Information Sciences 334–335 (2016) 250–272

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Evolving association streams

Andreu Sancho-Asensio[a,*], Albert Orriols-Puig[a], Jorge Casillas[b]

[a] Research Group in Electronic and Telecommunications Systems and Data Analysis, La Salle - Ramon Llull University, Quatre Camins 2, Barcelona 08022, Spain

[b] Department of Computer Science and Artificial Intelligence, University of Granada and the Research Center on Information and Communications Technology (CITIC-UGR), Granada E-18071, Spain

ARTICLE INFO

Article history:
Received 1 October 2014
Revised 10 November 2015
Accepted 19 November 2015
Available online 10 December 2015

Keywords:
Online learning
Data stream
Association rule
Concept drift
Genetic fuzzy systems

ABSTRACT

The increasing bulk of data generation in industrial and scientific applications has fostered practitioners' interest in mining large amounts of unlabeled data in the form of continuous, high speed, and time-changing streams of information. An appealing field is association stream mining, which models dynamically complex domains via rules without assuming any a priori structure. Different from the related frequent pattern mining field, its goal is to extract interesting associations among the forming features of such data, adapting these to the ever-changing dynamics of the environment in a pure online fashion-without the typical offline rule generation. These rules are adequate for extracting valuable insight which helps in decision making. This paper details Fuzzy-CSar, an online genetic fuzzy system designed to extract interesting rules from streams of samples. It evolves its internal model online, being able to quickly adapt its knowledge in the presence of drifting concepts. The different complexities of association stream mining are presented in a set of novel synthetic benchmark problems. Thus, the behavior of the online learning architecture presented is carefully analyzed under these conditions. Furthermore, the analysis is extended to real-world problems with static concepts, showing its competitiveness. Experiments support the advantages of applying Fuzzy-CSar to extract knowledge from large volumes of information.

1. Introduction

With the invention of digital computers, technology has allowed the industry to collect and store massive amounts of data concerning business processes, allowing for subsequent analysis and exploitation. Since then, the analysis and exploitation of large databases is a recurrent topic and there have been several contributions across a wide set of disciplines [5]. The need for extracting useful information from massive amounts of data, usually as a continuous, high speed and time-changing data stream has risen dramatically in industrial and scientific applications [5]. Stock markets and smart networks, among many others, are the primary targets of this kind of knowledge extraction [15,33]. However, traditional algorithms were not designed for handling data which is delivered in the form of streams and they are not able to extract accurate models in these environments [34], thus requiring online techniques. The challenges hampering the learning process under data streams are manifold: (1) obtaining a fast reaction time for changes in concept, (2) data can only be handled once, (3) storage limitations, and (4) varying noise levels.

* Corresponding author. Tel.: +34 677281045.
E-mail addresses: andreu.sancho@gmail.com, andreus@salleurl.edu (A. Sancho-Asensio), aorriols@salleurl.edu (A. Orriols-Puig), casillas@decsai.ugr.es (J. Casillas).

http://dx.doi.org/10.1016/j.ins.2015.11.043
0020-0255/

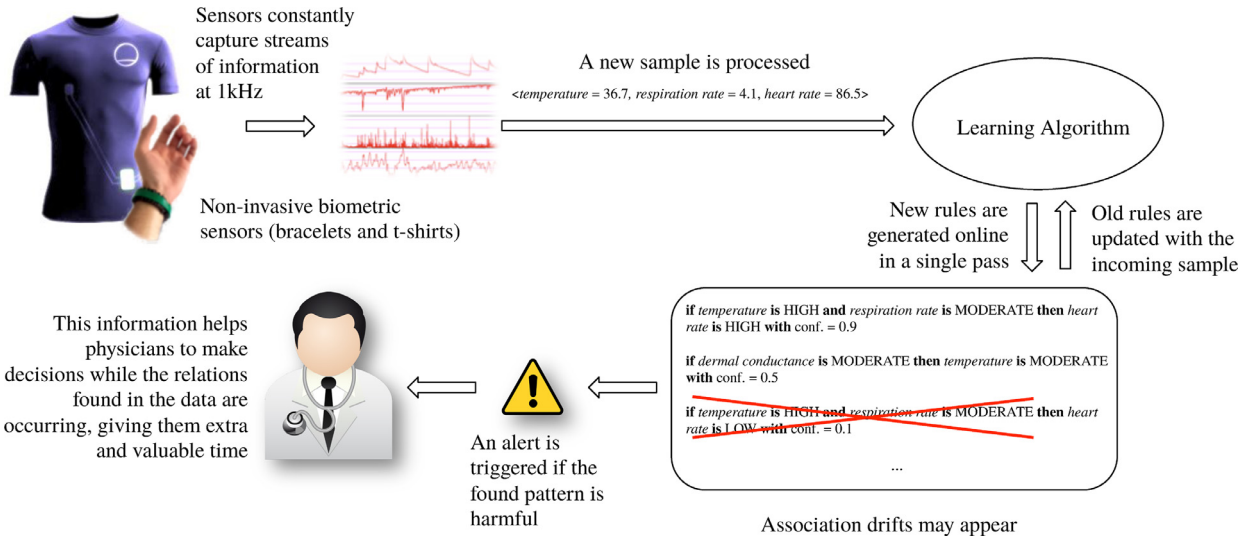

Fig. 1. An application of association stream mining. In this scenario association streams help physicians and doctors to make decisions, giving them extra and valuable time by modeling the scenario dynamically.

Despite the need for addressing the complexities of mining new, potentially useful information from data streams, current online mining field research focuses mainly on supervised methods, which assume an a priori relational structure for the set of features that define the problem. This issue is often distant from real-world situations, and it is emphasized when changes in concept occur: the undefined and ill-structured nature of the situation jointly with the dearth of labelled examples—or complete absence of them—from this new concept makes the application of a *pure* supervised algorithm ill-suited. Hence, solving the problems requires the use of hybrid methods which combine supervised with unsupervised techniques in order to deal with the problem [41].

A real case that entails a sound example of this issue is in detecting potential threats to web sites and network infrastructures. In this scenario there are a set of features that indicate suspicious acts on the infrastructure (i.e., strange characters in login interfaces or strange traffic flows, among others) by malicious users trying to identify the vulnerabilities of the system to take possession of it. It is worth mentioning that other anomaly detection strategies exist, such as statistical or based on data density. However these are typically based on labeled data and, therefore, they do not adapt themselves to concept changes. In our particular case we are not interested in directly detecting—and thus informing of the attack—but in continuously monitoring the system by adapting to the new trends that data are showing, which helps experts to decide if the system is under attack. Fig. 1 depicts a possible use-case of association stream, where multiple non-invasive biometric sensors (bracelets and t-shirts) that capture continuous physiological data in the form of streams are given to patients. In this environment, the goal is to help physicians and doctors to monitor and track the status of their patients in an online fashion.

In this regard, the unsupervised approach becomes a feasible alternative to the aforesaid issues. More specifically, *association stream mining*, focused on extracting associations among variables via production rules online and in a single pass, is specially attractive from the point of view of practitioners due to (1) the demand of interpretability of the patterns discovered in data—human-readable rules that provide valuable insight (e.g., *every time X occurs Y also happen*)—, (2) the need for discovering patterns while they are happening (e.g., the new steps of the web attack) and hence adapt to them, and (3) the high and continuous volumes of data to be processed, which require scalable learners.

Despite the importance of facing association streams, it is a new area and consequently, to the best of our knowledge, so far no fully operational approaches tackling the challenges discussed in this paper have been presented. The most similar algorithms for knowledge extraction under these conditions focus mainly on the identification of frequent variables—referred to as *online frequent pattern mining*—, relegating the generation of rules in a second place in an offline process [5], which make these mostly unpractical for handling association stream premises, where the rules discovered have to be present immediately and adapt to the changing dynamics of the continuous flow of data. Several investigations have been presented so far on frequent pattern mining, but the majority of these ignore the presence of concept drifts. Moreover, most of those algorithms are only able to deal with problems described by categorical features [16,22]. Even so, a few quantitative algorithms for frequent pattern mining have been proposed, most of them based on fuzzy logic [12,30], focusing on the identification of frequent variables and not on the final rules. Likewise, these systems are not designed for the harsh conditions of continuous flow and time-changing concepts that association stream stresses.

Association stream mining research area is rooted in traditional, offline association rule mining, one of the most well known Data Mining fields. It is used to extract interesting information from large data bases by means of describing the properties of data in the form of production rules, e.g., $X \Rightarrow Y$, where both X and Y are sets of features and $X \cap Y = \emptyset$ [1]. Association rule

mining algorithms do not assume any a priori structure for the problem. Thus, these methods automatically uncover connections between features. Two main types of representations are found in the literature: qualitative and quantitative association rules. Qualitative association rules are only able to handle problems defined by binary sets of items. In order to deal with continuous features, practitioners switched to quantitative association rules, by exploring two different strategies: (1) discretize the features and then deal with them in a purely qualitative fashion [44], which may lead to poor results due to the loss of information, and (2) using an interval-based representation [32]. Later on, fuzzy modeling was introduced in order to create highly legible models from domains with uncertainty and imprecision and to avoid unnatural boundaries produced by the interval-based representation [20,26].

The purpose of this paper is to follow up the work started in [36,38] by addressing the challenges of association stream extending the analysis of the *Fuzzy-Classifier System for Association Rules* (Fuzzy-CSar), a *Michigan-style learning classifier system* (LCS) that makes use of *genetic algorithms* (GAs) to evolve *independent-fuzzy-rule* sets online from streams of data. Inheriting the learning architecture of XCS [45] and UCS [37], Fuzzy-CSar is a general purpose unsupervised learning algorithm specialized in the extraction of highly interesting fuzzy association rules in a single step, that is, Fuzzy-CSar does not generate lists of frequent itemsets but it directly evolves the set of association rules, which results in an effective way to tackle association streams.

The behavior of Fuzzy-CSar under a set of online benchmark problems is first studied. Each of these problems faces a different aspect of the aforementioned challenges in the field of association stream mining (that is, a continuous stream of data, high speed, noisy, and time-changing patterns). Thereafter, the analysis is extended by comparing algorithm scalability and quality of results of Fuzzy-CSar to the ones for Fuzzy-Apriori [26], one of the most well known fuzzy association rule mining algorithms, in a set of real-world problems with static concepts. Finally, the experimentation is concluded in a synthetic environment comparing Fuzzy-CSar against FP-Growth [23] in a large data set with static concepts. These experiments will show how competitive Fuzzy-CSar is in both online and offline environments.

The contribution of this paper is twofold: (1) an introduction to association streams is provided: a new field inside data streams strongly related to online frequent pattern mining. In this regard and due to the dearth of benchmark problems, an original framework for algorithm evaluation is proposed, and (2) An algorithm for mining association stream is detailed based on the Michigan-style LCS framework, an online and mature architecture that has demonstrated to be successfully applied to data streams [39].

The remainder of this paper is organized as follows: Section 2 provides the basic concepts of association rule mining (ARM), details the ways of inferring rules from data and describes the difficulties of learning from online environments. Section 3 gives a detailed description of the proposed Fuzzy-CSar algorithm. Sections 4–6 provide the results of the experiments done with Fuzzy-CSar. Section 7 shows a SWOT analysis of Fuzzy-CSar. Finally, Section 8 summarizes and concludes the paper.

2. Framework

2.1. Association rules in a nutshell

ARM is a well-studied field which is aimed at extracting associations between variables in the form of production rules and it is formally defined as follows [1]: Let $I = \{i_1, i_2, \ldots, i_\ell\}$ be a set of ℓ binary features, typically called *items*. Let $T = \{tr_1, tr_2, \ldots, tr_N\}$ be a set of N transactions, where each transaction tr_j contains a binary vector of features indicating in each position if a particular item is present or not. X is an *itemset* if $X \subset I$. An itemset X has associated a measure of its importance in T: its support, which is computed as $sup(X) = |X(T)|/|T|$, where $X(T)$ is the set of variables in the antecedent part of the rule. An itemset is said to be a *frequent itemset* if its support is greater than a user-defined threshold, typically addressed as *minsup* in the literature. Then, an *association rule* R is an implication of the form $X \to Y$, where both X and Y are frequent itemsets and $X \cap Y = \emptyset$. Traditionally, association rules are assessed by two qualitative measures: their support and their confidence:

$$sup(X \to Y) = \frac{sup(X \cup Y)}{|T|}, \quad con(X \to Y) = \frac{sup(X \cup Y)}{sup(X)}. \tag{1}$$

Therefore, support indicates the frequency of occurring patterns and confidence evaluates the strength of the implication denoted in the association rule. Other quality measures can be used instead of the confidence to represent the reliability of the rule, being lift and accuracy the most popular ones [32]. The classic process of ARM consists of two steps: (1) finding all the frequent itemsets in T and (2) extracting rules from these frequent itemsets.

The ultimate goal of knowledge discovery and data mining is to find useful patterns from data. However, real-world problems are *flooded* with imprecision and uncertainty that hamper the learning of traditional machine learning algorithms. The use of fuzzy logic in ARM allows the creation of highly legible models from both qualitative and quantitative data. Let $I = \{i_1, i_2, \ldots i_\ell\}$ be a set of ℓ features, and let $A \subset I$ and let $C \subset I$, $A \cap C = \emptyset$. A fuzzy association rule is an implication of the form $X \to Y$ where:

$$X = \bigwedge_{i_i \in A} \mu_{\widetilde{A}}(i_i) \quad \text{and} \quad Y = \bigwedge_{i_j \in C} \mu_{\widetilde{C}}(i_j), \tag{2}$$

with $\mu_{\widetilde{A}}(i_i)$ being the membership degree of features in the antecedent part of the rule and $\mu_{\widetilde{C}}(i_j)$ the membership degree of features in the consequent part of the rule. In this regard, support is typically extended by using the product T-norm and

confidence is extended by using the *Dienes implication* [20]:

$$sup(X \to Y) = \frac{1}{|T|} \sum \mu_{\widetilde{A}}(X) \cdot \mu_{\widetilde{C}}(Y), \quad con(X \to Y) = \frac{\sum(\mu_{\widetilde{A}}(X) \cdot \max\{1 - \mu_{\widetilde{A}}(X), \mu_{\widetilde{C}}(Y)\})}{\sum \mu_{\widetilde{A}}(X)}. \tag{3}$$

where $\mu_{\widetilde{A}}(X)$ is the membership degree of the antecedent part of the rule and $\mu_{\widetilde{C}}(Y)$ is the membership degree of the consequent part of the rule.

