AIB2: An Abstraction Data Reduction Technique based on IB2

Stefanos Ougiaroglou[*]
stoug@uom.gr

Georgios Evangelidis
gevan@uom.gr

Department of Applied Informatics
University of Mecedonia
156 Egnatia str., 54006
Thessaloniki, Greece

ABSTRACT

Data reduction improves the efficiency of k-NN classifier on large datasets since it accelerates the classification process and reduces storage requirements for the training data. IB2 is an effective data reduction technique that selects some training items form the initial dataset and uses them as representatives (prototypes). Contrary to many other data reduction techniques, IB2 is a very fast, one-pass method that builds its reduced (condensing) set in an incremental manner. New training data can update the condensing set without the need of the "old" removed items. This paper proposes AIB2, a variation of IB2, which generates new prototypes instead of selecting them. AIB2 attempts to improve the efficiency of IB2 by positioning the prototypes in the center of the data areas they represent. The experimental study shows that the proposed method performs better than IB2.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

k-NN classification, data reduction, prototype selection and abstraction

1. INTRODUCTION

The k-Nearest Neighbour (k-NN) classification algorithm is an effective lazy classifier [6]. For each unclassified item x, it searches in the available training database and retrieves the k nearest items (neighbours) to x according to a distance metric. Then, x is assigned to the class where the most retrieved k nearest neighbours belong to. The k-NN classifier has some properties that make it a popular classifier: (i) it is considered to be an accurate classifier, (ii) it has many applications, (iii) it is easy to implement, and (iv) predictions result from an easy to understand procedure.

On the other hand, the k-NN classifier has two main drawbacks. The first one is the high computational cost involved. Since all distances between the unclassified items and the available training data must be estimated, the computational cost is high, and thus, classification is a time consuming process. In cases of very large datasets, the use of k-NN classifier may be even prohibitive. The second drawback is the high storage requirements for the Training Set (TS). Contrary to the eager classifiers that can discard the training data after the construction of the classification model (e.g., decision trees [9]), k-NN classifier needs all training data available for accessing when a new item has to be classified. These two weaknesses render k-NN classifier inefficient for large datasets.

Multi-attribute indexes [10] can efficiently accelerate the nearest neighbours search and speed-up k-NN classifier. However, storage requirements increase, since in addition to the training data, the index must be stored, too. Moreover, indexes can be applied only on datasets with moderate dimensionality (e.g., 2-10). In higher dimensions, the phenomenon of dimensionality curse makes even sequential scans more effective than indexes, thus, it is essential to first apply a dimensionality reduction technique.

Many Data Reduction Techniques (DRTs) have been proposed to address both weaknesses. Contrary to indexes, they can be used on data with high dimensionality. DRTs pre-process the TS and build a small representative set of it, usually called Condensing Set (CS). The goal is to reduce the storage requirements and computational cost of the classifier and at the same time keep classification accuracy at high levels. DRTs are distinguished into Prototype Selection [7] and Prototype Abstraction [11] algorithms and differ on the way they build their CS. Prototype selection algorithms select items from TS and place them in CS as representatives (prototypes), whereas, prototype abstraction algorithms generate prototypes by summarizing similar

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

TS items. Each prototype in CS represents a specific data area of the multidimensional space.

IB2 belongs to the popular family of Instance-Based Learning (IBL) algorithms [3, 2]. IB2 is an one-pass incremental prototype selection DRT. Actually, it is based on the earliest and well known prototype selection DRT, CNN-rule [8]. Contrary to CNN-rule and many other state-of-the-art DRTs, IB2 can build its CS in an incremental manner. It can take into consideration new TS items after the CS construction and without needing the old removed items. Hence, it can be used in dynamic/streaming environments [1] where new training data is gradually available. In addition, the incremental nature of IB2 allows its execution on training sets that cannot fit into main memory.

In this paper, we attempt to improve the performance of IB2 by considering the idea of prototypes abstraction. Our contribution is the development and evaluation of an abstraction version of IB2, we call Abstraction IB2 (AIB2). The proposed DRT retains all the properties of IB2, but it is faster and achieves higher reduction rates and improved accuracy.

