# Dynamic Learning of Cases from Data Streams

Fernando Orduña-Cabrera and Miquel Sànchez-Marrè

***Abstract*—This paper presents a dynamic adaptive framework for building a case library being able to cope with a data stream in the field of Case-Based Reasoning. The framework provides a three-layer architecture formed by a set of case libraries dynamically built. This Dynamic and Adaptive Case Library (DACL), can process in an incremental way a data stream, and can be used as a classification model or a regression model, depending on the predicted variable. In this paper, the work is focused on classification tasks. Each case library has a first layer formed by the dynamic clusters of cases, a second one formed by the meta-cases or prototypes of the cluster, and a third one formed by an incremental indexing structure. In our approach, some variant of *k*-d tress have been used, in addition to an exploration technique to get a more efficient retrieval time. This three-layer framework can be constructed in an incremental way. Several meta-case learning approaches are proposed, as well as some case learning strategies. The framework has been tested with several datasets. The experimental results show a very good performance in comparison with a batch learning scheme over the same data.**

***Index Terms*— Case-Based Reasoning, Data Streams, Dynamic Learning of Cases, Incremental Learning.**

## I. Introduction

In recent years, the problem of mining data streams has grown the attention of many researchers. Many real-world applications generate data continuously. For example, in network monitoring, telephone record calls, multimedia data, customer transactions, customer click streams, and so on. Advances in technology have facilitated new ways of continuously collecting data. In many applications, the volume of such data is so large that it may be impossible to store the data on disk. Furthermore, even when the data can be stored, the volume of the incoming data may be so large that it may be impossible to process any particular record more than once. Therefore, many data mining and database operations such as classification, regression, clustering, frequent pattern mining and indexing become significantly more challenging in this context [1]. The monitoring of many events in real time produces much information. In recent years, data stream mining field has grown rapidly. In [2] outlined some desirable properties for learning tasks in data streams: incrementality, constant time to process each example, single scan over the training set, and taking drift into account. Learning from data streams require incremental learning algorithms that take into account the problem of concept drift. The underlying concept or distribution of the data can change over time, and the mined models should be aware of the changes, and adapt themselves to the changes.

On the other hand, Case-Based Reasoning (CBR) systems solve new problems by retrieving and adapting the solutions to previously solved problems that have been stored in a case library [3][4][5]).

CBR is a very flexible reasoning paradigm, which can be used both as a classification technique and as a regression technique. The most common approach to predict a class label is the use of a simple CBR scheme (k-nearest neighbor classifier), but it can also be used to predict numerical variables, in a regression problem. This flexibility makes it a powerful tool for data mining. Furthermore, CBR integrates a learning step in its basic reasoning cycle. This learning activity makes CBR to be very suitable to be used for dynamic learning purposes. CBR systems become more competent over time, because they learn from experience. CBR approaches can process a data stream with a fine grained time window of length one. They can process tall the examples/cases one by one and adapt their model at each example.

In this paper, a dynamic adaptive framework is proposed to improve the CBR system performance coping especially with reducing the retrieval time, increasing the CBR system competence, and maintaining and adapting the case library to be efficient in size, especially in continuous domains (data streams) [6]. The framework proposed works for reasoning and learning both in supervised domains and unsupervised domains. One of the main contributions of the work is the proposal of a Dynamic Adaptive Case Library (DACL) framework. A DACL is composed of a set of dynamically built case libraries to cope with the heterogeneity and complexity of real domains. It learns cases and organizes them into dynamic cluster structures. The DACL is able to adapt itself to a dynamic environment, where new clusters, meta-cases or prototype of cases, and associated indexing structures (discriminant trees, *k*-d trees, etc.) can be formed, updated, or even removed. DACL offers a possible solution to the management of the large amount of data generated in an unsupervised continuous domain (data stream).

A very important aspect related to unsupervised continuous domains is the *incrementality problem*. General CBR systems assume that the set of cases available for building the case library is fixed and available at the beginning (batch learning). Then they build the memory indexing structures, like for instance, a *k*-d tree, decision/discriminant tree, etc. However,

This work was partially supported by the Spanish Thematic Network DAMA the Spanish Ministry MINECO, under Grant TIN2015-70308-REDT.

Fernando Orduña-Cabrera is with the Instituto Tecnológico Superior de Cajeme (ITESCA), Cajeme, Sonora, México (e-mail: forduna@itesca.edu.mx).