2.2. Obtaining rules from data

There are mainly three different strategies for obtaining association rules: (1) by means of candidate generate-and-test procedures, (2) by means of divide-and-conquer methodologies, and (3) by means of *metaheuristics*. In what follows, these strategies are briefly detailed.

The candidate generate-and-test family of procedures exploits the *downward closure* property which states that if an itemset is frequent, then all of its subsets are also frequent itemsets. These kinds of algorithms perform multiple scans on the database in order to obtain the frequent itemset candidates. Most often, there is a high number of candidates, so support counting for candidates can be time expensive. After the frequent itemsets are found, the algorithm combines these to produce the desired rules.

In order to scale up the generation of frequent itemsets and to avoid unnecessary candidate generation, divide-and-conquer procedures build an auxiliary structure to get statistics of the itemsets, thus avoiding multiple scans of the data. This auxiliary structure is typically based on a k-ary tree [2]. After the frequent itemsets are found, the algorithm combines these to produce the desired rules.

Another way of avoiding the breeding of candidate generation is by making use of metaheuristics. One of the most attractive families of such techniques are Evolutionary Algorithms (EAs). EAs can evolve the desired association rules directly from data without the two-step process of classic association rule mining algorithms. This is the case for LCSs, which are rule-based evolutionary learning systems. LCSs are classified into two main branches: on the one hand *Michigan-style LCSs* are online cognitive-inspired systems that combine a credit apportionment algorithm with EAs, where each individual is a single production rule, whose quality is evaluated online by the cognitive system. An EA is applied periodically to the population to discover new rules. On the other hand *Pittsburgh-style LCSs* are offline EAs, where every individual is a set of production rules that represent the complete solution for the given problem, thus all the solutions compete in the population. The genetic search is usually driven by a generational GA.

More recently, other EA strategies inspired by traditional LCSs have flourished: the Iterative Rule Learning (IRL) approach which has been broadly used in association rule mining [32]. Similarly to Michigan-style LCSs, in IRL each individual is a single production rule, and similarly to Pittsburgh-style LCSs, the genetic search is driven by a generational GA. IRL iteratively performs two steps: (1) learn a rule that covers part of the training examples via the genetic discovery mechanism and (2) remove the covered examples from the training set. Another popular strategy for mining association rules is the Genetic Cooperative-Competitive Learning (GCCL) branch. This approach combines the offline rule processing of Pittsburgh-style LCSs with the idea of Michigan-style LCSs that the solution is the whole population, and so, rules need to collaborate to cover all the input space. Recently, the GCCL approach has been successfully applied to mine association rules that model consumers' behavior [11].

2.3. Data stream mining

Learning from online streams of data presents a series of difficulties due to the continuous, potentially unbounded in size, high speed, noisy, and time-changing nature of data streams. Classical machine learning methods are not able to extract accurate models when the information comes in the form of a data stream, and these have to be adapted. Typically, the use of time windows to store parts of the stream is the most common way to adapt an offline algorithm. However, managing time windows has some inherent decision challenges that have to be addressed. For example, a key aspect lies in deciding the size of the time window because it controls the ability of forgetting past samples that are no longer useful [34]. Also, deciding which samples have to be forgotten is not a trivial task. On the other hand, incremental, online learners take advantage by handling the stream directly. Another major challenge of data streams lies in the detection of changes in the target concepts (*concept drifts*), which can be abrupt, incremental or recurring [34]. All the characteristics of data streams are included in the *massive online analysis* (MOA) framework [8], which supplies a large amount of algorithms for stream classification and clustering.

Recent contributions tackle the issues of data streams by performing some incremental statistical tests over the incoming stream. Due to the noisy characteristics of data streams, the use of fuzzy logic is broadly extended [9]. In this regard, there are several recent contributions in both supervised [39] and unsupervised [12] learning paradigms.

2.4. Association stream mining in a nutshell

Association stream mining is a novel field closely related to both data streams and association rule mining that aims at extracting associations from variables in the form of rules. The main characteristics of this field are: (1) continuous flow of information that has to be processed in a single pass, (2) there are restrictions in memory usage, (3) the concept to be learnt may change over time, and (4) it does not assume an underlying structure nor distribution in incoming fluxes of data.

Table 1
Characteristics of frequent pattern mining algorithms for data streams.

Reference	Category	Type	Data	Approach	Rules	Experiments
[29]	Heavy hitter	Counting	Categorical	–	No	–
[13]	Frequent pattern mining	Counting	Categorical	Decay factor	No	Synthetic
[14]	Frequent pattern mining	Counting	Categorical	Sliding	No	Synthetic
[46]	Top-k frequent itemsets	Counting	Categorical	Sliding	No	Synthetic
[31]	Frequent pattern mining	Tree	Categorical	Sliding	No	Mixed
[40]	Sequential pattern mining	Counting	Categorical	Sliding	No	Mixed
[47]	Frequent pattern mining	Tree	Categorical	Sliding	No	Synthetic
[17]	Heavy hitter	Counting	Categorical	–	No	Mixed
[16]	Closed itemsets mining	Tree	Categorical	Sliding	No	Synthetic
[21]	Ratio rules	Counting	Quantitative*	Sliding	Yes	Mixed
[48]	Frequent pattern mining	Hashing	Categorical	Sliding	No	Synthetic
[30]	Frequent pattern mining	Tree	Uncertain	Sliding	No	Synthetic
[42]	Closed itemsets mining	Tree	Categorical	Sliding	No	Mixed
[43]	Frequent pattern mining	Hashing	Categorical	Sliding	No	Mixed
[15]	Fuzzy association rules	Tree	Fuzzy	Sliding	Yes	Mixed
[33]	Frequent pattern mining	Queue	Categorical	Sliding	No	Synthetic
[7]	Frequent closed graphs mining	Graph	Categorical	Sliding	No	Mixed
[27]	Rare itemsets mining	Tree	Categorical	Sliding	No	Mixed
[25]	Frequent pattern mining	Hyper-structure	Uncertain	Sliding	No	Mixed
[3]	Probabilistic pattern mining	Counting	Quantitative	Sliding	No	Mixed
[49]	Top-k frequent itemsets	Tree	Categorical	Sliding	No	Mixed

In addition, a high degree of interpretability is desirable. The main difference with frequent pattern mining is that rules are generated in an online fashion and in a single step, hence limiting the learner's range of action. Notice that the notion of concept drift is also present: the learnt model $\mathcal{M}$ (that is, the set of rules that the algorithm has discovered so far) at time t_i may not hold at time t_j. We formalize these concepts in the following manner: let $t \in \{t_1, t_2, \ldots, t_T\}$ be the distinct time indexes. Let $\mathcal{F}^t = \{f_1^t, f_2^t, \ldots, f_\ell^t\}$ be the set of ℓ features (both continuous and categorical). A fuzzy association stream rule is a relation $X^t \Rightarrow Y^t, X^t \cap Y^t = \emptyset$:

$$X^t = \bigwedge_{f_i^t \in A^t} \mu_{\tilde{A}}(f_i^t) \quad \text{and} \quad Y^t = \mu_{\tilde{C}}(f_c^t). \tag{4}$$

For convenience, we define the concept of an association stream rule simplifying the consequent part by forcing a single variable. The change on rule matching degree is directly related to the change in concept.

Regarding the measures of interestingness (support, confidence, etc.), these have to be adapted to the incremental nature of association stream as well.

This type of dynamic environment turns out to be of the upmost importance in many real-world applications (e.g., network security monitoring), being more practical than *pure* supervised methods when concept drift occurs. It is important to note that despite association stream mining being closely related to frequent pattern mining, this latter field is focused exclusively in the identification of the frequent variables, relegating the generation of the final rules in a second place using a classic offline approach, hence making it unfeasible for the challenges that association stream miners have to handle.

A similar challenging environment is in mining frequent itemsets from data streams due to the imposed constraints—single-pass limitation, memory limitations, combinational explosion of itemset candidates and handling drifting concepts [8]. In frequent pattern mining, a fast data structure—typically a tree [49]—is used for identifying the frequencies of streaming itemsets and storing those with higher frequencies, and typically they require the use of a temporal buffer to store the incoming data. However, and as aforesaid, this strategy is often ill-suited for handling association stream problems (i.e., they do not identify the whole pattern while the process is running), and it is most accentuated when a drift occurs, as some of the published papers assume no drifting concepts in data.

Also, the results on this field detail the number of rules obtained, the time and memory required for the technique at the end of the run. Differently, our methodology is based on analyzing how the algorithm behaves per concept, how concept drifts penalize the model and the time required to learn.

2.5. Related work

Table 1 shows a survey on online frequent pattern mining algorithms—and related disciplines—for data streams. It shows, for each method: (1) the category of the algorithm (i.e., the distinct *variations* on the main field), (2) the type of algorithm (i.e., how does the algorithm work), (3) the type of data that the algorithm can process, (4) the approach for tackling the stream (i.e., the type of windowing used, if any), (5) whether the algorithm produces rules, and (6) the type of experimentation used in the reference.

Category. In the literature we see the following categories: (1) *heavy hitter*—find the singleton items with a support greater than the given threshold—, (2) *top-k frequent itemsets*—find the top k frequent items in a stream—, (3) *frequent pattern*

mining—find all the frequent itemsets—, (4) *closed itemsets discoverer*—find those itemsets that have no frequent supersets with the same frequency, thus eliminating redundancy [8]—, (5) *rare itemsets mining*—itemsets that do not occur frequently—, (6) *ratio rules*—find the quantitative knowledge between distinct itemsets inside a rule—, (7) *frequent closed graphs mining*—find those graphs that have no frequent supersets with the same frequency—, (8) *probabilistic pattern mining*—find those itemsets with a probability greater than a minimum threshold—, (9) *association rules*—find all the frequent (quantitative) rules—, and (10) *fuzzy association rules*—find the all the frequent fuzzy rules.

Type. In the literature we find the following types of algorithms: (1) *counting-based*—an iterative counting algorithm—, (2) *tree-based*—a tree structure is deployed for pattern identification and extraction—, (3) *queue-based*—a queue structure is used for pattern discovery—, (4) *hashing-based*—one or more hash tables are exploited for frequent itemset discovery—, (5) *graph-based*—a graph structure is utilized for pattern discovery—, and (6) *hyper-structure-based*—a set of dynamic structures are combined for pattern identification.

Data. In the literature we find the following data: (1) *categorical* data, (2) *real* or *continuous* data, (3) *uncertain* data (i.e., probabilistic), and (4) *fuzzy* data. It is worth noting that the most common by far is the use of categorical data. Notice that the quantitative Ratio Rules (marked with a $*$) only apply for an interval composed of integers and not the general real-coded quantitative association rules.

Approach. In the literature we find the following approaches for tackling the stream, which are grouped into two distinct approaches: (1) the method does not distinguish recent items from older ones, learning from a static batch of instances—referred to as *landmark-window*, and (2) the algorithm gives more importance to recent transactions, learning from a dynamic batch of instances, referred to as *sliding-window* or *decay factor*, depending on how the algorithm process *old* data.

Rules. Most often, frequent pattern mining algorithms do not produce rules, as Table 1 reflects. However, a small fraction of these algorithm can generate rules naturally.

Experiments. In the literature we find the following two basic experimental setups: (1) synthetic environments and (2) real-world data sets. Table 1 shows that a mixture of both approaches is the most typical configuration. However, it is important to note that, in general, few experiments are performed, and a pair of synthetic/real-world environments being the most common configuration. Moreover, just a tiny fraction of the literature considers drifting concepts.

In this work we present Fuzzy-CSar with the aim of exploiting the online nature and highly adaptive behavior of fuzzy Michigan-Style LCSs to mine high-quality association rules from streams of unlabelled data. The learning architecture of Fuzzy-CSar is detailed in the next section.

3. Description of Fuzzy-CSar

Fuzzy-CSar is designed for mining fuzzy association rules from data that contain both quantitative and categorical attributes by means of combining GAs and apportionment of credit mechanisms in an online fashion. Hence, Fuzzy-CSar evolves a population of fuzzy association rules in an incremental way and it is able to quickly adapt to concept changes. Fig. 2 schematically illustrates Fuzzy-CSar. In what follows, the knowledge representation and the learning organization of Fuzzy-CSar are detailed.

3.1. Knowledge representation

Fuzzy-CSar evolves a *population* [*P*] of *individuals*, where each consists of (1) a *fuzzy association rule* and (2) a set of parameters that evaluate the quality of the rule. The fuzzy association rule is represented as **if** x_i is $\widetilde{A}_i^k$ and $\ldots$ and x_j is $\widetilde{A}_j^k$ **then** x_c is $\widetilde{A}_c^k$, in which the antecedent contains a set of ℓ_a input variables $x_i, \ldots, x_j$ ($0 < \ell_a < \ell$, where ℓ is the number of variables) and the consequent consists of a single variable x_c which is not present in the antecedent. It is worth noting that Fuzzy-CSar allows rules to have an arbitrary number of input variables in the antecedent. As depicted, each variable is represented by a disjunction of *linguistic terms* $\widetilde{A}_i^k = \{A_{i1} \vee \ldots \vee A_{in_i}\}$, where n_i is the total number of linguistic terms. In the experiments shown in this paper, all variables share the same semantics, which are defined by means of Ruspini's strong fuzzy partitions that satisfy the equality $\sum_{j=1}^{n_i} \mu_{A_{ij}}(x) = 1,\ \forall x_i$ (for more information the reader is referred to [18]). Each partition is a set of uniformly distributed triangular-shaped membership functions due to its interpretability tradeoff [11]. The matching degree of a data sample e with an individual (rule) k is computed as follows: for each antecedent variable x_i, the membership degree $\mu_{A_j^k}(e_i)$ of each of its linguistic terms is computed, and then these are aggregated by means of a T-conorm (disjunction). In this work, the *bounded sum* ($\min\{1, \mu_{A_i}(x) + \mu_{A_j}(x)\}$) is utilized as T-conorm, where $\mu_{\widetilde{A}_i^k} = \bigwedge \mu_{A_j^k}$ is the matching degree of all the variables of the antecedent part of the rule. The system is able to deal with missing values by considering that $\mu_{\widetilde{A}_i^k}(e_i) = 1$ if the value of the feature given by the environment e_i is not known. Next, the matching degree of the antecedent part of the rule is determined by the T-norm (conjunction), in this work using the *product* ($\prod \mu_{\widetilde{A}_i^k}(e_i)$) of the matching degree of all the input variables. Following that, the membership degree of each of the linguistic terms of the consequent variable ($\mu_{\widetilde{A}_c^k}$) is computed using the T-conorm. Finally, the matching degree of the whole rule is calculated using the *Dienes implication* ($\max\{1 - \mu_{\widetilde{A}}(x), \mu_{\widetilde{C}}(y)\}$) of the antecedent and consequent matching degrees.