CNN-rule and IB2 are presented in Section 2. Section 3 describes in detail the proposed AIB2 algorithm. Section 4 presents an experimental study where k-NN is applied on the original training sets and the corresponding condensing sets produced by CNN, IB2 and AIB2 on seven datasets. Section 5 concludes the paper.

2. CNN-RULE AND IB2 ALGORITHMS

The Condensing Nearest Neighbour (CNN) rule [8] is the earliest and also a reference prototype selection DRT in the context of k-NN classification. CNN-rule (and many other DRTs) builds its CS based on the following simple idea. Items that lie in the "internal" data area of a class (i.e., far from class decision boundaries) are useless during the classification process. Thus, they can be removed without loss of accuracy. By adopting this idea, CNN-rule tries to place into CS only the items that lie in the close-class-border data areas. These are the only essential items for the classification process.

CNN-rule tries to keep the close-class-border items as follows (see Algorithm 1). Initially, a TS item is moved in CS (line 2). Then, CNN-rule applies the 1-NN rule and classifies the items of TS by scanning the items of CS (line 6). If an item is misclassified, it is moved from TS to CS (lines 7–11). The algorithm continues until there are no moves from TS to CS during a complete scan of TS (line 13). This ensures that the content of TS is correctly classified by the content of CS. The remaining content of TS is discarded (line 17).

CNN-rule considers that misclassified items are probably close to decision boundaries and so they must be included in CS. An advantage of the algorithm is that it is a non-parametric approach. It determines the number of the prototypes automatically, without user-defined parameters. A drawback is that the resulting CS depends on the order that class labels of items enter the CS. Thus, CNN-rule builds a different CS by examining the same training data in a different order.

Algorithm 1 CNN-rule

```
Input: $TS$ Output: $CS$
 1: $CS \leftarrow \varnothing$
 2: pick an item of $TS$ and move it to $CS$
 3: repeat
 4:   $stop \leftarrow TRUE$
 5:   for each $x \in TS$ do
 6:     $NN \leftarrow$ Nearest Neighbour of $x$ in $CS$
 7:     if $NN_{class} \neq x_{class}$ then
 8:       $CS \leftarrow CS \cup x$
 9:       $TS \leftarrow TS - x$
10:       $stop \leftarrow FALSE$
11:     end if
12:   end for
13: until $stop == TRUE$ {no move during a pass of $TS$}
14: discard $TS$
15: return $CS$
```

Algorithm 2 IB2

```
Input: $TS$ Output: $CS$
 1: $CS \leftarrow \varnothing$
 2: pick an item of $TS$ and move it to $CS$
 3: for each $x \in TS$ do
 4:   $NN \leftarrow$ Nearest Neighbour of $x$ in $CS$
 5:   if $NN_{class} \neq x_{class}$ then
 6:     $CS \leftarrow CS \cup x$
 7:   end if
 8:   $TS \leftarrow TS - x$
 9: end for
10: return $CS$
```

Furthermore, CNN-rule cannot handle new training data, i.e., is non-incremental. The algorithm involves multiple passes over the data and it cannot use training data available at a later time to update its CS. To construct an updated CS, CNN-rule needs the complete TS (new and old data), thus, the items of TS that do not enter the original CS should be retained. In addition, CNN-rule requires that all training items are memory resident.

Although IB2 is based on CNN-rule, it is faster and incremental. Therefore, it can handle new training data or datasets that cannot fit into memory. Actually, IB2 is an one-pass version of CNN-rule (see Algorithm 2) and works as follows. When a new TS item x arrives (line 3), it is classified by the single nearest neighbour rule by scanning the contents of the current CS (line 4). If x is wrongly classified, it is placed in CS (lines 5-7). Then, x is discarded since it has been examined (line 8).

Similarly to CNN-rule, IB2 is non-parametric and its CS highly depends on the order items in TS. Contrary to CNN-rule, the CS built by IB2 does not ensure that it can correctly classify all examined training data.

IB2 is very fast because it is a one pass algorithm. Training data segments can update an existing CS in a simple manner and without considering the "old" (removed) data that had been used for CS construction. Each new TS item can be examined, be placed or not in CS, and then, removed. Additionally, IB2 can handle new class labels. All these properties render IB2 an appropriate DRT for dynamic/streaming environments where new training data may be gradually available or for training databases which cannot fit into the device's memory.