Miquel Sànchez-Marrè is with Universitat Politècnica de Catalunya-BarcelonaTech, Barcelona, Catalonia, Spain (e-mail: miquel@cs.upc.edu).

when a CBR system is facing an unsupervised continuous domain, the system should build and update the case library structure/s in an *incremental* way (incremental learning).

## II. RELATED WORK

In machine learning literature, several works have addressed the problem of learning from data streams [7] [8], and other works studied the time changing concept problem [9] [10][11][12]. Most common techniques used are temporal windows, which determines the training set for the learning algorithm, and the weighting of examples, which attempts to decrease the relevance of the older examples, and increase the relevance of the new ones. Also there are some mixture techniques. Some authors [13][14][15] propose the use of adaptive time windows in order to minimize the generalization error of the classification models.

Continuous problem domains (i.e., domains where cases are generated from a continuous data stream) require different underlying representations and place additional constraints on the problem solving process [16]. Ram and Santamaria define three characteristics where the problem domain is continuous, and those are: First, they require *continuous representations*,. Second, they require *continuous performance*. Third, these problem domains require *continuous adaptation* and learning. As the problems encountered become more varied and difficult, it becomes necessary to use fine-grained, detailed knowledge in an incremental manner to act, and to rely on continuous feedback from the environment to adapt actions and learn from experiences.

Reasoning about continuous domains is not an easy task. Moreover, this is a domain where CBR can rapidly extend its benefits because data is systematically collected for its analysis. A CBR system that continuously interacts with an environment must be able to create autonomously new situation cases (new concepts or clusters) based on its perception of the local environment in order to select the appropriate steps to achieve the current mission goal [17], but a general framework is still missing. Some systems that use case-based methods in continuous environment are described in [18][19][20].

There are two other central problems derived from the continuous nature of some domains. First of all, the *size of the case library* could grow very fast as the CBR system is learning new cases without an extensive improvement in the competence of the system, as pointed out in [21]. Two natural human cognitive tasks appear as the solution to these problems: forgetting ([22] and sustained relevant learning [23]. On the other hand, learning many cases could provoke an *overhead in the case library organization*. As new cases are stored in the case library, it will be necessary to update the case library organization [24].

## III. THE DYNAMIC ADAPTIVE CASE LIBRARY FRAMEWORK

When CBR systems are deployed in continuous domains, the case maintenance becomes critical. An uncontrolled growth of the case library can cause some serious problems of performance, where the retrieval efficiency and the quality of the retrieval is affected. When the case base has large amount of data, some inconsistent cases could be stored. This condition affects directly in the performance of the library.

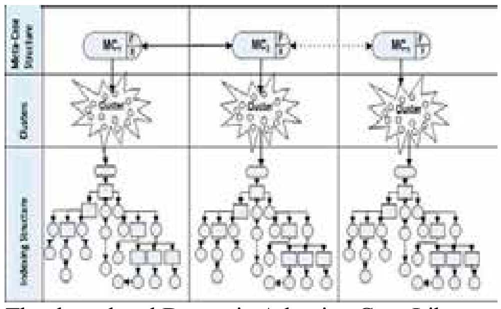

Fig. 1. The three-level Dynamic Adaptive Case Library

The architecture proposed presents a Dynamic Adaptive Case Library (see figure 1), with the aim of giving a possible solution to the management of the large amount of data generated in a continuous domain. The library will be dynamically built and formed of several sub-libraries. Each sub-library is organized hierarchically at three levels: *The Meta-case*: The Meta-case is the prototype of a concrete cluster of cases. *The clusters*: The set of cases belonging to the same cluster, and that are being represented by the meta-case. *The Indexing structures*: They implement the way that all the cases are organized in the sub-library. In our proposal, the cases are organized in a hierarchical indexing structures (k-d trees, discriminant trees, etc.), but could be organized with other indexing approaches.

Each time a new cluster of cases is created, a new sub-library is grown up. Next, the Meta-case structure and the retrieval and learning steps in a DACL will be detailed.