Each individual has eight parameters that evaluate the quality of the rule: (1) the support *sup*, (2) the confidence *con*, (3) the lift *lif*, (4) the accuracy *acc*, (5) the fitness *F*, (6) the experience *exp*, (7) the numberosity *num*, which reckons the number of copies of the individual in the population, and (8) the average size of the action sets *as* in which the individual has participated.

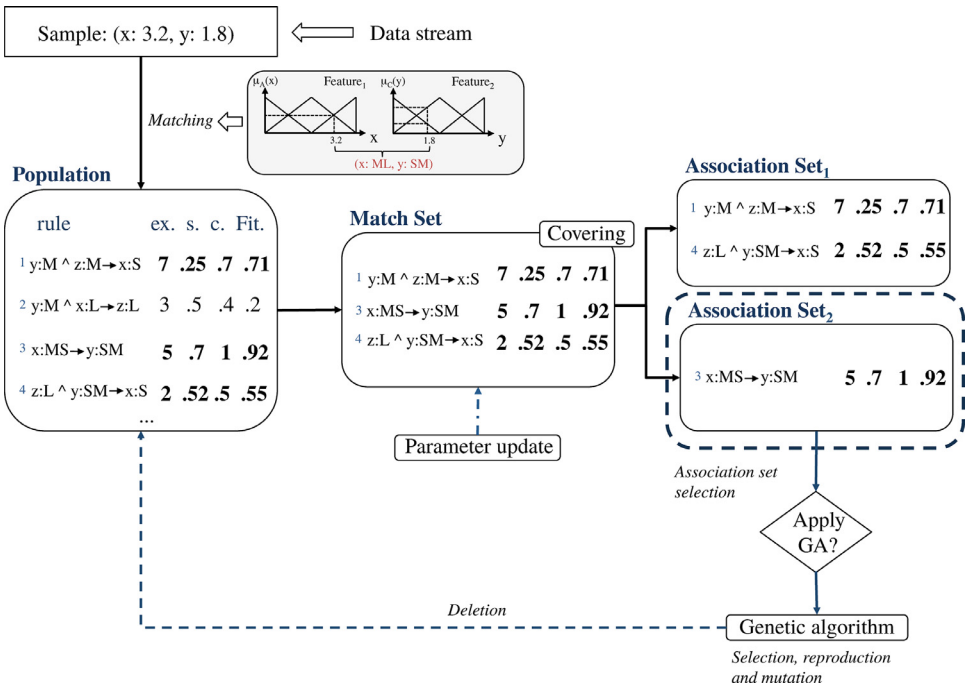

Fig. 2. Schematic illustration of the learning architecture of Fuzzy-CSar.

3.2. Learning interaction

Fuzzy-CSar incrementally learns from stream of data samples provided by the environment. That is, at each learning iteration, Fuzzy-CSar receives a sample and takes the following actions to update incrementally the individual's parameters and discover rules. Algorithm 1 shows a high-level description of the learning stage of Fuzzy-CSar. First, the system creates the *match set* [M]

Algorithm 1: A high-level description of the Fuzzy-CSar learning algorithm.

```
procedure TrainFuzzy-CSar( trainingSample e_t , Population [P] at time t )
Data: e_t has the form {x_i}_{i=1}^{ℓ}
Result: Population [P] at time t + 1
begin
    e_t' ← granulation(e_t);
    generate [M] out of [P] using e_t';
    if |[M]| < θ_mna then
        generate θ_mna − |[M]| matching individuals using e_t' and updating [P];
    end
    group individuals in [M] by their antecedent forming distinct [A]_i;
    select [A] probabilistically using Eq. (5);
    subsume individuals in [A];
    update individuals in [M] using Eqs. 6 to 10; // Therefore, all [A]_i are updated.
    if the average time of [A] since last GA > θ_GA then
        perform a genetic event in [A] considering e_t' and updating [P];
    end
end
```

with all the individuals in the population that match the data sample with a degree greater than 0 in both the antecedent and the consequent parts of the rule. If [M] contains less than θ_{mna} individuals, where θ_{mna} is a configuration parameter, the *covering operator* is triggered until having θ_{mna} individuals in [M]. Then, individuals in [M] are organized into different *association set candidates* $[A]_i$ grouping individuals by their antecedent. Each $[A]_i$ is given a probability to be selected that is proportional to the average confidence of the individuals that belong to it. The selected [A] goes through a *subsumption* process, which aims at reducing the number of rules that express similar associations among variables. Afterward, the parameters of all the individuals

in $[M]$ are updated in an incremental fashion. At the end of the iteration, a niche-based, steady-state GA [45] is applied to $[A]$ if the average time since its last application is greater than θ_{GA} (another user-defined parameter). It is important to note that the GA is not *generational*, so it functions in an online way. This process is repeatedly applied, therefore, updating the parameters of existing individuals and creating new promising rules online.

Six elements are needed to be further detailed in order to understand how the system works: (1) the covering operator, (2) the procedure to create association set candidates, (3) the association set subsumption mechanism, (4) the parameter update procedure, (5) the rule discovery by means of a GA and (6) the replacement mechanism. In the following, each one of these elements is described in more detail.

3.2.1. Covering operator

Given the sample data e, the covering operator generates a new individual that matches e with *maximum* degree: for each variable e_i, the operator randomly decides with probability $1 - P_\#$ (where $P_\#$ is a configuration parameter) whether the variable has to be in the antecedent of the rule, with the following two constraints: (1) that, at least, a variable has to be selected and (2) that, at most, $\ell - 1$ variables can be included in the antecedent. Then, one of the remaining variables is selected to be in the consequent. Each one of these variables is initialized with the linguistic label that maximizes the matching degree with the corresponding input value. Afterward, rule generalization is introduced by permitting the addition of any other linguistic term with probability $P_\#$ with the restriction that each variable in the antecedent and consequent contains a maximum number of linguistic terms at most. Fuzzy-CSar deals with missing values in e by ignoring the corresponding input variable e_j. Individual's parameters are set to initial values; that is, $sup = exp = 0$, $con = lif = acc = num = 1$, $F = 0.01$, and as is set to the size of the actual $[A]$.

As an illustrative example and following the schematic of Fig. 2, suppose that a new data sample $e = (x : 3, y : 1.9)$ is given by the environment and there are no matching individuals in the population. Then, the covering operator is triggered and a new matching individual is generated. First, this operator decides the variable in the antecedent part of the rule: assume that with probability $1 - P_\#$ variable y is selected as the antecedent variable (and, therefore, x is selected as the consequent). That is, if $P_\# = 0.4$ for each input variable, Fuzzy-CSar gets a uniform random number and, if this number is smaller than $1 - P_\# = 0.6$, the variable is selected as an antecedent. If we assume three linguistic terms, each one of the variables is initialized with the linguistic label that maximizes the matching degree with the corresponding input value, let us assume: x: M and y: L. Afterward, the rule generalization is triggered by randomly adding other linguistic terms with probability $P_\#$, so the final individual may end as **if** y is L **then** x is $\{M \vee L\}$ (notice that in this case only variable x has received a generalization event adding the new linguistic term L). Finally, the individual's parameters are set to its initial values.

3.2.2. Creation of association set candidates

The purpose of generating $[A]_i$ or niches is to group rules that express similar associations to establish a competition among them. This strategy lets the best individuals take over their niche hence evolving highly fit individuals. Fuzzy-CSar considers that two rules are similar if they have exactly the same variables in their antecedent, regardless of their corresponding linguistic terms. Therefore, this grouping strategy creates N_a candidates, where N_a is the number of rules in $[M]$ with different variables in the antecedent. The underlying principle is that rules with the same antecedent variables may express similar knowledge. For this purpose Fuzzy-CSar sorts $[M]$ according to the number of variables of the contained individuals. Once sorted, the different $[A]_i$ are generated by grouping those sharing the same variables in the antecedent part of the rule. Note that, under this strategy, rules with different variables in the consequent can be grouped in the same $[A]_i$. Afterward, the final set $[A]$ is selected following a roulette-wheel strategy, where each $[A]_i$ has a probability of being selected proportional to its accumulated confidence:

$$p^k_{sel} \leftarrow \frac{\sum_{i \in [A]_k} w_i \cdot con_i}{\sum_{j \in [M]} w_j \cdot con_j}, \tag{5}$$

where $w_i = 1$ if $exp_i > \theta_{exp}$ and $\frac{1}{10}$ otherwise, and θ_{exp} is a user-defined parameter.

3.2.3. Association set subsumption

A subsumption mechanism inspired by the one present in XCS was designed with the aim of reducing the number of different rules that express the same knowledge. The process works as follows: each rule in $[A]$ is checked for subsumption with each other rule of the set. A rule r_i is a candidate subsumer of r_j if it satisfies two conditions: (1) r_i has a similar confidence than r_j and it is experienced enough (that is, con^i is, at least, a 90% of con^j and $exp^i > \theta_{exp}$), and (2) r_i is more general than r_j. A rule r_i is more general than r_j if all the input and the output variables of r_i are also defined in r_j and r_i has, at least, the same linguistic terms than r_j for each one if its variables. Each time a rule r_i subsumes a rule r_j, r_i increases its numerosity count and the subsumed one r_j is deleted from population.

3.2.4. Parameter update

At the end of each learning iteration, the parameters of all individuals that belong to $[M]$ are updated. First, the experience of each individual is incremented. Second, the support of each rule is updated as

$$sup_{t+1} \leftarrow sup_t + \frac{\mu_{\tilde{A}}(e) \cdot \mu_{\tilde{C}}(e) - sup_t}{exp}, \tag{6}$$

where $\mu_{\widetilde{A}}(e)$ and $\mu_{\widetilde{C}}(e)$ are the matching degree of the antecedent and the consequent respectively. Then, the confidence is computed as

$$con_{t+1} \leftarrow \frac{imp_{t+1}}{ant\_mat_{t+1}}, \tag{7}$$

where $imp_{t+1} \leftarrow imp_t + \mu_{\widetilde{A}}(e) \cdot \max\{1 - \mu_{\widetilde{A}}(e), \mu_{\widetilde{C}}(e)\}$, and $ant\_mat_{t+1} \leftarrow ant\_mat_t + \mu_{\widetilde{A}}(e)$. Initially, both imp_t and ant_mat_t are set to 0. Next, the lift is calculated as

$$lif_{t+1} \leftarrow \frac{sup_{t+1}}{ant\_mat_{t+1} \cdot con\_mat_{t+1}}, \tag{8}$$

where $con\_mat_{t+1} \leftarrow con\_mat_t + \mu_{\widetilde{C}}(e)$. Similarly to ant_mat_t, con_mat_t is initially set to 0. Next, the accuracy is updated as

$$acc_{t+1} \leftarrow sup_{t+1} + 1 - (ant\_mat_{t+1} + con\_mat_{t+1} - sup_{t+1}). \tag{9}$$

Thereafter, the fitness of each rule in $[M]$ is calculated as

$$F \leftarrow \left(\frac{sup_{t+1} \cdot lif_{t+1} + acc_{t+1}}{2}\right)^{\nu}, \tag{10}$$

where ν is a user-set parameter that permits to control the pressure toward highly fit individuals. It is worth noting that the fitness function can result in individuals with fitness values greater than one, so if it happens, the fitness value is truncated to one. The fitness computation has been designed using the recommendations found in [32]. Finally, the association set size—notice that this size is computed after the subsumption mechanism—estimate of all rules that belong to the selected $[A]$ is computed as the average size of all the $[A]$'s in which it has participated.

3.2.5. Discovery component

Fuzzy-CSar uses a steady-state, niche-based, incremental GA to discover new and promising rules. The GA is applied to the selected $[A]$. The GA is triggered when the average time from its last application upon the individuals in $[A]$ exceeds the threshold θ_{GA}, a user-defined parameter. It selects two parents p_1 and p_2 from $[A]$ using tournament selection [37]. The two parents are copied into offspring ch_1 and ch_2, which undergo crossover and mutation. Fuzzy-CSar uses a uniform crossover operator which crosses the antecedents of the rules by two points, permitting cross-points within variables. It is applied with probability P_χ. The resulting offspring may go through three different types of mutation: (1) mutation of the antecedent variables (with probability $P_{I/R}$), which randomly chooses whether a new antecedent variable has to be added to the rule or one of the antecedent variables has to be removed from it; (2) mutation of the consequent variable (with probability P_C), which selects one of the variables of the antecedent and exchanges it with the variable of the consequent; and (3) mutation of the linguistic terms of the variables (with probability P_μ), which selects one of the existing variables of the rule and mutates its value in one of the three possible ways: *expansion*, *contraction* or *shift*. Expansion chooses a linguistic term not represented in the corresponding variable and adds it to this variable; thus, it can be applied only to variables that do not have all the linguistic terms. Contraction selects a linguistic term represented in the variable and removes it; so, it can be applied only to variables that have more than one linguistic term. Shift switches a linguistic term by its immediate inferior or superior one. The parameters of the offspring are initialized as follows: if the crossover is not applied, the parameters are copied from the selected parent. Otherwise, these are set to the average value between the corresponding parameters in the parents. In both cases, the fitness is decreased to 10% of the parental fitness. Experience is set to 0 and numerosity to 1.

3.2.6. Replacement mechanism

The new offspring are introduced into the population, but first each individual is checked for subsumption with their parents—the same procedure for association set subsumption is followed. If any parent is identified as a possible subsumer for the offspring, the offspring is not inserted into the population and the numerosity of the parent is increased. Otherwise, $[A]$ is checked for the most general rule that can subsume the offspring and if no subsumer can be found, the individual is finally inserted into $[P]$.