3. THE PROPOSED AIB2 ALGORITHM

The proposed AIB2 algorithm is a prototype abstraction version of IB2. Therfore, AIB2 is a non-parametric, fast, one-pass DRT that can handle new class labels. In addition, it is also appropriate for dynamic/streaming environments and can be applied on datasets that cannot fit into memory. Of course, like IB2, AIB2 does not take into account the phenomenon of concept drift [12] that may exist in data streams. IBL-DS [5] belongs to the IBL family of algorithms and can deal with this phenomenon.

The main difference between AIB2 and IB2 is the following. The TS items that are correctly classified by the 1-NN rule are not ignored. They contribute to the CS construction by updating their nearest prototype in CS. The main idea behind AIB2 is that prototypes should be at the center of the data area they represent. To achieve this, AIB2 adopts the concept of prototype weight. Each prototype has a weight value as an extra attribute that denotes the number of TS items it represents. The weight values are used for updating the prototype attributes in the multidimensional space.

AIB2 is presented in Algorithm 3. Initially, CS is populated by an item of TS whose weight is initialized to 1 (lines 2–3). For each TS item x, AIB2 searches in CS and retrieves its nearest prototype NN (line 5). If x is misclassified, it is placed in CS and its weight is initialized to one (lines 6–8). If x is correctly classified, NN's attributes are updated by taking into consideration its current weight and the attributes of x. Effectively, NN "moves" towards x in the multidimensional space (lines 10–12). Finally, the weight of NN is increased by 1 (line 13) and x is removed (line 15).

Algorithm 3 AIB2

```
Input: $TS$ Output: $CS$
 1: $CS \leftarrow \varnothing$
 2: pick an item $y$ of $TS$ and move it to $CS$
 3: $y_{weight} \leftarrow 1$
 4: for each $x \in TS$ do
 5:   $NN \leftarrow$ Nearest Neighbour of $x$ in $CS$
 6:   if $NN_{class} \neq x_{class}$ then
 7:     $x_{weight} \leftarrow 1$
 8:     $CS \leftarrow CS \cup x$
 9:   else
10:     for each attribute $attr(i)$ do
11:       $NN_{attr(i)} \leftarrow \frac{NN_{attr(i)} \times NN_{weight} + x_{attr(i)}}{NN_{weight} + 1}$
12:     end for
13:     $NN_{weight} \leftarrow NN_{weight} + 1$
14:   end if
15:   $TS \leftarrow TS - x$
16: end for
17: return $CS$
```

By updating the prototypes, AIB2 ensures that each prototype lies near the center of the data area it represents. The idea is that a CS built by AIB2 will contain better representatives compared to the CS built by IB2 and will achieve higher accuracy during classification. Furthermore, we expect that updating the prototypes will reduce the items that enter the CS (lines 6–8), and thus, AIB2 will achieve higher reduction rates and lower preprocessing cost compared to IB2.

Table 1: Dataset description

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

4. PERFORMANCE EVALUATION

4.1 Experimental setup

The three presented DRTs were evaluated using seven real life datasets distributed by the KEEL dataset repository[1] [4] and summarized in Table 1. All datasets were used without applying data normalization. Datasets MGT, LS and TXR are distributed sorted on the class label. This affects all examined DRTs because their performance depends on the order of data in TS. Hence, we randomized the items of these datasets. The three algorithms were implemented in C and the Euclidean distance was the distance metric used.

We did not include more DRTs in our experimental study because our purpose was to focus on incremental DRTs. Of course, CNN-rule is a non-incremental DRT, but it is considered a reference DRT and is being used in most research papers for comparison purposes. In addition, CNN-rule is the ancestor of IB2 and AIB2 and it makes sense to use it in our experiments.

The three DRTs were evaluated by estimating three measurements: (i) Accuracy (ACC) and the corresponding k value that was used, (ii) Reduction Rate (RR), and, (iii) Preprocessing Cost (PC) in terms of distance computations. Accuracy was estimated by executing the k-NN classifier over the CS built by each DRT. For each dataset and DRT, we report the average values of these measurements obtained via five-cross-fold validation. We used the five already constructed pairs of training and testing sets distributed by the KEEL repository. For every dataset and DRT, we report the k value that achieved the highest accuracy. In effect, we ran the five-cross-fold validation routine several times for different k values and selected the best one. Of course, all measurements were estimated after the arrival of all TS items. For k>2, two or more classes can be the most common. Such ties during nearest neighbour voting were resolved by assigning the new item to the class of the single nearest neighbour.

