 | 99.34 | 56.947 |
| | | 99.45 | 49.811 | | | 99.45 | 52.493 |
| | | 99.57 | 43.476 | | | 99.57 | 45.319 |
| | | 99.69 | 34.603 | | | 99.69 | 35.609 |
| 82.73 | 7.981 | 85.35 | 7.837 | 68.28 | 8.952 | 85.35 | 8.569 |
| | | 87.92 | 7.312 | | | 87.92 | 7.828 |
| | | 90.51 | 6.415 | | | 90.51 | 6.727 |
| | | 93.10 | 5.158 | | | 93.10 | 5.271 |

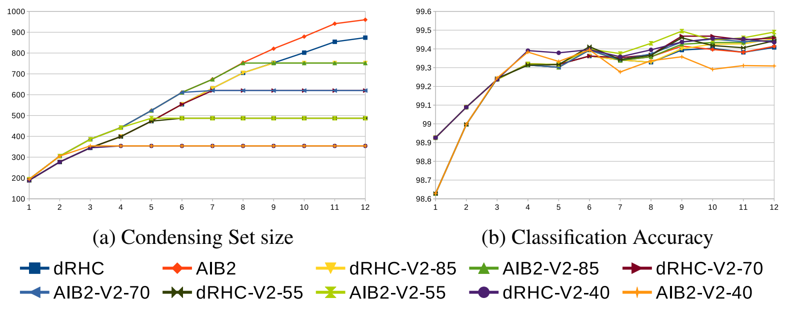


**Fig. 4.** LIR: Condensing Set Size and Classification Accuracy per data segment

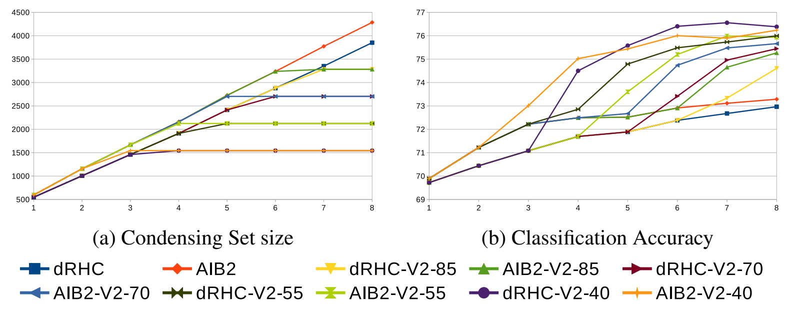


**Fig. 5.** MGT: Condensing Set Size and Classification Accuracy per data segment

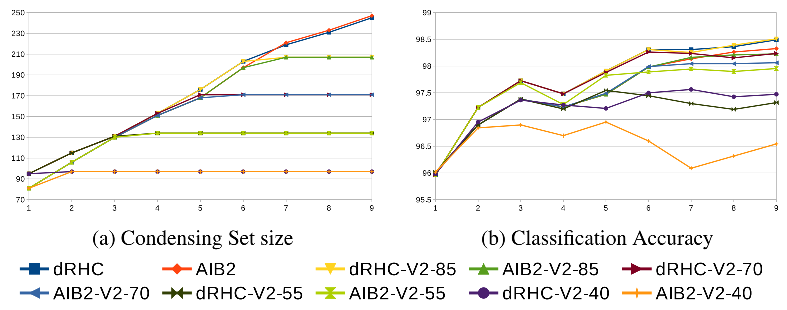
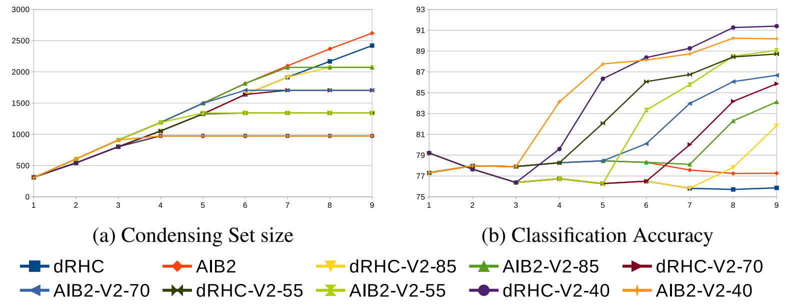
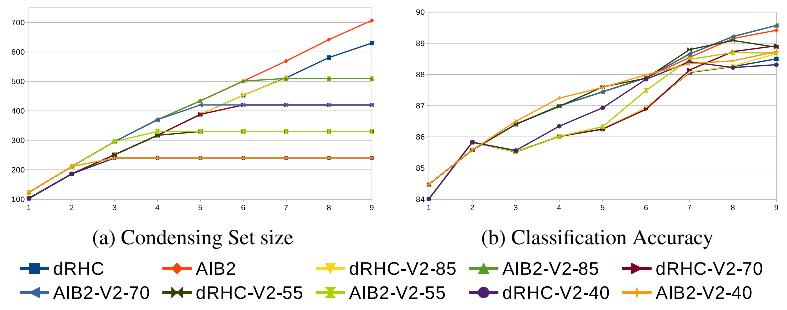