If the population is full, exceeding individuals are deleted from $[P]$ with probability directly proportional to their association set estimate and inversely proportional to its fitness. Moreover, if an individual k is sufficiently experienced, and its fitness F^k is significantly lower than the average fitness of the individuals in $[P]$, its deletion probability is further increased:

$$p^k \leftarrow \frac{d^k}{\sum_{\forall j \in [P]} d^j}, \tag{11}$$

where

$$d^k \leftarrow \begin{cases} \frac{as \cdot num \cdot F_{[P]}}{F^k} & \text{if } exp^k > \theta_{del} \text{ and } F^k < \delta F_{[P]}, \\ as \cdot num & \text{otherwise}, \end{cases} \tag{12}$$

where θ_{del} and δ are two configuration parameters. Thus, the deletion algorithm balances the individual's allocation in the different $[A]$'s by pushing toward the deletion of rules belonging to large association sets. At the same time, it favors the search towards highly fit individuals, since the deletion probability of rules whose fitness is much smaller than the average fitness is increased.

3.3. Cost of the algorithm

Similarly to other Michigan-style LCS [37], the cost of the algorithm increases linearly with the maximum population size N, the maximum number of variables Max_{var} used per rule, and semi-logarithmically with the cost of sorting the match set, as depicted in Eq. (13):

$$Cost_{Fuzzy\text{-}CSar} = O(N \cdot Max_{var} \cdot |[M]| \cdot \log |[M]|), \tag{13}$$

where $|[M]|$ is the size of the match set. It is important to note that Fuzzy-CSar does not depend directly on the number of transactions, which makes it very desirable for mining huge databases. The knowledge is continuously evolved and can be consulted anytime so it provides the capability to monitor the environment in an online fashion. Fuzzy-CSar has evolved since its first conception [36]; the candidate generation is different from the original version, where it was proposed for offline quantitative association rule mining. As its learning architecture supported online learning, we upgraded its original components to the problem of association stream mining. We refined its operators for data stream processing.

3.4. Parameter setting guidelines

As in common with many related members of the Michigan-style LCS family, Fuzzy-CSar has several parameters that have to be properly configured in order to obtain high quality solutions. These are explained in the following:

- **Population size** N. It specifies the available workspace for the evolutionary search. It is critical to set this parameter high enough to prevent the deletion of accurate individuals.
- **Generalization in initialization** $P\#$. It controls the degree of generalization of Fuzzy-CSar, thus it is important to set it large enough in order to let the algorithm to generalize. However, it should be set small enough to avoid the generation of over general individuals.
- **Fitness pressure** ν. In XCS and its derivatives, this parameter drives the genetic search towards an accurate set of individuals avoiding over generals [37], and it is common to set it to 10. However, in the particular case of Fuzzy-CSar, this parameter has little effect, as shown through experimental support.
- **GA frequency threshold** θ_{GA}. This threshold controls the application rate of the evolutionary discovery component. Ideally, it has to be set small enough to allow the discovery of new and interesting solutions, but large enough to prevent forgetting.
- **Crossover probability** P_χ. It controls the application rate of the crossover operator. Since all GAs work in roughly the same way–decomposition and reassembly—having this parameter set to a high value is a common practice.
- **Mutation rates** $\{P_C, P_{l/R}, P_\mu\}$. Typically, it is set to a low value (e.g., $1/\ell$) to minimize the deletion of good solutions. Studies related to GAs agree with the notion that they should be kept small [37].
- **Other thresholds** $\{\theta_{exp}, \theta_{del}, \theta_{sub}, \theta_{mna}, \delta\}$ These define the behavior of Fuzzy-CSar in certain situations such as the minimum experience required for an individual in order to subsume, the minimum experience required for being deleted from $[P]$, the number of covered classifiers, or the ratio of the average population fitness used in the deletion phase. Studies on Fuzzy-UCS [37] suggest that these parameters have little effect on the behavior of the algorithm.

3.5. Insights on why Fuzzy-CSar works

Michigan-style LCSs are open frameworks that foster crossbreeding between different learning paradigms. Butz [10] identified a set of pressures that guide Michigan-style LCSs to obtain accurate results and that explain why LCSs work. These are the following: the fitness pressure which pushes $[P]$ toward more accurate individuals, the set pressure and the subsumption pressure which pushes $[P]$ toward generalization, the mutation pressure which pushes toward more specific solutions, and the deletion pressure which pushes $[P]$ toward fittest individuals. Notice that this deletion scheme erases individuals that are no longer useful due to a concept drift. Despite the fact that these studies are referred to XCS, these can be extrapolated to other systems that follow the same framework.

3.6. Example

Assume a three dimensional problem, each variable ranging in [0, 100], and three uniformly distributed triangular-shaped membership functions per variable (S, M and L). Fuzzy-CSar starts with an empty population and the following data sample is sent from the environment: $e = \{10, 50, 45\}$. Then, the covering mechanism is triggered and θ_{mna} individuals are generated. Assuming $\theta_{mna} = 2$:

- I1: x_1 *is* $\{S\} \Rightarrow x_3$ *is* $\{M\}$,
- I2: x_2 *is* $\{M\} \wedge x_3$ *is* $\{M\} \Rightarrow x_1$ *is* $\{S\}$,

and are inserted into $[M]$. Next, the individuals are updated as follows, assuming $\nu = 1$:

- I1.$exp = 1$, I1.$sup = 0.72$, I1.$con = 0.9$, I1.$lif = 1$, I1.$acc = 0.74$, I1.$fitness = 0.73$,
- I2.$exp = 1$, I2 $sup = 0.72$, I2.$con = 0.6$, I2.$lif = 1$, I2.$acc = 0.74$, I2.$fitness = 0.73$.

Afterward, these individuals are grouped into the distinct $[A]_i$: in this case two distinct $[A]_i$ are generated, one for each individual. Because I1 has more confidence, it has more probability of being selected as $[A]$ for applying the genetic event. However, assuming $\theta_{GA} = 10$, this event will not be triggered at this point due to the low experience of the individuals. A new data sample is sent from the environment, $e = \{5, 0, 74\}$, and $[M]$ is generated containing I1. Notice that I2 is not in $[M]$ because the matching degree of I2 is 0. A new matching individual is generated and inserted into $[M]$:

- I3: x_1 *is* $\{S\} \Rightarrow x_2$ *is* $\{S\}$.

As before, elements in $[M]$ are evaluated:

- I1.*exp* $= 2$, I1.*sup* $= 0.594$, I1.*con* $= 0.917$, I1.*lif* $= 0.985$, I1.*acc* $= 0.323$, I1.*fitness* $= 0.454$,
- I3.*exp* $= 1$, I3.*sup* $= 0.9$, I3.*con* $= 1$, I3.*lif* $= 1$, I3.*acc* $= 0.9$, I3.*fitness* $= 0.9$.

Next, I1 and I3 are grouped into the same $[A]_i$ because they share the same antecedent variables. Because the low average experience of this $[A]$ (much smaller than θ_{GA}), the genetic event will not be triggered; it will be triggered later in the run when the average experience of the selected $[A]$ exceeds θ_{GA}. Once the GA is triggered, two parents are selected. Assuming I1 and I3 as the selected parents, they generate O1 and O2. Crossover and mutation are then triggered:

- O1: x_1 *is* $\{S\} \Rightarrow x_3$ *is* $\{M\}$,
- O2: x_2 *is* $\{S\} \Rightarrow x_1$ *is* $\{M\}$.

Whereas O1 remains identical to I1 and is subsumed (increasing I1.*num*), O2 has mutated and is inserted into $[P]$ (as I4). Afterward, exceeding classifiers are deleted from $[P]$ based on their fitness. This learning cycle will be repeated until the end of the stream. An association drift may happen and, therefore, old matching rules will not be accurate. Suppose that now the relation x_1 *is* $\{S\} \Rightarrow x_3$ *is* $\{M\}$ does not hold and a new data sample comes: $e = \{7, 1, 1\}$. Then $[M]$ is generated containing I1, I3 and I4. Notice that now these rules will be penalized as their consequents do not fully match with the input example, so their support, confidence, lift and accuracy will decay. Therefore, the GA will be triggered and new and matching individuals will be generated containing the new relationship. It is important to note that individuals are not deleted instantly but they are left alive a certain amount of time—depending on their numerosity parameter. This way we avoid premature deletion of patterns.

Knowing the intrinsics of Fuzzy-CSar, in the subsequent sections, the system is analyzed in a set of different environments.

4. Experiments on association stream mining

Our hypothesis is that Fuzzy-CSar is able to accurately model what the stream of information hides (in terms of rule support, confidence, lift and accuracy) and that it is able to adapt to distinct drifting concepts due to its adaptive architecture. For that purpose, we designed a series of experiments with distinct complexities that model extreme cases of real-world scenarios. These are detailed in the following.

4.1. On the difficulty of evaluating association streams

We want to test our algorithm on problems with similar complexities as those that would appear in real-world association stream environments. However, unlike in supervised and clustering stream fields (see SEA-related problems [39] and MOA for instance), for association stream mining, there is no formal way to quantitatively evaluate what happens with the learned model when a concept drift occurs. That is mainly due to the fact that in association stream we do not rely on a class label or cluster to perform evaluations (i.e., Kappa statistic and *F*-measure, just to mention two in the supervised case and the validation indexes in the case of clustering). Typically, in the closely related online frequent pattern mining field, concept drifts are either avoided by the use of statistics over the incoming data—thus eliminating the old model once it has occurred and re-learning a new one—or not handled under the assumption that the stream will not have any drift. In this regard, the difficulty is how to evaluate the performance of the algorithm.

To test Fuzzy-CSar under association stream we have designed an environment that allows us to generate online data streams from a series of predefined rules. The data are then generated from uniformly distributed and triangular-shaped membership functions. Because the cut-off points are known from the predefined rules, variables are set using a pseudo-random generator forcing them to have a membership degree greater than 0.5 according to the respective fuzzy label in the predefined rule. The underlying idea is that the algorithm should discover the rules out of the stream of data and adapt to drifting concepts. Our proposal uses a hold out strategy generating two distinct sets (train and test sets) of samples out of the predefined rules in an online fashion. Fig. 3 schematically shows this process. It is important to note that by proceeding this way we may introduce some bias in the results, but we decided to experiment with it this way as (1) we needed a framework for analyzing the results quantitatively and to the best of our knowledge no other frameworks are available for association stream and (2) other Fuzzy-Michigan Style LCSs have been found to be accurate under distinct environments [39].

The procedure to evaluate the system is the following: after a training stage, the system enters in test mode. During the test stage the environment sends previously unseen samples. We searched for the best matching rule out of $[M]$, that is, the rule that matches both antecedent and consequent with *maximum matching degree*. If there is a tie, we select the most general rule. The chosen rule is compared with the predefined (optimal) rule that has generated the test sample. Normalized Hamming distance between these two binary-coded rules is computed. Basically, we count the number of linguistic terms that differs between both

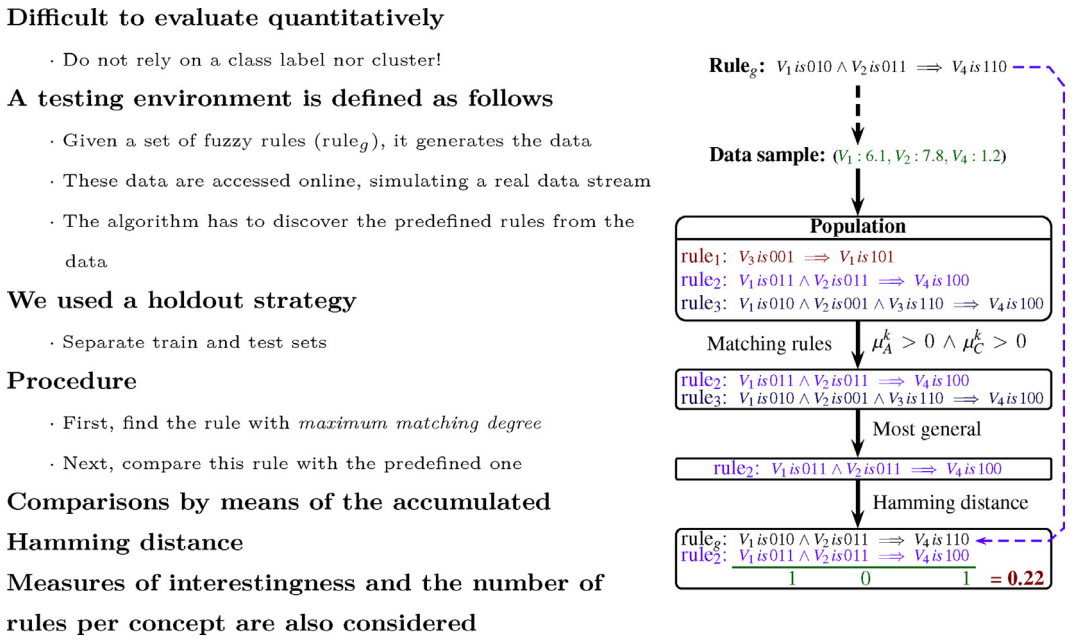

Fig. 3. Schematic illustration of the experimental methodology used to generate online data streams.

rules and divide by the maximum number of available linguistic terms to get a number between 0 (which means we found the optimal rule) and 1 (which would means the chosen rule has nothing to do with the optimal one). If the sample does not trigger any rules of the pool, error 1 is assigned.

4.2. Methodology of experimentation

Fuzzy-CSar has several configuration parameters which enable it to adjust the behavior of the system to evolve models of maximal quality. The configuration parameters have been obtained experimentally following the recommendations found in [35]. This study detected that Michigan-style LCSs are sensitive to the generalization in initialization $P_\#$ and the fitness pressure ν, while the setting of the other parameters had little effect on the final behavior. With this information in mind, we selected values that, on average, allow Fuzzy-CSar to perform well on all the problems. In this regard Fuzzy-CSar was configured with the following parameters for all the experiments: $\nu = 1$, $P_\# = 0.2$, $P_\chi = 0.8$, $\{P_{I/R}, P_\mu, P_C\} = 0.1$, $\delta = 0.1$, $\theta_{\text{GA}} = 12$, $\theta_{exp} = 24$, $\{\theta_{del}, \theta_{sub}\} = 20$, $maxLingTermsPerVariable = 3$, θ_{mna} automatically sets to the number of variables, and the population was set to 1 000 individuals. In addition, all the variables use Ruspini's strong fuzzy semantics with five linguistic terms (*XS*, *S*, *M*, *L* and *XL*). Results are averages of 30 runs, using each time a different random seed.