### A. The Meta-case

The idea of using a Meta-case as a representative case of several similar cases was introduced by Sànchez-Marrè in [25]. The aim is to show a formal proposal of how to a Meta-case ($Mc$) can be built. For our goal, a Meta-case is the prototype of a set of related cases. The centroid value is generated taking into account the whole cases stored in the indexing structure. With the following formula: $Mc_j^i = \frac{1}{n_i}\sum_{k=1}^{n_i} C_j^k$ *where* j = 1, …, m. The average distance (centroid) of the set of cases in that cluster is computed. A case ($C^i$) and a Meta-case ($Mc$) are described by $m$ attribute values. That is the first proposal that was introduced in [26]. Our proposal of constructing Meta-cases [27] is where the stochastic methods is introduced and prove it. DACL have as representative case a $Mc$, this $Mc$ works like a clustering filter where the decision to learn a new incoming case ($Mc$) is made, this decision concerns to a method to evaluate the $Mc's$ and find the most appropriate $Mc$ where to learn the $Nc$. Here follows the formalization of this process:

$$C^i = (C_1^i, C_2^i, \cdots, C_m^i)$$
$$Mc^i = (Mc_1^i, Mc_2^i, \cdots, Mc_m^i)$$

Where $n_i = \#Cases\ represented\ by\ the\ Meta-case(Mc^i)$

$$Mc_j^i = \frac{1}{n_i}\sum_{k=1}^{n_i} C_j^k \qquad j = 1, \dots, m \qquad \text{(1a)}$$

*If j is a qualitative attribute, and*

$$Mc_j^i = mode(C_j^k) \quad k = 1, \dots, n_i \quad j = 1, \dots, m \qquad \text{(1b)}$$

*If j is a quantitative attribute*

The Meta-case structure improves the performance of the retrieval time according to the proposals of Orduña and Sànchez-Marrè in [27][26]. The Meta-case is related to the clustering and learning processes.

## B. Retrieval Process in a DACL

Retrieving similar cases regarding to a new case is a process that needs to be done accurately. A new algorithm (DACL Retrieval algorithm) to retrieve the most similar case or cases in the DACL it is defined. First, the distance between the new case and all the Meta-cases must be computed. The most similar Meta-case will be selected, and its corresponding k-d tree, will be traversed for searching the most similar cases.

## C. Learning Process in a DACL

Retain task aims to maintain (Basic retaining/learning algorithm) a competent Case Library with a high coverage. The Retain process decides whether the new case needs to be stored in the case library, by updating an existing sub-library, building a new sub-library or simply ignoring the case. The process to make a decision is guided by the learning algorithm.
The basic retaining/learning algorithm (McSel-1) works as follows: it receives a solved new case (Nc) and then computes the distance to all the Meta-cases, with the aim to find the closest Meta-cases. Once the best Meta-case is found, it proceeds to compare whether the distance found falls within the α threshold (previously defined by the experts or tuned by trial and error experimentation). Then, it proceeds to store the new case into the current library (see figure 2). Otherwise, if the distance is higher than the α threshold value, a new sub-library must be created containing the new solved case. The last consideration in the algorithm is when the distance of the solved case falls within the ratio of two or more meta-cases. The solved case is randomly stored into one sub-library.

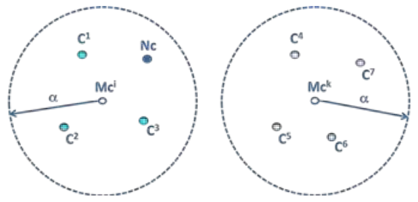

Fig 2. Basic Learning algorithm (McSel-1)

## D. Other Meta-case building strategies

In addition to the MCSel-1 strategy, we have proposed other approaches. Using a relaxed maximum radius, and avoiding impasses with real Meta-cases. These strategies are described below.

### 1) Using the Maximum Radius for building Meta-cases

The criterion uses the same procedures for building the prototypes using the idea of a radius around the Meta-case (McSel-1) but being a dynamic one regarding the farthest case of the prototype. The case at maximum distance of the Meta-case (RMax) is used with an extra relaxation condition (γ), to decide whether a new case should be stored in the library or not. This means that the criteria must successfully compute the $Mc^j$ where

$$\mathrm{j} = \arg min\{D(\mathrm{Nc}, Mc^j) \mid D(\mathrm{Nc}, Mc^j) \leq Mc^j.\mathrm{RMax} * (1+\gamma)\},\ \gamma \in [0,1]$$