**Fig. 6.** PD: Condensing Set Size and Classification Accuracy per data segment



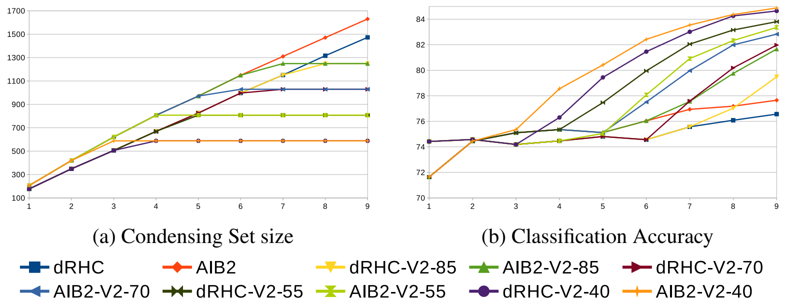
(a) Condensing Set size (b) Classification Accuracy

**Fig. 7.** PDN: Condensing Set Size and Classification Accuracy per data segment



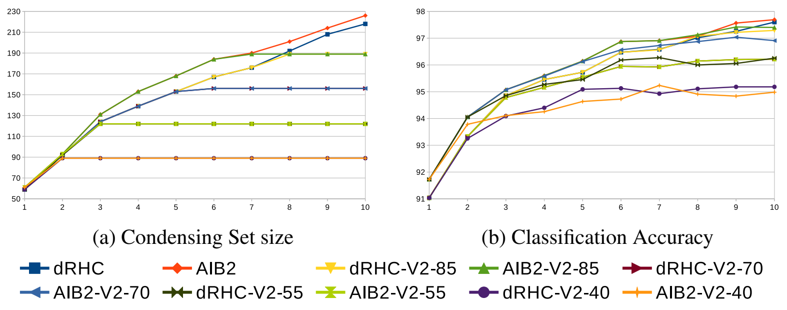
(a) Condensing Set size (b) Classification Accuracy

**Fig. 8.** LS: Condensing Set Size and Classification Accuracy per data segment



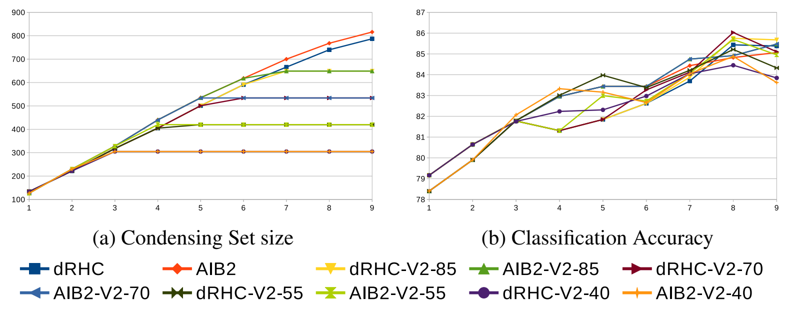
(a) Condensing Set size (b) Classification Accuracy