4.2 Comparisons

The results of our experiments are presented in Table 2. Each column lists the results related to a classifier. Best

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

measurements are in bold. PC measurements are in million distance computations. For reference, Table 2 presents the accuracy measurements obtained by the k-NN classifier on the original TS (Conventional k-NN classifier).

Although the three DRTs did not reach the accuracy levels of conv-k-NN, they were very close to them. With the exception of LR and TXR, CNN-rule achieved higher accuracies than IB2 and AIB2. However, it required much higher preprocessing cost and it achieved lower reduction rates than IB2 and AIB2.

As we expected, the proposed algorithm achieved better performance than IB2 in all datasets and in terms of all comparison criteria. In the LR and TXR datasets, AIB2 achieved higher accuracy than CNN-rule. Although, we expected even better performance, we believe that the improvements achieved by AIB2 are noteworthy.

Table 2: DTR Comparison in terms of Accuracy (ACC(%)), Reduction Rate (RR(%)) and Preprocessing Cost (PC(millions of computations))

Dataset		**Conv. k-NN**	**CNN**	**IB2**	**AIB2**
LR	ACC:	96.01	92.84	91.98	**94.12**
	k:	4	1	1	1
	RR:	-	83,54	85.66	**88.12**
	PC:	-	163.03	23.36	**20.10**
MGT	ACC:	81.32	**80.71**	79.84	80.36
	k:	14	18	38	42
	RR:	-	60.08	70.60	**71.90**
	PC:	-	281.49	34.61	**33.05**
PD	ACC:	99.37	**98.68**	98.04	98.33
	k:	4	1	1	1
	RR:	-	95.36	96.23	**97.19**
	PC:	-	11.75	1.78	**1.38**
LS	ACC:	91.22	**90.35**	89.37	89.42
	k:	4	6	6	1
	RR:	-	80.22	84.62	**86.72**
	PC:	-	17.99	2.22	**1.92**
SH	ACC:	99.82	**99.77**	99.74	99.72
	k:	1	1	1	1
	RR:	-	99.37	99.44	**99.46**
	PC:	-	45.30	8.26	**7.89**
TXR	ACC:	99.02	97.16	96.35	**97.69**
	k:	4	1	1	1
	RR:	-	91.90	93.33	**94.95**
	PC:	-	5.65	0.84	**0.66**
PH	ACC:	90.10	**87.83**	85.77	86.22
	k:	1	1	4	4
	RR:	-	76.04	80.85	**81.75**
	PC:	-	13.45	1.96	**1.84**
Avg	ACC:	93.84	**92.48**	91.58	92.26
	RR:	-	83,79	87.25	**88.58**
	PC:	-	76.95	10.43	**9.55**

5. CONCLUSIONS

Data reduction is an important issue in the context of k-NN classification. This paper proposed the AIB2 algorithm, an abstraction DRT that is based on the well-known IB2 algorithm. The main concept behind AIB2 is that each generated prototype should be at the center of the data area it represents. The experimental results illustrate that AIB2 can achieve higher reduction rates and classification accuracy and lower preprocessing cost than IB2.

6. REFERENCES

[1] C. Aggarwal. *Data Streams: Models and Algorithms.* Advances in Database Systems Series. Springer Science+Business Media, LLC, 2007.

[2] D. W. Aha. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *Int. J. Man-Mach. Stud.*, 36(2):267–287, Feb. 1992.

[3] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Mach. Learn.*, 6(1):37–66, Jan. 1991.

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

[5] J. Beringer and E. Hüllermeier. Efficient instance-based learning on data streams. *Intell. Data Anal.*, 11(6):627–650, Dec. 2007.

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

[7] S. Garcia, J. Derrac, J. Cano, and F. Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(3):417–435, Mar. 2012.

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

[9] L. Rokach. *Data Mining with Decision Trees: Theory and Applications.* Series in machine perception and artificial intelligence. World Scientific Publishing Company, Incorporated, 2007.

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

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

[12] A. Tsymbal. The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15, The University of Dublin, Trinity College, Department of Computer Science, Dublin, Ireland, 2004.