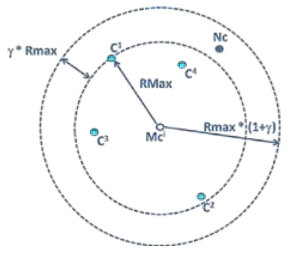

Fig 3. Learning with RMax and relaxation factor (McSel-4)

### 2) Learning Meta-cases avoiding impasses with real MC (McSel-5)

There is the possibility to work with real Meta-cases instead of the virtual Meta-cases. The difference relies in the fact that the prototypes of each cluster (virtual meta-cases) are replaced for *real cases*. This means that the prototype of a cluster of cases is a case existing within the set of cases of the cluster. Concretely, the real Meta-case will be the nearest real case to the virtual meta-case.
A variation of strategies McSel-1 and learning with real Meta-cases could be used as a good solution for *impasse situations*. Impasse situations happen when a new case which must be stored in the DACL is equally similar to more than one virtual meta-case (prototype). Even though a case could be at the same distance to several virtual Meta-cases, perhaps the distance to the corresponding real Meta-cases will not be the same, and the impasse situation could be solved. The strategy will be named as McSel-5. The strategy is depicted in figure 4.

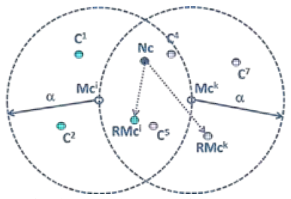

Fig. 4. Impasse resolution using real Meta-cases (MCSel-5)

## E. Incremental building of NIAR *k*-d Trees

The proposed NIAR *k*-d tree technique [28] has two main steps based on the computation of the average value of the corresponding attribute among the sub-tree cases, and selecting for that attribute, the value of the Nearest Instance/case to the Average as the Root (partition value). Several experimental results with some databases have shown that the retrieval in NIAR *k*-d tree is faster than in the standard *k*-d tree approach, and that using a Partial Matching Exploration (PME) technique to traverse the *k*-d tree, the accuracy in classification tasks is just a little bit lower than some exhaustive exploration techniques (see [28][29]). In the current work, the NIAR *k*-d tree structure, in addition to the PME technique has been used as the indexing strategy in the DACL framework.
A NIAR *k*-d tree can be created in an *incremental* way, processing each case at a time. Each new case arriving from the data stream, traverses the tree, and according to the comparison

with the corresponding partition values at each attribute node, follows the traversal until a non-full bucket is found (leave), or a full bucket is found (stored cases is equal to d). If the leave is not full, then the case is stored in that leave, updating several statistics associated with each attribute node (mean value of the attribute, standard deviation of the attribute, current new mean value, current new standard deviation, number of cases, and number of disturbance values) in all cases in the subtree. If the leave is full, then a split operations must take place. The leave node is converted into an attribute node, with the next attribute according to the specified criteria (random, etc.), and the corresponding statistics are initialized (mean value and deviation of the new attribute, etc.). Two new leaf nodes are created and the (d+1) cases are stored to the corresponding leaf.
When the system is continuously learning new cases, it can happen that the splitting value of a node, which should be the nearest value to the average value of the attribute, is not anymore the nearest value. This will be caused by the fact that the average value of the attributes is changing continuously. This situation could provoke that the indexing *k*-d tree could start to be not well balanced in all its subtrees, worsening the retrieval time. This will be a hard problem if the new values of the attribute, which are arriving at the DACL, are very disturbing.
For our proposal, we will consider that a value of an attribute,
$x_{n+1}$ is a dist. value ⇔ $|Av(x_n) - x_{n+1}| > \beta * stdev(x_n),\ \ \beta > 0$
$x_{n+1}$ is a nor. value ⇔ $|Av(x_n) - x_{n+1}| \leq \beta * stdev(x_n),\ \ \beta > 0$
That means the values with high dispersion will be those that are far from the mean value of the attribute. *Av*() is the mean value and *stdev*() is the standard deviation of the distribution of values. β is a modifying factor for the stdev() value. Initially, β is proposed to be set to 1 (β = 1), but other values can be used.