**Fig. 9.** LSN: Condensing Set Size and Classification Accuracy per data segment



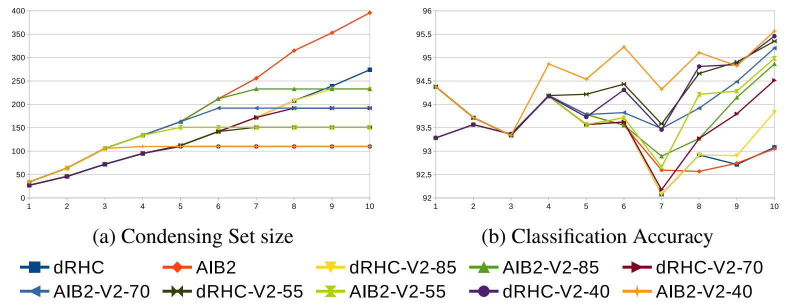
(a) Condensing Set size (b) Classification Accuracy

**Fig. 10.** TXR: Condensing Set Size and Classification Accuracy per data segment



(a) Condensing Set size (b) Classification Accuracy

**Fig. 11.** PH: Condensing Set Size and Classification Accuracy per data segment



(a) Condensing Set size (b) Classification Accuracy

**Fig. 12.** TN: Condensing Set Size and Classification Accuracy per data segment

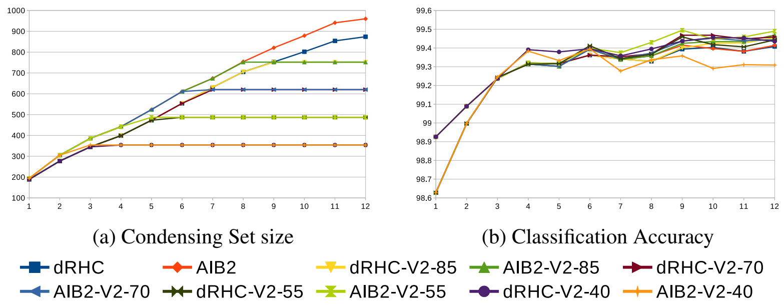


**Fig. 13.** KDD: Condensing Set Size and Classification Accuracy per data segment

instead of dRHC and AIB2 without loss of accuracy when there is need for a condensing set with a fixed size.

**Table 4.** Results of Wilcoxon signed rank test

| **Methods** | **Accuracy** | |
|---|---|---|
| | **w/l/t** | **Wilcoxon** |
| dRHC vs dRHC-V2 ($T$=85%) | 4/12/0 | 0.030 |
| dRHC vs dRHC-V2 ($T$=70%) | 5/11/0 | 0.026 |
| dRHC vs dRHC-V2 ($T$=55%) | 6/10/0 | 0.121 |
| dRHC vs dRHC-V2 ($T$=40%) | 8/8/0 | 0.326 |
| AIB2 vs AIB2-V2 ($T$=85%) | 6/9/1 | 0.132 |
| AIB2 vs AIB2-V2 ($T$=70%) | 7/9/0 | 0.255 |
| AIB2 vs AIB2-V2 ($T$=55%) | 8/8/0 | 0.756 |
| AIB2 vs AIB2-V2 ($T$=40%) | 7/9/0 | 0.836 |
| dRHC-V2 ($T$=85%) vs AIB2-V2 ($T$=85%) | 4/12/0 | 0.179 |
| dRHC-V2 ($T$=70%) vs AIB2-V2 ($T$=70%) | 8/8/0 | 0.918 |
| dRHC-V2 ($T$=55%) vs AIB2-V2 ($T$=55%) | 6/9/0 | 0.061 |
| dRHC-V2 ($T$=40%) vs AIB2-V2 ($T$=40%) | 6/10/0 | 0.352 |
| dRHC vs AIB2 | 7/9 | 0.408 |

## 5. Conclusion and Directions for Further Work

A new noise-tolerant prototype generation algorithm is considered in detail. It maintains a fixed size condensing set by monitoring a stream of training data, or by managing a large dataset that cannot fit in memory. The new algorithm is code-named AIB2-V2 and has some common characteristics with dRHC-V2. Both comprise improved variations of the AIB2 and dRHC algorithms, respectively. Contrary to dRHC-V2, AIB2-V2 avoids ranking when replacing its prototypes. When a new prototype enters the condensing set and the threshold limit has been reached, the pre-marked prototype with the lowest $AnA$ is removed. The experimental study yields promising results. Even when the condensing set generated by the new algorithm is less than half the size of that generated by the AIB2 algorithm, there is no loss in accuracy and in many cases the accuracy achieved by AIB2-V2 is even higher. By having the size of the condensing set to vary up to a pre-specified maximum value $T$ and practically remain constant, the preprocessing costs involved remain low and constant throughout the execution of each one of the two algorithms. Moreover, the latter can be implemented to execute on devices with limited memory.