4.3. Experiment 1

4.3.1. Description

In the first experiment we want to check the reaction capacity of Fuzzy-CSar under distinct rule drifts. This environment is inspired by the Hulten et al. rotating hyperplane problem [28] and contains abrupt concept drifts and noisy inputs. The problem is described as a multi-dimensional space defined by a set of eight rules which make use of three continuous variables $\{x_1, x_2, x_3\}$ ranging between [0, 1] from which only two are relevant at a given concept. The three variables swap positions in each concept in the following way: during the first concept the variable x_1 is in the antecedent part, x_2 is in the consequent part, and x_3 is left taking random values (i.e., being a noisy input). In the second concept, however, a swap in the antecedent/consequent variable position happens and the variable x_2 now appears in the antecedent part of the rule and x_1 in the consequent. In the third concept another change happens and x_1 returns to the antecedent part of the rule. The variable x_3 is in the consequent during the third concept. This time, the variable x_2 takes random values. The fourth and final drift happens in the same way as in the simple rule determination problem, that is: the variable x_1 is in the antecedent part of the rule, x_2 is in the consequent part, and x_3 takes random values. Table 2 shows the new rule base for the problem.

The association stream consists of 50 000 samples and the concept is changed every 12 500. Thus, the system has four different concepts and 12 500 randomly generated training samples per concept. The pseudo-random generator uses a uniform distribution. To test the models, independent sets consisting of 2500 test samples are generated per concept. The learner is trained for 500 data samples and then its model is evaluated by the corresponding test set in order to plot the results.

Table 2
Predefined rule base used to generate the first experiment on association stream. The consequent variable is accentuated in bold. Notice that, from concept to concept, one variable is left taking random values.

Concept 1	x_1					$\mathbf{x_2}$					Concept 3	$\mathbf{x_3}$					$\mathbf{x_1}$				
	VS	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*		*VS*	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*
R_1			*X*	*X*	*X*			**X**	**X**	**X**	R_1	*X*	*X*				**X**	**X**			
R_2	*X*	*X*				**X**	**X**				R_2				*X*	*X*				**X**	**X**
Concept 2	$\mathbf{x_2}$					$\mathbf{x_1}$					Concept 4	$\mathbf{x_1}$					$\mathbf{x_2}$				
	VS	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*		*VS*	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*
R_1	*X*	*X*	*X*			**X**	**X**	**X**			R_1			*X*	*X*	*X*	**X**	**X**	**X**		
R_2				*X*	*X*				**X**	**X**	R_2	*X*	*X*							**X**	**X**

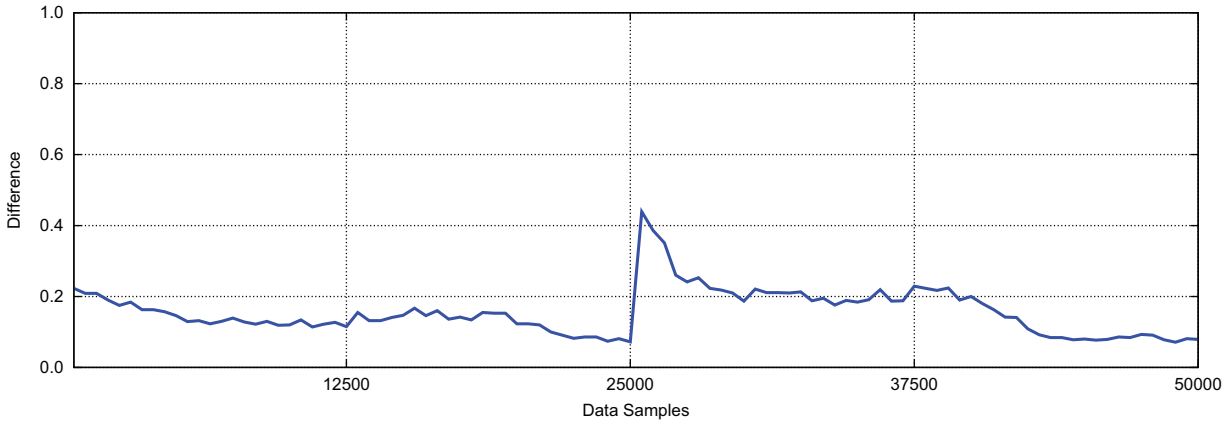

Fig. 4. Results obtained in the first experiment. Every 12 500 data samples there is a concept drift. Curves are averages of 30 runs.

Table 3
Number of rules and its average quality in terms of support, confidence, lift and accuracy at the end of each concept.

Concept	Rules	Sup	Con	Lif	Acc
1	64.03 ± 6.86	0.35 ± 0.01	0.72 ± 0.02	1.17 ± 0.07	0.59 ± 0.02
2	64.70 ± 4.21	0.35 ± 0.01	0.71 ± 0.02	1.13 ± 0.06	0.57 ± 0.02
3	64.20 ± 5.27	0.31 ± 0.01	0.65 ± 0.03	1.04 ± 0.09	0.52 ± 0.02
4	64.13 ± 5.67	0.33 ± 0.01	0.69 ± 0.02	1.09 ± 0.06	0.55 ± 0.02

Three key aspects are interesting to be analyzed in the results: (1) the time the algorithm requires to learn during the first stages of the problem, (2) the reaction of the algorithm to sudden concept drifts, and (3) the number of rules and its *quality* at the end of each concept.

4.3.2. Results

In the following we elaborate the interesting aspects of the experiment results.

- **Time required to learn during the first stages of the problem.** Fuzzy-CSar starts with an error close to 20%, a relatively low value. Moreover, the learning curve is step and at the end of the first concept it reaches an error about 15% .
- **Reaction against sudden concept drifts.** As shown in Fig. 4, the first concept drift causes no trouble to the proposed algorithm. That is partly due to the heuristics of Fuzzy-CSar when generating the covering individuals and partly due to the genetic discovery. The third concept is also very interesting because x_3, the noisy variable now, is suddenly swapped with x_2. Also, x_1 returns to the antecedent of the rule. Is precisely in this concept where the differences with the previous problem arise: at the beginning it rockets the error curve above 40%. Fuzzy-CSar has a fast reaction capacity and in a few data samples it lowers the error curve to near 20%. Again, the two main sources of learning help it to adapt quickly. Finally, in the fourth concept, the variable x_2 is swapped with x_3 again. In this last part of the problem, Fuzzy-CSar returns to a very low error rate, ending up 10%.
- **Number of rules and its *quality* at the end of each concept.** Interestingly the number of rules discovered are far less than the population limit of 1000 individuals for this problem, as depicted in Table 3. It is worth mentioning the high values of average confidence and lift obtained by the search engine of Fuzzy-CSar.

Table 4
Predefined rule base used to generate the second experiment stream. The consequent variable is accentuated in bold.

Concept 1	$\mathbf{x_1}$					$\mathbf{x_2}$					$\mathbf{x_3}$				
	VS	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*
R_1			*X*	*X*	*X*	**X**	**X**	**X**							
R_2			*X*	*X*	*X*								**X**	**X**	**X**

Concept 2	$\mathbf{x_1}$					$\mathbf{x_2}$					$\mathbf{x_3}$				
	VS	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*
R_1	*X*	*X*			*X*			**X**	**X**	**X**					
R_2	*X*	*X*			*X*						**X**	**X**	**X**		

Concept 3	$\mathbf{x_1}$					$\mathbf{x_4}$					$\mathbf{x_2}$					$\mathbf{x_3}$				
	VS	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*
R_1	*X*	*X*						*X*						**X**	**X**					
R_2	*X*	*X*						*X*								**X**	**X**			

Finally, the average time required by Fuzzy-CSar for processing a single data sample (i.e., all the training stage of Algorithm 1) in this problem using our configuration is of 0.000103 ± 0.000014 s.

4.4. Experiment 2

4.4.1. Description

This problem has the particularity that rules with the same antecedents have different consequents. This issue makes the problem difficult for any association stream algorithm. Also, the number of variables is increased in the last concept. The problem uses four continuous variables $\{x_1, x_2, x_3, x_4\}$ ranging between [0, 1]. During the first concept, two distinct rules are generated. While these two rules have distinct consequent variables (x_2 and x_3, respectively), both share the variable x_1, which is always in the antecedent part of the rule. Then an abrupt concept drift occurs by completely changing the fuzzy labels of each variable. It is worth mentioning that in the two first concepts the variable x_4 takes random values. Finally, in the last concept, another major concept drift happens adding the variable x_4, jointly with x_1 in the antecedent part of the rules. Table 4 depicts the rule base used to generate the data of this problem.

The data stream consists of 37 500 data samples and the concept is changed every 12 500 samples. The training data samples are randomly generated following the corresponding rule base and using a uniform distribution. To test the models, independent sets consisting of 2500 test samples are generated per concept. The learner is trained for 500 data samples and then its model is evaluated by the corresponding test set.

4.4.2. Results

The time Fuzzy-CSar requires to learn during the first stages of the problem, its reaction to concept drifts, and the number of rules and their quality at the end of each concept are analyzed in the following.

- **Time required to learn during the first stages of the problem.** The algorithm starts near a 25% error, a low error value, and it maintains close to this value during the first concept.
- **Reaction against concept drifts.** When the first concept drift occurs the error rockets beyond 40%. Fuzzy-CSar has a fast reaction capacity and a few data samples later it lowers the error curve below 20%. Finally, in the last concept, a major concept drift occurs and Fuzzy-CSar first increases its error rate above 35% to, in a few thousand data samples later, end near 15%, a low error value, as depicted in Fig. 5.
- **Number of rules and its *quality* at the end of each concept.** As expected the number of rules at the end of each concept is higher than in the previous one due to its difficulty, shown in Table 5. Notice the values of the quality measures in the second concept: these are the effects of the mixture of rules. Interestingly, in the subsequent concept they rise, showing that Fuzzy-Csar is highly adaptive.

Finally, the average time required by Fuzzy-CSar for processing a single data sample in this problem using our configuration is of 0.000045 ± 0.00001 s.

4.5. Experiment 3

4.5.1. Description

This problem extends some of the ideas from the previous problems but in a higher dimensional search space and in which each concept has a different number of rules and length. This problem makes use of five continuous variables $\{x_1, x_2, x_3, x_4, x_5\}$ ranging between [0, 1] from which, depending on the concept, some of these variables are relevant. In the first concept, the rule to be studied uses the first four variables. In the second concept the following variables $\{x_1, x_2, x_4\}$ are the relevant ones. Finally,

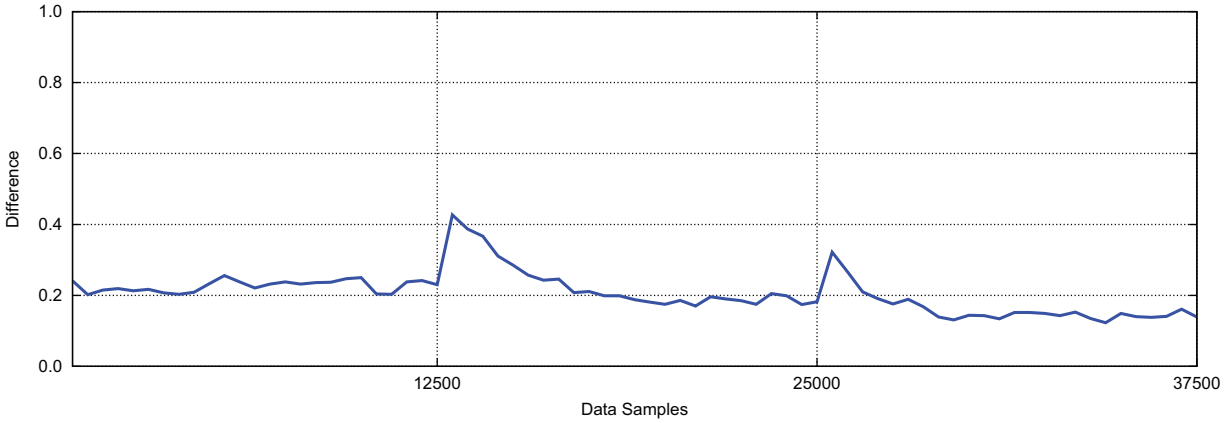

Fig. 5. Results obtained in the second experiment. Every 12 500 data samples there is a concept drift.

Table 5
Number of rules and their average quality in terms of support, confidence, lift and accuracy at the end of each concept.

Concept	Rules	Sup	Con	Lif	Acc
1	101.90 ± 6.45	0.41 ± 0.04	0.75 ± 0.03	1.04 ± 0.01	0.54 ± 0.02
2	100.83 ± 10.38	0.15 ± 0.02	0.49 ± 0.11	0.70 ± 0.22	0.46 ± 0.02
3	104.63 ± 7.30	0.42 ± 0.03	0.74 ± 0.03	1.01 ± 0.01	0.55 ± 0.01

Table 6
Rule base used to generate the third problem. The consequent variable is accentuated in bold. Notice that, from concept to concept, some variables are left taking random values.

Concept 1	$\mathbf{x_1}$					$\mathbf{x_2}$					$\mathbf{x_3}$					$\mathbf{x_4}$				
	VS	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*
R_1	X									X	X					**X**				
R_2		X							X			X					**X**			
R_3			X					X					X					**X**		
R_4				X					X			X							**X**	
R_5					X					X	X									**X**
Concept 2						$\mathbf{x_2}$					$\mathbf{x_3}$					$\mathbf{x_4}$				
						VS	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*
R_1						X									X					**X**
R_2							X							X				**X**		
R_3								X					X			**X**				
Concept 3	$\mathbf{x_1}$					$\mathbf{x_2}$					$\mathbf{x_3}$					$\mathbf{x_4}$				
	VS	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*	*VS*	*S*	*M*	*L*	*VL*
R_1	X															**X**				
R_2								X										**X**		
R_3															X					**X**

in the third concept, the following variables $\{x_1, x_2, x_3, x_4\}$ are the relevant ones, but in distinct rules. Table 6 shows the rule base used to generate the problem. In addition, the random generator uses the normal distribution $N(0, 1)$.