Fortunately, the DACL framework can easily and incrementally compute the new average values for all the attributes, according to the following formula:

$$Av(x_{n+1}) = \frac{n\,Av(x_n) + x_{n+1}}{n+1}$$

Where $Av(x_k)$ is the average mean value of the attribute x according to its first k values ($x_1$, ..., $x_k$).
Also the standard deviation can be computed in an incremental way through this formula due to Welford (Welford, 1962):
$stdev(x_{n+1}) = stdev(x_n) + (x_{n+1} - Av(x_n)) * (x_{n+1} - Av(x_{n+1}))$
One first strategy is that the DACL will rebuild a concrete NIAR *k*-d subtree, when the following condition would be met:
$x_{n+1}$ is a disturb. value ⇔ $|Av(x_n) - x_{n+1}| > \beta * stdev(x_n),\ \ \beta > 0$
This means that subtree rebuilding would be started each time the above condition is found, when a new case is going to be learnt into the DACL. This strategy is named as IncMTree-1.

Another strategy, which is a variation of the previous one, is that the DACL system will rebuild a concrete NIAR *k*-d subtree, at asynchronous time periods, when the following condition would be satisfied, since the last time the task was fired:

#Disturbance values ≥ δ * N

where δ is a specified percentage. We propose as initial trial that δ= 0.2, and N is the size of the corresponding sub-library.
This criterion means that when the number of disturbance values is higher than a specified percentage (for instance the 20%) of the number of cases of the sub-library, the task of rebuilding the indexing NIAR *k*-d tree corresponding to the sub-library will be started. A new NIAR *k*-d tree will be generated with the possible new splitting values at each node of the tree. This strategy is named as *Incremental Maintenance of trees triggered by a percentage of disturbance values* (IncMTree-2).

## *F. A strategy for learning relevant cases*

One of the important problems in unsupervised continuous domains is the *size* of the case library. As the continuous data stream is being processed incrementally, it must be used a concrete case learning strategy.
The standard strategy propose *learning all the cases* that has been processed and solved, with the hope to increase the competence of the case library, by increasing the coverage of the cases within the case library. This strategy will be named as *All Case Learning* (CaseL-1). Of course, this strategy, even though could increase the competence of the CBR system, could enlarge too much the size of the case library.
In order not to increase the size of the Case library in an unnecessarily way, a possible new strategy could be thought to learn only the most relevant cases. A *relevant case* would be a case that increases the coverage of the case library (i.e, the competence of the CBR system). The coverage of the case library is increased when the new case learnt could solve different cases to the cases that could be solved previously without this new case. In our DACL approach, for each corresponding Meta-case, we have a NIAR k-d tree structure.

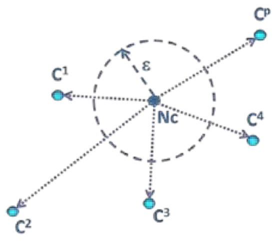

Fig. 3. Relevant case learning (CaseL-2)

Therefore, it means that when a new case must be learnt, a discriminating process has been done traversing the tree until the appropriate leave of the tree has been located. Our proposal is stated as follows:

A new case ($N_c$) will be learnt ⇔ $\min D(N_c, C^j) \geq \varepsilon$

where ε∈[0,1] and $C^j$ ($j = 1, \dots, p_j$) are the $p_j$ cases stored in the leave $j$ of the tree, where $N_c$ should be stored. An initial value of ε = 0.2 is proposed.
The rationale under the proposal is that a new case would be relevant whether it is enough different to all the other cases stored in the corresponding leave. Thus, it must be, at least, at a distance ε to the most similar case in the leave.
This strategy is named as *Relevant Case Learning* (CaseL-2), and it is illustrated in figure 3.

## IV. Experimental work

The experimental work comprised two main scenarios. First, testing several databases from the UCI repository (Abalone, Balance, Car evaluation, Ecoli, Glass, Ionosphere, Iris, Pima and Waveform) in a *batch learning mode* using the DACL framework, but in a supervised way. That means that the number of class labels was known, and the number of case libraries of the DACL was fixed a priori. Thus, DACL was used not in a dynamical mode but in a static multi-library style. In the second scenario, the same databases, used as they were unsupervised datasets, were tested in an *incremental learning mode*, with the whole Dynamic Adaptive Case Library framework, (with meta-case learning strategies and case learning approaches, the NIAR *k*-d trees, and the PME technique). Therefore, this way, the hierarchical structures (NIAR *k*-d trees) were incrementally constructed, and all the cases in the databases used were processed in a step by step mode.