Suggested directions for future work include the development of new variations of noise-tolerant prototype generation algorithms that will fully exploit the potential of handling data streams involving the concept drift. This could be achieved by further increasing the value of the importance measure for newly generated prototypes, next to that of old prototypes involving static attribute values in the course of time. Moreover, we also plan to introduce parallelism to the DRTs, in order to speed up the construction of the condensing set.

## References

1. Aggarwal, C.: Data Streams: Models and Algorithms. Advances in Database Systems Series, Springer Science+Business Media, LLC (2007)
2. Aha, D.W.: Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. Int. J. Man-Mach. Stud. 36(2), 267–287 (Feb 1992), http://dx.doi.org/10.1016/0020-7373(92)90018-G
3. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. 6(1), 37–66 (Jan 1991), http://dx.doi.org/10.1023/A:1022689900470
4. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S.: 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)
5. Beringer, J., Hüllermeier, E.: Efficient instance-based learning on data streams. Intell. Data Anal. 11(6), 627–650 (Dec 2007), http://dl.acm.org/citation.cfm?id=1368018.1368022
6. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theor. 13(1), 21–27 (Sep 2006), http://dx.doi.org/10.1109/TIT.1967.1053964
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (Dec 2006), http://dl.acm.org/citation.cfm?id=1248547.1248548
8. Gama, J.a., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 329–338. KDD '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1557019.1557060
9. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. 34(3), 417–435 (Mar 2012), http://dx.doi.org/10.1109/TPAMI.2011.142
10. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems, Elsevier Science (2011)
11. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory 14(3), 515–516 (1968)
12. James, M.: Classification algorithms. Wiley-Interscience, New York, NY, USA (1985)
13. Lallouet, A., Law, Y.C., Lee, J.H., Siu, C.F.: Constraint Programming on Infinite Data Streams. In: Walsh, T. (ed.) International Joint Conference on Artificial Intelligence.

pp. 597–604. Barcelone, Spain (Jul 2011), `https://hal.archives-ouvertes.fr/hal-01009693`

14. Olvera-Lopez, J.A., Carrasco-Ochoa, J.A., Trinidad, J.F.M.: A new fast prototype selection method based on clustering. Pattern Anal. Appl. 13(2), 131–141 (2010)
15. Olvera-Lpez, J.A., Carrasco-Ochoa, J.A., Trinidad, J.F.M.: Object selection based on clustering and border objects. In: Kurzynski, M., Puchala, E., Wozniak, M., Zolnierek, A. (eds.) Computer Recognition Systems 2, Advances in Soft Computing, vol. 45, pp. 27–34. Springer (2008)
16. Ougiaroglou, S., Arampatzis, G., Dervos, D.A., Evangelidis, G.: Generating fixed-size training sets for large and streaming datasets. In: Kirikova, M., Nørvåg, K., Papadopoulos, G.A. (eds.) Advances in Databases and Information Systems. pp. 88–102. Springer International Publishing, Cham (2017)
17. Ougiaroglou, S., Evangelidis, G.: AIB2: An abstraction data reduction technique based on ib2. In: Proceedings of the 6th Balkan Conference in Informatics. pp. 13–16. BCI '13, ACM, New York, NY, USA (2013), `http://doi.acm.org/10.1145/2490257.2490260`
18. Ougiaroglou, S., Evangelidis, G.: Efficient data abstraction using weighted IB2 prototypes. Comput. Sci. Inf. Syst. 11(2), 665–678 (2014), `http://dx.doi.org/10.2298/CSIS140212036O`
19. Ougiaroglou, S., Evangelidis, G.: RHC: a non-parametric cluster-based data reduction for efficient k-NN classification. Pattern Analysis and Applications 19(1), 93–109 (2014), `http://dx.doi.org/10.1007/s10044-014-0393-7`
20. Ougiaroglou, S., Evangelidis, G.: WebDR: A web workbench for data reduction. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, vol. 8726, pp. 464–467. Springer Berlin Heidelberg (2014), `http://dx.doi.org/10.1007/978-3-662-44845-8_36`
21. Ramrez-Gallego, S., Krawczyk, B., Garca, S., Woniak, M., Herrera, F.: A survey on data preprocessing for data stream mining. Neurocomput. 239(C), 39–57 (May 2017), `https://doi.org/10.1016/j.neucom.2017.01.078`
22. Sánchez, J.S.: High training set size reduction by space partitioning and prototype abstraction. Pattern Recognition 37(7), 1561–1564 (2004)
23. Sheskin, D.: Handbook of Parametric and Nonparametric Statistical Procedures. A Chapman & Hall book, Chapman & Hall/CRC (2011)
24. Triguero, I., Derrac, J., Garcia, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. Trans. Sys. Man Cyber Part C 42(1), 86–100 (Jan 2012), `http://dx.doi.org/10.1109/TSMCC.2010.2103939`
25. Tsymbal, A.: 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)
26. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Mach. Learn. 38(3), 257–286 (Mar 2000), `http://dx.doi.org/10.1023/A:1007626913721`