The data stream consists of 50 000 data samples and the concept is changed in the following way: the first concept lasts for 20 000 data samples, the second lasts for 15 000 samples, and the third concept lasts for 15 000 data samples. The samples are randomly generated following the corresponding rule base. To test the models, independent sets consisting of 2500 test samples are generated per concept. The learner is trained for 500 data samples and then its model is evaluated by the corresponding test set.

4.5.2. Results

In the following the key aspects of the results obtained by Fuzzy-CSar are analyzed.

- **Time required to learn during the first stages of the problem.** As Fig. 6 depicts, the algorithm starts with an error rate of about 30%, which quickly lowers to and stays stable on about 25% and remains near this value.

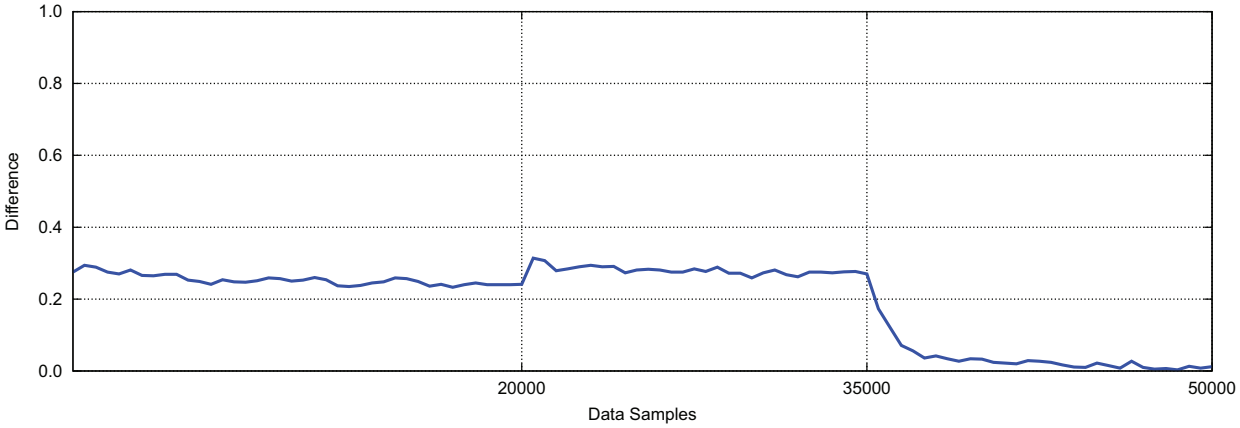

Fig. 6. Results obtained in the third experiment. Concept drift happens at iteration 20 000 and 35 000.

Table 7
Number of rules and its average quality in terms of support, confidence, lift and accuracy at the end of each concept.

Concept	Rules	Sup	Con	Lif	Acc
1	112.73 ± 6.63	0.08 ± 0.03	0.66 ± 0.02	1.93 ± 1.15	0.71 ± 0.01
2	107.17 ± 7.21	0.02 ± 0.01	0.47 ± 0.12	1.36 ± 3.01	0.82 ± 0.01
3	137.43 ± 8.41	0.02 ± 0.01	0.48 ± 0.11	1.27 ± 2.75	0.79 ± 0.02

- **Reaction against concept drifts.** The first concept drift happens and Fuzzy-CSar shows a very small error peak. This change is soft enough for Fuzzy-CSar to adapt it. During this stage the algorithm improves its error. However, the noise added by x_3 and x_5 affects Fuzzy-CSar negatively. Finally, in the third concept, Fuzzy-CSar is able to recover to nearly a 1% error rate in a step curve. Notice the flat appearance of the error curve in the second concept. Despite the fact that Fuzzy-CSar has trouble at finding the exact rules, its generalization capacity is actuating and the number of individuals used is reduced during the entire concept.
- **Number of rules and its*quality*at the end of each concept.** Table 7 shows the number and quality of the rules obtained by Fuzzy-CSar. The low value of average support and the large variation in lift is noticeable.

Finally, the average time required by Fuzzy-CSar for processing a single data sample in this problem using our configuration is of 0.000057 ± 0.000001 s.

4.6. Discussion

Fuzzy-CSar has shown a remarkable response in a series of online environments that model extreme conditions that we could be found in real-world situations where a quick response is critical (e.g., Smart Grids monitoring). These experiments allowed us to obtain quantitative results on how well the algorithm performs under association streams. As we hypothesized, the online architecture is able to accurately model the hidden information from the stream of data. Moreover, the experiments done support that the presented framework is able to quickly adapt to drifting concepts.

In the following section, Fuzzy-CSar is carefully analyzed under a real data stream environment with unknown dynamics.

5. Experiment on a real data stream problem

First described by Harries et al. [24], the New South Wales electricity market problem has been used broadly in data stream literature [34] as a real problem with unknown dynamics. The problem consists of the modeling of the electricity market of a particular area of Australia to extract key knowledge that enables experts to create a compelling online environment for it. The data set contains 45 312 samples, each one of these data refers to a period of 30 min, and they have eight fields: the time stamp, the day of the week, the period, the New South Wales electricity price, the New South Wales electricity demand, the Victoria electricity price, the Victoria electricity demand and the scheduled electricity transfer between states. The class has been removed from the data set in order to perform an unsupervised learning. The procedure used in this experiment is the following: Fuzzy-CSar is trained for 336 samples (i.e., every *week*), and then the quality of the output rules is analyzed. The hypothesis is that Fuzzy-CSar will be able to find high quality rules in terms of support and confidence and, from week to week, the algorithm will conserve much of its previously inferred rules, adapting itself to the unknown dynamics of the problem. To do so, after a *week* (336 samples), the output rules obtained by Fuzzy-CSar are compared with the ones obtained in the previous week. Only identical rules that use exactly the same variables and linguistic terms are considered.

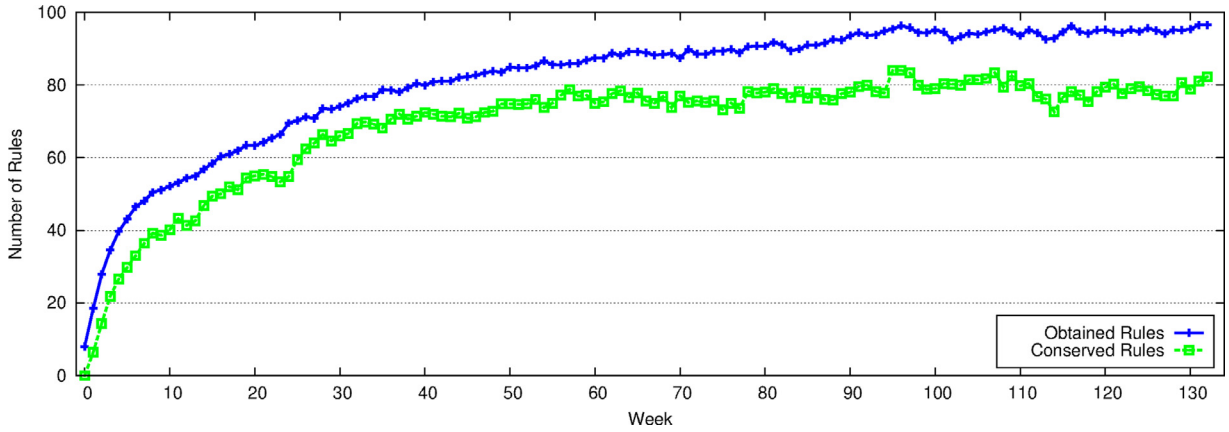

Fig. 7. Number of rules obtained with a confidence greater than 0.95 on the New South Wales Electricity market problem.

5.1. Methodology of experimentation

In this experiment, Fuzzy-CSar was configured with a population size of 400 rules and the following parameters: $\nu = 1$, $P_{\#} = 0.2$, $P_{\chi} = 0.8$, $\{P_{l/R}, P_{\mu}, P_{C}\} = 0.1$, $\delta = 0.1$, $\theta_{\text{GA}} = 25$, $\theta_{exp} = 50$, $\{\theta_{del}, \theta_{sub}\} = 20$, $maxLingTermsPerVariable = 3$, and $\theta_{mna} = 2$. All the variables use Ruspini's strong fuzzy semantics with five linguistic terms (*XS*, *S*, *M*, *L* and *XL*). The experiment has been repeated for 30 runs, each with a different random seed. Results are the averages of these runs.

5.2. Analysis of the results

Fig. 7 depicts the number of rules with a confidence greater than 0.95 obtained by the proposed algorithm. As we can see, the algorithm is able to obtain and keep rules of good quality. It is clearly visible that, as expected, a large part of the rules are conserved between week periods.

In order to show the knowledge provided by Fuzzy-CSar, an interesting couple of the discovered rules are depicted in the following. The first rule describes the feature *transfer*, which refers to the scheduled electricity transfer between Victoria and New South Wales states.

R$_1$: **IF** *NSWPrice* **is** {*XS* or *S* or *M*} **and** *NSWDemand* **is** {*S* or *M* or *L*} **THEN** *transfer* **is** {*S* or *M* or *L*} [*sup*: 0.61; *con*: 0.99; *lif*: 1.0; *acc*: 0.61].

R_1 shows that the degree of presence of the price and demand in New South Wales (features *NSWPrice* and *NSWDemand*, respectively) are key factors for the electricity transfer between the two states.

The second rule describes the feature *NSWPrice*, which refers to the electricity price in New South Wales.

R$_2$: **IF** *NSWDemand* **is** {*S* or *M* or *L*} **and** *VICPrice* **is** *XS* **and** *transfer* **is** {*XS* or *S* or *M*} **THEN** *NSWPrice* **is** {*XS* or *S* or *M*} [*sup*: 0.78; *con*: 1.0; *lif*: 1.0; *acc*: 0.78].

R_2 indicates that the price of electricity in New South Wales not only depends on its demand, but on the price of electricity in Victoria and in the degree of transfer between these two states. The results obtained by Fuzzy-CSar are very competitive in terms of rule quality (support, confidence, lift and accuracy). This experiment supports the fact that Fuzzy-CSar can be applied to real environments to obtain information that is potentially useful to experts in the field.

6. Experiments on real-world data sets with static concepts

Fuzzy-CSar has demonstrated a competitive behavior under synthetic online environments, showing adaptivity to the challenges of learning under association stream and in a real-world data stream problem. In this section, Fuzzy-CSar's behavior is analyzed under a variate set of real-world problems with static concepts to test its robustness. Despite the fact that Fuzzy-CSar is an online algorithm that learns from streams of data, it can handle offline problems. In what follows, the study of the behavior of Fuzzy-CSar is extended using a large collection of real-world problems that do not contain any kind of concept drift (i.e., concepts do not change over time). The goal of the analysis is twofold: (1) demonstrate the computational complexity and scalability of Fuzzy-CSar, and (2) show the quality of the models created by Fuzzy-CSar, comparing the results with those of Fuzzy-Apriori, the most well-known traditional fuzzy association rule mining algorithm. It is important to note that we compare Fuzzy-CSar models against Fuzzy-Apriori ones to check the behavior of our proposal with the results of a well-known algorithm under variables that are *continuous*. Recall that, by construction, Fuzzy-Apriori generates models that have lower support and confidence, so it is not our primary intention to perform a *classic comparison*, but to highlight how Fuzzy-CSar handles these kinds of problems.

Table 8
Properties of the data sets considered for the study. Columns describe: the identifier (Id.), the number of instances (#Inst), the number of features (#Fe), the number of continuous features (#Re) and the number of integer features (#In).

Id.	#Inst	#Fe	#Re	#In	Id.	#Inst	#Fe	#Re	#In
app	106	7	7	0	*seg*	2310	19	19	0
bpa	345	6	1	5	*son*	208	60	60	0
fam	63 756	10	0	10	*spa*	4597	57	57	0
gls	214	9	9	0	*thy*	215	5	4	1
h-s	270	13	1	12	*veh*	846	18	0	18
irs	150	4	4	0	*wav*	5000	40	40	0
mag	19 020	10	10	0	*wdbc*	569	30	30	0
μca	216	21	21	0	*wne*	178	13	13	0
pho	5404	5	4	1	*wpbc*	198	33	33	0
pim	768	8	8	0	*yst*	1484	8	8	0
rng	7400	20	20	0					

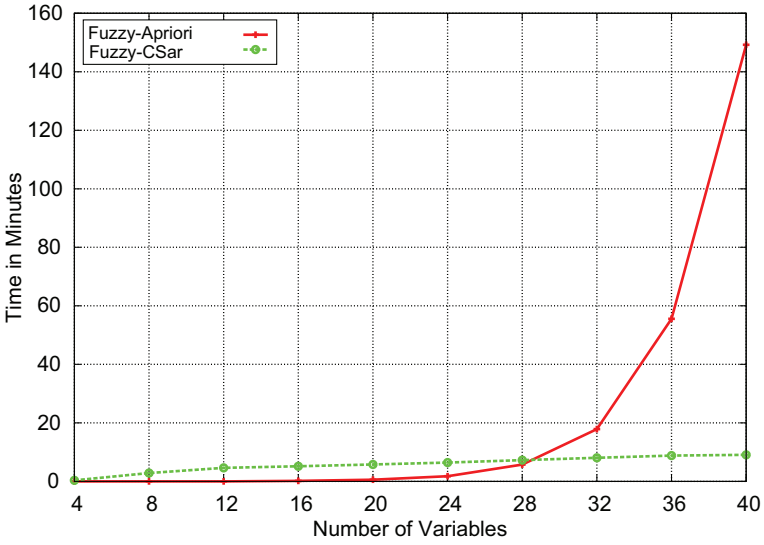

Fig. 8. Relationship between the runtime (in minutes) and the number of variables with 100 000 samples in Fuzzy-CSar.

The 21 selected problems, which show different characteristics, have been taken from the KEEL public repository [4], except for the *fam* data set which was obtained from the UCLA Statistics Data Sets Archive website[1] and *μca*, which is from a local repository. Table 8 shows the properties of the real-world data sets considered for the experimental study.