The experimental setting was done under the following characteristics:

• The above nine databases were tested

• Different strategies were tested. These combinations are the result of the crossing of 3 Meta-case Selection strategies (Basic learning algorithm with radius α [MCSel-1], Avoiding impasses with real Mc [MCSel-5] and Learning using maximum radius RMax with relaxation [MCSel-4]), the incremental maintenance strategy for the NIAR k-d Tree (IncMaintTree [IncMtree-1] and two strategies for the Learning of cases (AllCaseLearning [CaseL-1] and RelCaseLearning [CaseL-2]).

• For each database and for each strategy, 10 execution runs were done sampling randomly the cases to get different ordering of the cases. In addition, one more run was done with the original ordering of each database.

• For each execution and for each database several statistics were computed: the average accuracy (in percentage); the average time retrieval (time in μs), the detection of new prototypes.

### A. Experimental results

In this subsection, the results obtained regarding the following parameters are detailed: detection of meta-cases, evolution of the detection of Meta-cases, average accuracy, and average time retrieval obtained with the Iris and Balance datasets.

#### 1) Detection of Meta-cases

Regarding the discovering of Meta-cases and prototypes, all the databases have been processed and the different meta-cases (prototypes) were found to correspond accurately with the real hidden class labels.

In some of the databases, the exact number of prototypes corresponding with the same number of classes in the original database was found. For instance, it is the case of Pima and Ionosphere (2 classes and 2 prototypes found), Iris and Balance (3 classes and 3 prototypes found), Car (4 classes and 4 prototypes) and Ecoli (8 classes and 8 prototypes). In the other databases, the exact number of classes was not discovered, but this that not means that the predictive performance of the system were worse. On the contrary, on same execution runs it will be observed that the predictive power of a different number of prototypes is higher than with executions having the same number of prototypes.

#### 2) Performance evaluation

The average precisions values found are reported in the following table for Iris and Balance datasets (table I).

It is important to mention that the algorithm's learning process updates its learning base each time a new case and/or meta-case appears, which has as a result that the initial precision value is lower and grows as more meta-cases are identified and more cases are learnt.

Table I. Mean precision values for Iris and Balance databases

| | #CL | #MCs | #NCL | Accuracy | $RMax\gamma$ |
|---|---|---|---|---|---|
| Iris | #CL1=50 | 3 | 3 | 87% | 0.425 |
| | #CL2=68 | | | | |
| | #CL3=32 | | | | |
| Balance | #CL1=298 | 3 | 3 | 68% | 5.9 |
| | #CL2=304 | | | | |
| | #CL3=23 | | | | |

Notwithstanding, if we compare the accuracy results of the non-incremental version of our DACL approach and the incremental version, we can observe that the incremental versions only have an accuracy decrease of 5-7%. See table II.

Table II. Comparison of accuracy in non-incremental DACL versus incremental DACL approaches

| | **Non-incremental DACL Accuracy (%)** | **Incremental DACL Accuracy (%)** |
|---|---|---|
| Iris | 91.8 | 87 |
| Balance | 74.9 | 68 |

Some results obtained from the tests done with the incremental strategies are detailed for the first six strategies for the Iris dataset (see table III). Other results from other datasets are not described due to lack of space.

Table III. Iris database performance with the 6 strategies

| | | **Iris Data Base Evaluation** | | | | | |
|---|---|---|---|---|---|---|---|
| Strategy | Mc's | Incremental k-d Tree | | Strategy | Mc's | Incremental k-d Tree | |
| | | **Avg Time** | **%Succ** | | | **Avg Time** | **%Succ** |
| S1: MCSel-1 IncMTree-1 CaseL-1 | 3 | 101.5 | 75 | S4: MCSel-1 IncMTree-1 CaseL-2 | 3 | 106.45 | 85 |
| | 2 | 129.4 | 90 | | 2 | 105.6 | 100 |
| | 3 | 113.95 | 90 | | 3 | 128.05 | 85 |
| | 3 | 119 | 90 | | 2 | 103.4 | 85 |
| | 2 | 99.85 | 80 | | 2 | 104.55 | 95 |
| | 3 | 104 | 80 | | 3 | 81.95 | 80 |
| S2: MCSel-5 IncMTree-1 CaseL-1 | 3 | 30.85 | 90 | S5: MCSel-5 IncMTree-1 CaseL-2 | 3 | 13.15 | 75 |
| | 3 | 15.15 | 90 | | 2 | 27.6 | 90 |
| | 3 | 15.8 | 90 | | 2 | 22.15 | 95 |
| | 3 | 27.6 | 85 | | 3 | 24.15 | 85 |
| | 2 | 29.6 | 85 | | 2 | 23.85 | 90 |
| | 2 | 28.1 | 90 | | 2 | 29.2 | 100 |
| S3: MCSel-4 IncMTree-1 CaseL-1 | 8 | 19.65 | 45 | S6: MCSel-4 IncMTree-1 CaseL-2 | 8 | 5.05 | 60 |
| | 6 | 43.65 | 50 | | 10 | 3.7 | 75 |
| | 8 | 58.55 | 65 | | 7 | 5.5 | 70 |
| | 2 | 105.25 | 90 | | 6 | 7.95 | 70 |
| | 2 | 36.45 | 75 | | 13 | 36.55 | 75 |
| | 12 | 24.15 | 80 | | 4 | 21.8 | 70 |