**Stefanos Ougiaroglou** holds a B.Sc in Computer Science (2004) from Alexander TEI of Thessaloniki, Greece, a M.Sc. in Computer Science (2006) from Aristotle University of Thessaloniki, Greece and a PhD in Computer Science (2014) from University of Macedonia, Thessaloniki, Greece. His PhD studies were supported by a Scholarship from State Scholarships Foundation (I.K.Y) of Greece. From 2016, he is postdoctoral researcher at the department of Applied Informatics of University of Macedonia. Moreover, S. Ougiaroglou holds a Certificate of Pedagogical and Teaching Competence (2010),

ASPAITE, Greece. Currently, he is a laboratory lecturer at the department of Information and Electronic Engineering of the International Hellenic University (IHU), Thessaloniki, Greece where he teaches Databases, Data Structures, Data Mining, Algorithms and Programming, Operating Systems, Web Languages and Technologies and Development of Internet Applications. His research interests include Data Mining algorthms, Data streams, Data management for Mobile Computing and Educational Technology. He has authored several journal and conference papers in these fields. Personal webpage: `https://www.iee.ihu.gr/~stoug/`

**Dimitris A. Dervos** (`https://d-a-d.weebly.com`) is with the Department of Information and Electronic Engineering of the International Hellenic University in Thessaloniki, Greece. He holds a BSc degree in Physics and a Ph.D degree in Computer Science from the Aristotle University of Thessaloniki, plus an M.A. degree in Physics, and an M.Sc. degree in Computer Engineering from the University of Southern California. His teaching record includes undergraduate and graduate level topics ranging from Algorithms and Data Structures to Database Technology and Data Mining taught at academic institutions in Greece, and in the U.K. His research interests primarily focus in the areas of Bibliography Data Analysis, and Data Mining. He has led the research team who received the 2005 Thomson ISI / ASIS&T Citation Analysis Research Grant.

**Georgios Evangelidis** is a Professor at the Department of Applied Informatics, School of Information Sciences, University of Macedonia, Thessaloniki, Greece. He is the Director of the Software and Data Engineering Laboratory of the Department since 2016. He holds a BSc in Mathematics (1987, Aristotle University, Thessaloniki), and MSc and PhD in Computer Science (1990 and 1994, Northeastern University, Boston, MA). His research interests and published work (about 120 papers and book chapters) are in the area of Information Management: Databases, Data Mining, Information Retrieval, Analysis of Bibliographic Databases. He teaches Databases, Data Mining and Geographic Information Systems. He has been coordinator and/or participant in more than 20 Greek government and EU funded research and development projects. Personal webpage: `http://users.uom.gr/~gevan/`

*Received: May 18, 2019; Accepted: November 18, 2019.*