The aforementioned data sets were intended for classification tasks and hence these have been adapted by removing the class feature. All the experiments were run using a computer with 2.8 GHz Core-i5 CPUs and 4 GB of RAM running Linux. Both Fuzzy-CSar and Fuzzy-Apriori were programmed in C++. Fuzzy-CSar was configured with a population size of 6400 individuals and was run for 100 000 data samples. The remaining parameters were configured as follows: $\nu = 1$, $P_{\#} = 0.5$, $P_{\chi} = 0.8$, $\{P_{!/R}, P_{\mu}, P_C\} = 0.1$, $\delta = 0.1$, $\theta_{\text{GA}} = 50$, $\theta_{exp} = 1000$, $\{\theta_{del}, \theta_{sub}\} = 20$, $maxLingTermsPerVariable = 3$, and $\theta_{mna} = 2$. Fuzzy-Apriori was configured using standard values [26]: $\{minSupport, minConfidence\} = 0.05$. All the variables use Ruspini's strong fuzzy semantics with five linguistic terms (*XS*, *S*, *M*, *L* and *XL*). The experiments have been repeated for 30 runs, each with a different random seed. Results are the averages of these runs.

6.1. Analysis of the computational complexity and scalability

To check the computational complexity and scalability of Fuzzy-CSar, a series of experiments have been performed using the *wav* data set. This data set consists of 5000 samples and has 40 features. This experiment has been conducted to analyze the complexity and scalability of Fuzzy-CSar: analysis of the relationship between the runtime and the number of variables when using all the 5000 transactions of the data set using 100 000 data samples, i.e., each sample is shown 20 times to the algorithm.

The exponential nature of Fuzzy-Apriori is clearly visible (Fig. 8), specially when using 28 (and more) variables its running time starts increasing abruptly. On the other hand, Fuzzy-CSar is not dramatically affected by the number of variables,

[1] http://www.stat.ucla.edu/data/fpp.

demonstrating a high scalability. Notice that due to the learning scheme costs of LCSs (generating the population, the match set, the association set and the GA episode), it turns out that Fuzzy-CSar works best with large data sets and it is outperformed by simpler algorithms such as Fuzzy-Apriori in low dimensionality ones. However, recall that the nature of the problems handled by these two algorithms differs as Fuzzy-CSar is designed for handling association streams.

The experiments support Fuzzy-CSar having a high-grade of scalability in regard to the number of variables, which is one of the main issues in data streams, as these online learning algorithms scale properly on the number of samples but have difficulties when dealing with great number of variables. These kinds of algorithms can be stopped whenever required and the evolved rule set can be used for modeling the environment. In this regard, the more learning iterations Fuzzy-CSar has performed, the more general and accurate rules (in terms of support, confidence, lift and accuracy) should be.

6.2. Analysis of the quality of the models

To show the quality of the models generated by Fuzzy-CSar, we compared the results with those of Fuzzy-Apriori in terms of (1) number of rules, (2) average number of variables obtained in the antecedent part of the rules, (3) support, (4) confidence, and (5) average time required for obtaining such results. In order to make a fair comparison, the output rules of Fuzzy-CSar and Fuzzy-Apriori have been filtered allowing only high quality rules (i.e., whose confidence is greater than 0.75).

Table 9 shows the results obtained by both algorithms. It is clearly noticeable that Fuzzy-CSar outperforms Fuzzy-Apriori in terms of support and confidence. It is worth mentioning the overwhelming number of rules obtained by Fuzzy-Apriori in some of the problems, a product of the generate-and-test procedure: this family of techniques suffer from an exponential behavior. Notice that the average time required for solving such problems is also huge in the case of Fuzzy-Apriori. Additionally, Fuzzy-Apriori is not able to finish the *spa* problem. In comparison Fuzzy-CSar evolves a much more balanced population of rules in terms of support and confidence, and also the average time required for obtaining such solutions is much lower.

A statistical analysis has been conducted to evaluate the significance of the results following the recommendations given by Demšar [19]. More specifically, we compared the support and the confidence levels of both Fuzzy-CSar and Fuzzy-Apriori by means of the non-parametric Wilcoxon signed-ranks test. The approximate p-values resulting from these pairwise comparisons are provided in the analysis. The results obtained by the statistical tests at $\alpha = 0.05$ are summarized in the following: in the case of the support value we obtain $z = -4.014$, hence rejecting the hypothesis that both algorithms perform the same on average with a computed p-value of 5.956×10^{-5}. In the case of the confidence level we obtain $z = -3.945$, hence rejecting the hypothesis that both algorithms perform the same on average with a computed p-value of 1.204×10^{-4}. In this regard, Fuzzy-CSar outperforms Fuzzy-Apriori in both support and confidence.

6.3. A duel with FP-Growth

Authors acknowledge that Fuzzy-Apriori is known not to perform while dealing with high dimensional problems. However, we decided to use it because (1) it is well known and thus it is well suited as a baseline algorithm that allows us to benchmark our method, (2) it is able to handle continuous variables, and (3) the source code is available. Recall that this comparison is made in offline scenarios just to show that our proposal is able to work in classic offline environments giving competitive results.

As we could not find the source code for Fuzzy-FP-Growth [6], authors decided to perform a prospective experiment using the classic FP-Growth in a synthetic environment to check the behavior of both FP-Growth and Fuzzy-CSar. This environment used for the test is similar to the one used in Section 4: it allows for the generation of data from a set of continuous rules, each composed of 40 real-valued variables. There are three *master rules* used for generating the data, as shown in Table 10.

Therefore, variables V_1 to V_{39} are always in the antecedent part of the master rules, and variable V_{40} is always in the consequent part of the master rules. Variables take values following a uniform distribution. Notice that the intervals remain constant for all the variables in a master rule. This is a straightforward way to generate data and allows the inspection of the results obtained by the different algorithms analyzed.

We generated 100 000 instances with the aforesaid master rules, that is: 1/3 of the instances by the generator $Rule_1$, 1/3 of the instances by the generator $Rule_2$ and finally, the remaining 1/3 of the instances by the generator $Rule_3$. After generating this data and storing it in a text file, we discretized the data using 10 different bins using WEKA, so FP-Growth can handle the problem.

The configuration used for Fuzzy-CSar is the same as above in this section except for the population size, which was reduced to 1000 individuals, and θ_{GA}, which was set to 100. FP-Growth was configured with $minsupport = 0.05$ and $minconfidence = 0.1$. The experiment has been repeated 10 times and the results are the averages of these runs.

As the results show in Table 11, FP-Growth is a very fast algorithm that is able to extract frequent itemsets from large amounts of data. However, it needs the classic two-step process in order to obtain the desired rules, so it does not fit when processing association streams. Also, it is not designed for tackling concept drifts, where the dynamics of the data being learnt may change unexpectedly.

With respect to Fuzzy-CSar, although being slower than FP-Growth at obtaining the solution, the rules evolved were of higher quality: this is unsurprising since Fuzzy-CSar optimizes for best rule quality (in terms of lift, confidence and accuracy), whereas FP-Growth only accounts for minimum support. However, what was surprising was the fact that FP-Growth rules use lesser variables per rule, whereas Fuzzy-CSar shows much richer rules in terms of variable usage.

Table 9
Comparisons of results between Fuzzy-CSar and Fuzzy-Apriori. Columns describe: the identifier of the data set (Id.), the average number of rules (#*R*), the average number of variables in the antecedent (#AV), the average support (Sup), the average confidence (Con), and the average time in seconds (*T*). These values are extracted from rules with a confidence level ≥ 0.75.

	Fuzzy-CSar results					Fuzzy-Apriori results				
Id.	#*R*	#AV	Sup	Con	*T*	#*R*	#AV	Sup	Con	*T*
app	661.200 ± 27.491	3.387 ± 1.816	0.534 ± 0.054	0.910 ± 0.002	99.377 ± 2.782	713	2.804 ± 0.682	0.086 ± 0.002	0.787 ± 0.002	0.230 ± 0.030
bpa	308.700 ± 17.595	2.846 ± 1.619	0.728 ± 0.052	0.919 ± 0.002	49.036 ± 3.145	614	2.651 ± 0.523	0.093 ± 0.003	0.748 ± 0.003	0.220 ± 0.040
fam	1397.900 ± 40.852	4.548 ± 2.920	0.679 ± 0.049	0.914 ± 0.003	237.626 ± 4.733	7441	3.718 ± 1.183	0.094 ± 0.003	0.872 ± 0.005	71.563 ± 0.501
gls	1176.100 ± 37.095	4.186 ± 2.738	0.678 ± 0.033	0.914 ± 0.002	186.504 ± 5.601	4608	3.863 ± 1.443	0.105 ± 0.004	0.865 ± 0.005	0.181 ± 0.000
h-s	1769.400 ± 62.764	4.640 ± 3.191	0.568 ± 0.042	0.896 ± 0.003	302.307 ± 7.623	17 026	3.926 ± 1.197	0.080 ± 0.002	0.871 ± 0.005	5.204 ± 0.018
irs	165.900 ± 12.501	2.026 ± 0.548	0.402 ± 0.022	0.906 ± 0.003	16.964 ± 1.267	89	1.876 ± 0.288	0.076 ± 0.001	0.729 ± 0.003	0.230 ± 0.170
mag	1706.600 ± 25.192	4.754 ± 2.936	0.594 ± 0.041	0.913 ± 0.002	330.333 ± 8.890	7164	3.358 ± 0.753	0.082 ± 0.002	0.796 ± 0.003	14.464 ± 0.010
μca	2039.500 ± 35.183	4.351 ± 2.929	0.382 ± 0.036	0.923 ± 0.003	453.461 ± 8.143	7 162 405	8.151 ± 1.667	0.061 ± 0.010	0.930 ± 0.025	51520.736 ± 2328.668
pho	196.100 ± 15.788	2.435 ± 1.095	0.678 ± 0.036	0.895 ± 0.003	31.530 ± 2.182	183	1.945 ± 0.227	0.088 ± 0.002	0.693 ± 0.003	0.061 ± 0.000
pim	786.400 ± 25.227	3.872 ± 2.308	0.696 ± 0.032	0.915 ± 0.001	149.680 ± 3.372	724	2.898 ± 0.451	0.077 ± 0.001	0.807 ± 0.003	0.102 ± 0.000
rng	2131.100 ± 86.244	5.474 ± 6.149	0.486 ± 0.055	0.920 ± 0.003	417.035 ± 20.010	1 727 145	5.753 ± 1.412	0.075 ± 0.001	0.848 ± 0.001	7854.716 ± 234.547
seg	1854.800 ± 67.865	4.273 ± 2.758	0.439 ± 0.046	0.913 ± 0.003	363.935 ± 11.729	596 772	6.214 ± 3.468	0.083 ± 0.001	0.911 ± 0.006	896.718 ± 22.663
son	2030.000 ± 55.596	3.709 ± 2.396	0.399 ± 0.045	0.898 ± 0.003	839.944 ± 20.936	1 941 347	3.250 ± 0.295	0.061 ± 0.000	0.779 ± 0.001	40737.684 ± 2958.617
spa	2207.300 ± 59.057	20.084 ± 9.903	0.642 ± 0.014	0.983 ± 0.000	952.786 ± 38.115	–	–	–	–	–
thy	209.100 ± 29.063	2.317 ± 1.049	0.670 ± 0.086	0.935 ± 0.002	23.307 ± 3.291	182	2.440 ± 0.730	0.138 ± 0.011	0.826 ± 0.005	0.080 ± 0.020
veh	2098.300 ± 44.445	4.466 ± 2.650	0.411 ± 0.038	0.914 ± 0.003	363.354 ± 10.156	40 779	3.408 ± 0.780	0.068 ± 0.001	0.799 ± 0.003	19.065 ± 1.008
wav	2132.300 ± 41.075	4.489 ± 4.404	0.415 ± 0.054	0.891 ± 0.005	545.313 ± 8.845	1 262 758	3.372 ± 0.457	0.066 ± 0.000	0.761 ± 0.002	9417.744 ± 621.681
wdbc	2049.800 ± 49.859	4.800 ± 5.074	0.458 ± 0.059	0.929 ± 0.003	455.763 ± 13.764	4 273 614	4.757 ± 0.908	0.064 ± 0.000	0.823 ± 0.002	60643.766 ± 3704.684
wne	2146.100 ± 27.012	5.016 ± 3.015	0.359 ± 0.030	0.896 ± 0.002	336.199 ± 4.969	4292	2.440 ± 0.324	0.069 ± 0.000	0.728 ± 0.002	0.291 ± 0.006
wpbc	2149.600 ± 23.711	4.015 ± 2.577	0.480 ± 0.043	0.917 ± 0.002	497.874 ± 7.220	852 845	3.643 ± 0.404	0.063 ± 0.000	0.796 ± 0.002	5274.364 ± 32.128
yst	608.000 ± 14.669	3.743 ± 2.474	0.815 ± 0.017	0.926 ± 0.002	95.588 ± 4.809	2447	3.631 ± 1.227	0.120 ± 0.010	0.897 ± 0.010	0.307 ± 0.000

Table 10
Environment used to compare Fuzzy-CSar with FP-Growth.

Rule ID	Actual rule
$Rule_1$	$V_1[0, 0.25] \wedge V_2[0, 0.25] \wedge \cdots \wedge V_{39}[0, 0.25] \Rightarrow V_{40}[0, 0.25]$
$Rule_2$	$V_1[0.375, 0.625] \wedge V_2[0.375, 0.625] \wedge \cdots \wedge V_{39}[0.375, 0.625] \Rightarrow V_{40}[0.375, 0.625]$
$Rule_3$	$V_1[0.75, 1] \wedge V_2[0.75, 1] \wedge \cdots \wedge V_{39}[0.75, 1] \Rightarrow V_{40}[0.75, 1]$

Table 11
Results obtained. Columns describe: the average number of rules (#*R*), the average number of variables (#AV), the average support (Sup), the average confidence (Con), and the average time in seconds (*T*).

Method	#*R*	Sup	Con	#AV	*T*
Fuzzy-CSar	92.71 ± 1.0961	0.051 ± 0.0013	0.97 ± 0.0025	24.56 ± 6.5955	220.24 ± 9.1135
FP-Growth	18720	0.0502 ± 0.0035	0.397 ± 0.0142	2.26 ± 2.2451	134.32 ± 2.0212

Table 12
SWOT analysis of Fuzzy-CSar.