The results in the nine databases tested showed a good time performance and accuracy average on several execution runs, even though the case library was incrementally populated. They achieved good accuracy values, just a little bit worse than

compared with the non-incremental processing of the cases. From the experimentation done we can conclude that the DACL framework is a promising approach to cope incrementally with data streams.
This approach has been able to discover the Meta-cases (prototypes) in a very good way, and the number of prototypes is sometimes equal to the existing number of the "hidden" class labels in the databases. Thus, in this sense, the DACL incremental processing and learning of cases, can be considered as an incremental clustering technique. Notwithstanding, in the scenarios when the discovered number of prototypes is different from the number of existing ones, the accuracy of the CBR system is very good, and even better than when discovering the same number of prototypes.
Moreover, the average final precision is just a little bit worse than the average precision obtained with the same DACL approach, but just using the usual non-incremental way. In such non-incremental scenarios, the Case Library is seeded up with some training set of cases, and tested with a testing set, and of course, it should have better accuracy results, because at the beginning, this configuration has many more cases in the Case Library than the corresponding incremental configuration. These experiments provided the confirmation that the proposed DACL framework is especially suitable to cope with incremental data streams. As showed later the DACL approach has been able to detect and construct the same number of meta-cases (prototypes) than the existing ones in the databases.

## V. Conclusions

It is not easy to cope with data streams, where in an incremental way, a lot of cases are being generated, and must be processed by the CBR system. This way, the system has not so much cases in its Case Library like in a non-incremental processing scenario. This means that it is pretty more difficult to achieve good accuracy percentages, because at the beginning of the processing, normally the precision will be lower than when working in a non-incremental scenario.
The experimentation done has outlined that the DACL approach is able to satisfactorily cope with unsupervised and incremental scenarios.
The entire Dynamic Adaptive case Library framework has been tested with simulated unsupervised domains where the cases were incrementally processed, and all the DACL structures were incrementally created and updated, with good results relating to the automatic discovery of prototypes (meta-cases), which correspond to the actual existing prototypes. In addition, good retrieval time measures and accuracy measures were obtained. Finally, and mainly, the proposed DACL framework is able to cope, with good performance, with the incremental processing of data streams, than most of the techniques used in CBR cannot.

## References

[1] C. Aggarwal. Data Streams: Models and Algorithms. Springer, 2007
[2] G. Hulten and P. Domingos. Catching up with the data: research issues in mining data streams. *In Proc. of Workshop on Research issues in Data Mining and Knowledge Discovery, 2001.*
[3] M.M. Richter and R.O. Weber. "Case-Based Reasoning: a text-book". Springer-Verlag, 2013
[4] B. López. "Case-Based Reasoning: a Concise Introduction". Morgan & Claypool, 2013.
[5] López de Mántaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, and M. T. Cox, K. Forbus, M. Keane, A. Aamodt and I. Watson. "Retrieval, reuse, revision and retention in case-based reasoning". *The Knowledge Engineering Revie*w 20(3), 215-244, 2005.
[6] Babcock B., Babu S., Datar M., Motwani R., and Widom J. Models and issues in data stream systems. In P. Kolaitis, editor, Proceedings of the 21nd Symposium on Principles of Database Systems, pages 1–16. ACM Press, 2002.
[7] João Gama and Mohamed Medhat Gaber (Eds.). "Learning from Data Streams: Processing Techniques in Sensor Networks", Springer, 2007.
[8] João Gama. "Knowledge Discovery from Data Streams". Chapman and Hall/CRC, 2010.
[9] G. Hulten, L. Spencer, and P. Domingos. "Mining time-changing data streams". In F. Provost, editor, Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining. ACM Press, 2001
[10] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In P. Langley, editor, Proceedings of ICML-00, pages 487–494. Morgan Kaufmann Publishers, San Francisco, US, 2000.
[11] M. Kubat and G. Widmer. Adapting to drift in continuous domain. In Proceedings of the 8th European Conference on Machine Learning, pages 307–310. Spinger Verlag, 1995
[12] M. Maloof and R. Michalski. "Selecting examples for partial memory learning". *Machine Learning*, 41:27–52, 2000
[13] R. Klinkenberg. "Learning drifting concepts: Example selection vs. example weighting". *Intelligent Data Analysis*, 8(3):281 – 300, 2004.
[14] R. Klinkenberg and I. Renz. "Adaptive information filtering: Learning in the presence of concept drifts". In Learning for Text Categorization, pages 33–40. AAAI Press., 1998
[15] G. Widmer and M. Kubat. "Learning in the presence of concept drift and hidden contexts". Machine Learning, 23:69–101, 1996.
[16] Ram A. and J. C. Santamaría (1997). "Continuous Case-Based Reasoning. Artificial Intelligence 90, pp. 86-93.
[17] Haris S. and R. Slobodan (2005). Autonomous Creation of New Situation Cases in Structured Continuous Domains. Springer-Verlag, pp. 537—551
[18] Urdiales C., E.J. Pérez, J. Vázquez-Salceda, M. Sànchez-Marrè and F. Sandoval (2006). "A Purely Reactive Navigation Scheme for Dynamic Environments using Case-Based Reasoning". *Autonomous Robots* 21, pp. 65-78.
[19] Kruusmaa M. (2003). Global Navigation in Dynamic Environments Using Case-Based Reasoning. Autonomous Robots 14, pp. 71-91.
[20] Ram A., R. C.Arkin, K. Moorman and R. J. Clark (1997). Case-based reactive navigation: a method for on-line selection andadaptation of reactive robotic control parameters. Systems, Man, and Cybernetics Part B 3, pp. 376-394.
[21] Miyashita K. and K. Sycara. Improving system performance in case-based iterative optimization through knowledge filtering. Procc. of IJCAI 1995, Morgan Kaufmann, pp. 371—376. 1995.
[22] Keane M. T. and B. Smyth (1995). "Remembering to Forget: A Competence-Preserving Case Deletion Policy for Case-based Reasoning systems". Procc. of IJCAI 1995, Morgan Kaufmann, 377-382.
[23] Sànchez-Marrè M., U. Cortés, I. Rodríguez-Roda, and M. Poch. "Sustainable case learning for continuous domains". *Environmental Modelling and Software* 14(5):349-357, 1999.
[24] Meléndez J, J. Colomer and J. Ll. de la Rosa. "Expert Supervision Based on Cases". Proc. of 8th IEEE International Conference on Emerging Technologies and Factory Automation, Vol. 1, pp. 431—440, 2001.
[25] Sànchez-Marrè M, U. Cortés, I. Rodríguez-Roda and M. Poch. "Using Meta-cases to Improve Accuracy in Hierarchical Case Retrieval". Computación y Sistemas (4), pp. 53, 2000.
[26] F. Orduña Cabrera, F. and M. Sànchez-Marrè, M (2009). Dynamic Adaptive Case Library for Continuous Domains. In Proc. of 12th CCIA'2009. Frontiers in Artificial Intelligence and Applications Series, Vol. 202, pp. 157-166.
[27] F. Orduña Cabrera, and M. Sànchez-Marrè. "Embedding k-d Trees and Exploration Techniques within a Multiple Case Library to Improve Case Retrieval". Submitted work, 2015.
[28] F. Orduña Cabrera, and M. Sànchez-Marrè. "Using NIAR k-d Trees to Improve the Case-Based Reasoning Retrieval Step". In Proc. of MICAI 2013). LNAI, vol. 8266, pp. 314-325, 2013.
[29] F. Orduña Cabrera, 2015. "A Dynamic Adaptive Framework for improving Case-Based Reasoning System Performance". Ph.D Dissertation. Dept. of Comp Science, Univ. Politècnica de Catalunya.