Strengths	Weaknesses
- It faces an unexplored problem: association stream mining - It is a fast algorithm that adapts itself to concept changes and that only needs a single pass over the data to obtain the desired rules. - It deals with large data sets. - It outputs rules that are highly legible.	- It may generate a large amount of rules. - It has many parameters that have to be properly configured. - It requires, in general, a large population to capture the desired patterns out of streams of data.
Opportunities	Threats
- It showed a competitive performance which opens opportunities for further research on rule reduction mechanisms and new fuzzy knowledge representations. - Its rule deletion mechanism can be further improved by adding rule-aging mechanisms and an unsupervised concept drift detector. - Association stream mining is a field that remains virtually unexplored but with a high potential to be applied in real-world data stream problems.	- The complex design of the algorithm may discourage practitioners. - Association stream mining is a new field and some researchers may not know how to take advantage of it.

7. SWOT analysis of Fuzzy-CSar

To finish the study of Fuzzy-CSar, a final SWOT analysis is performed as follows, summarized in Table 12, where *strengths* represent the main advantages of Fuzzy-CSar, *weaknesses* show the drawbacks of the algorithm, *opportunities* suggest the further lines of research to enhance the system performance, and *threads* indicate issues that other machine learning approaches may take advantage of.

Fuzzy-CSar has three main strengths: first, it is a fast algorithm that adapts itself to concept drifts and that only needs a single step over the data to obtain the desired fuzzy association rules. Second, Fuzzy-CSar inherits the online architecture and thus it is capable of mining large amounts of data. Third, as the field of association stream mining requires a highly readable output, Fuzzy-CSar exploits its fuzzy representation offering highly legible models to users.

The main weakness of Fuzzy-CSar is that it may generate a large amount of rules, which may hinder the interpretability of the results given by the algorithm. Related to this issue, the proposed algorithm has many parameters that have to be properly configured in order to obtain the desired output. Besides, Fuzzy-CSar requires a moderate population to capture the desired patterns.

A possible threat is that, although being competent in both dynamic and static environments, the proposed algorithm is a complex system and therefore practitioners have to know how to configure it correctly. This issue may discourage newcomers to explore the possibilities of Fuzzy-CSar. Furthermore, as the field of association stream mining is very recent, some researchers may not know how to take advantage of it.

Fuzzy-CSar shows some interesting opportunities that are being considered as future work: (1) the inclusion of rule reduction mechanisms without significant loss of model quality, (2) a more detailed research of other fuzzy knowledge representations, checking the tradeoff between flexibility and interpretability, (3) a new rule deletion mechanism that contains rule-aging and an unsupervised concept drift detector to improve the responsiveness of the system, and (4) application in real-world unsupervised data stream problems.

8. Summary and conclusions

In this paper we have addressed the problem of knowledge discovery from streams of unlabelled real-world data with continuous features from the particular point of monitoring the system by continuously evolving association rules, which we call association stream mining. The main features of this novel field are as follows: (1) it consists of continuous flows of data that have to be processed in a single step, (2) there are restrictions in memory usage, (3) the concept to be learnt may drift over time, and (4) it does not assume any underlying structure nor fixed data distribution. Also, a high degree of interpretability is desirable. It is relevant to highlight that association stream mining differs from frequent pattern mining in that frequent pattern mining is focused on the identification of frequent variables, relegating the generation of the final rules to second place using a classic offline approach, which make it mostly ill-suited for our purposes.

The challenges of association stream mining have been addressed by introducing Fuzzy-CSar, a Michigan-style Learning Fuzzy-Classifier System for extracting association rules. Differing from the classic association mining strategies, the proposed technique performs a single pass over data—without any windowing mechanism—to obtain the output rules by defining a maximum number of rules to undercover. Our approach handles problems that deliver a continuous flow of unlabelled data, which is usual in many industrial and scientific applications (e.g., Smart Grids) and where the data samples provided to the system have to be processed on the fly in order to be able to mine useful information.

An extensive analysis has been performed in order to check the behavior of the proposed algorithm under association stream problems. First by making use of synthetic environments, each one with a different complexity, and following that, with a real data stream problem with unknown dynamics.

Fuzzy-CSar's discovery mechanisms (i.e., covering and the genetic events) allow the system to perform competitively under the harsh conditions of association streams. Whenever a concept drift happens, new rules are generated (and old ones deleted) by means of these mechanisms, making Fuzzy-CSar adaptive to dynamic environments.

The experimentation has been extended by carefully analyzing the complexity and scalability of Fuzzy-CSar and the quality of the models evolved by the algorithm. The incremental nature of the presented algorithm is very noticeable when compared to a standard association rule mining method such as Fuzzy-Apriori. The relation between runtime and number of variables and the relation between runtime and number of transactions has shown an outstanding behavior. Finally, the analysis of the quality of the models under real-world data sets with static concepts has shown that Fuzzy-CSar is very competitive in terms of both support and confidence. Due to the learning scheme costs of Fuzzy-CSar (i.e., generating the population, the match set, the association set and the GA episode), it works best with large data sets, and it is outperformed by simpler algorithms such as Apriori in low dimensionality ones.

Acknowledgment

This paper has been supported by the Spanish Ministry of Economy and Competitiveness under grant no. TIN2014-57251-P. The authors would like to thank Ian Kavanagh and Rebecca Cray for their support in proofreading the manuscript.

References

[1] R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, in: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, ACM, Washington, D.C., USA, 1993, pp. 207–216.
[2] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of the 20th International Conference on Very Large Data Bases, BLDB, Santiago, Chile, 1994, pp. 487–499.
[3] R. Akbarinia, F. Masseglia, Fast and exact mining of probabilistic data streams, in: H. Blockeel, K. Kersting, S. Nijssen, F. Železný (Eds.), Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, 8188, Springer, Berlin, Heidelberg, 2013, pp. 493–508.
[4] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, J. Multiple-Valued Logic Soft Comput. 17 (2011) 255–287.
[5] P. Angelov, Autonomous Learning Systems: From Data Streams to Knowledge in Real-Time, first ed., Wiley, 2012.
[6] M. Antonelli, P. Ducange, F. Marcelloni, A. Segatori, A novel associative classification model based on a fuzzy frequent pattern mining algorithm, Expert Syst. Appl. 42 (2015) 2086–2097.
[7] A. Bifet, G. Holmes, B. Pfahringer, R. Gavaldà, Mining frequent closed graphs on evolving data streams, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, USA, 2011, pp. 591–599.
[8] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, T. Seidl, MOA: massive online analysis, a framework for stream classification and clustering, J. Mach. Learn. Res. 11 (2010) 44–50.
[9] A. Bouchachia, E. Lughofer, D. Sánchez, Online fuzzy machine learning and data mining, Inf. Sci. 220 (2013) 1–4.
[10] M. Butz, Rule-Based Evolutionary Online Learning Systems – A Principled Approach to LCS Analysis and Design, first ed., Springer, 2006.
[11] J. Casillas, F. Martínez-López, Mining uncertain data with multiobjective genetic fuzzy systems to be applied in consumer behaviour modelling, Expert Syst. Appl. 36 (2009) 1645–1659.
[12] B. Chandra, S. Bhaskar, A novel approach of finding frequent itemsets in high speed data streams, in: 8th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD, 2011, pp. 40–44.
[13] J. Chang, W. Lee, Finding recent frequent itemsets adaptively over online data streams, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, USA, 2003, pp. 487–492.
[14] J. Chang, W. Lee, Effect of count estimation in finding frequent itemsets over online transactional data streams, J. Comput. Sci. Technol. 20 (2005) 63–69.
[15] P. Chen, H. Su, L. Guo, Y. Qu, Mining fuzzy association rules in data streams, in: 2nd International Conference on Computer Engineering and Technology, ICCET, 2010, pp. 153–158.
[16] J. Cheng, Y. Ke, W. Ng, Maintaining frequent closed itemsets over a sliding window, J. Intell. Inf. Syst. 31 (2008) 191–215.
[17] G. Cormode, F. Korn, S. Muthukrishnan, D. Srivastava, Finding hierarchical heavy hitters in streaming data, ACM Trans. Knowl. Discovery Data 1 (2008) 2:1–2:48.
[18] W. Pedrycz, Granular Computing: Analysis and Design of Intelligent Systems, first ed., CRC Press/Francis Taylor, Boca Raton, 2013.

[19] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[20] D. Dubois, E. Hüllermeier, H. Prade, A systematic approach to the assessment of fuzzy association rules, Data Min. Knowl. Discovery 13 (2006) 167–192.
[21] W. Fan, T. Watanabe, K. Asakura, Ratio rules mining in concept drifting data streams, in: Proceedings of the World Congress on Engineering and Computer Science, WCECS '09, San Francisco, USA, 2009, pp. 809–814.
[22] Z. Farzanyar, M. Kangavari, N. Cercone, Max-FISM: mining (recently) maximal frequent itemsets over data streams using the sliding window model, Comput. Math. Appl. 64 (2012) 1706–1718.
[23] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: a frequent-pattern tree approach, Data Min. Knowl. Discovery 8 (2004) 53–87.
[24] M.B. Harries, C. Sammut, K. Horn, Extracting hidden context, Mach. Learn. 32 (1998) 101–126.
[25] C. HewaNadungodage, Y. Xia, J. Lee, Y. Tu, Hyper-structure mining of frequent patterns in uncertain data streams, Knowl. Inf. Syst. 37 (2013) 219–244.
[26] T. Hong, C. Kuo, S. Chi, Trade-off between computation time and number of rules for fuzzy mining from quantitative data, Int. J. Uncertainty Fuzziness Knowledge-Based Syst. 9 (2001) 587–604.
[27] D. Huang, Y. Koh, G. Dobbie, Rare pattern mining on data streams, in: A. Cuzzocrea, U. Dayal (Eds.), Data Warehousing and Knowledge Discovery, Lecture Notes in Computer Science, 7448, Springer, Berlin, Heidelberg, 2012, pp. 303–314.
[28] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: 2001 ACM International Conference on Knowledge Discovery and Data Mining, 2001, pp. 97–106.
[29] R. Karp, S. Shenker, C. Papadimitriou, A simple algorithm for finding frequent elements in streams and bags, ACM Trans. Database Syst. 28 (2003) 51–55.
[30] C. Leung, H. Boyu, Mining of frequent itemsets from streams of uncertain data, in: EEE 25th International Conference on Data Engineering, ICDE '09, 2009, pp. 1663–1670.
[31] A. Marascu, F. Masseglia, Mining sequential patterns from data streams: a centroid approach, J. Intell. Inf. Syst. 27 (2006) 291–307.
[32] M. Martínez-Ballesteros, F. Martínez-Álvarez, A. Troncoso, J. Riquelme, An evolutionary algorithm to discover quantitative association rules in multidimensional time series, Soft Comput. 15 (2011) 2065–2084.
[33] M. Memar, M. Deypir, M. Sadreddini, S. Fakhrahmad, A block-based approach for frequent itemset mining over data streams, in: 8th International Conference on Fuzzy Systems and Knowledge Discovery, 2011, pp. 1647–1651.
[34] M. Núñez, R. Fidalgo, R. Morales, Learning in environments with unknown dynamics: towards more robust concept learners, J. Mach. Learn. Res. 8 (2007) 2595–2628.
[35] A. Orriols-Puig, New challenges in learning classifier systems: mining rarities and evolving fuzzy models, Arquitectura i Enginyeria La Salle, Universitat Ramon Llull, Barcelona, 2008 Ph.D. thesis.
[36] A. Orriols-Puig, J. Casillas, F. Martínez-López, Unsupervised learning of fuzzy association rules for consumer behavior modeling, Mathware Soft Comput. 16 (2009) 29–43.
[37] A. Orriols-Puig, J. Casillas, E. Bernadó-Mansilla, Fuzzy-UCS: a Michigan-style learning fuzzy-classifier system for supervised learning, IEEE Trans. Evol. Comput. 13 (2009) 260–283.
[38] A. Orriols-Puig, J. Casillas, Evolution of interesting association rules online with learning classifier systems, in: Lecture Notes in Computer Science, 6471, 2010, pp. 21–37.
[39] A. Orriols-Puig, J. Casillas, Fuzzy knowledge representation study for incremental learning in data streams and classification problems, Soft Comput. 15 (2011) 2389–2414.
[40] C. Raïssi, P. Poncelet, Sampling for sequential pattern mining: from static databases to data streams, in: 7th IEEE International Conference on Data Mining, 2007, pp. 631–636.
[41] X. Shi, W. Fan, P.S. Yu, Efficient semi-supervised spectral co-clustering with constraints, in: 10th IEEE International Conference on Data Mining, 2010, pp. 1043–1048.
[42] Q. Tu, J. Lu, J. bin Tang, J. Yang, The FP-tree algorithm used for data stream, in: Chinese Conference on Pattern Recognition, 2010, pp. 1–5.
[43] E. Wang, A. Chen, Mining frequent itemsets over distributed data streams by continuously maintaining a global synopsis, Data Min. Knowl. Discovery 23 (2011) 252–299.
[44] K. Wang, S. Hock, W. Tay, B. Liu, Interestingness-based interval merger for numeric association rules, in: 4th International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1998, pp. 121–127.
[45] S. Wilson, Classifier fitness based on accuracy, Evol. Comput. 3 (1995) 149–175.
[46] R. Wong, A. Fu, Mining top-k frequent itemsets from data streams, Data Min. Knowl. Discovery 13 (2006) 193–217.
[47] H. Yang, H. Liu, J. He, Delay: a lazy approach for mining frequent patterns over high speed data streams, in: R. Alhajj, H. Gao, J. Li, X. Li, O. Zaïane (Eds.), Advanced Data Mining and Applications, Lecture Notes in Computer Science, 4632, Springer, Berlin, Heidelberg, 2007, pp. 2–14.
[48] S. Yen, Y.S. Lee, C. Wu, C. Lin, An efficient algorithm for maintaining frequent closed itemsets over data stream, in: B. Chien, T. Hong, S. Chen, M. Ali (Eds.), Next-Generation Applied Intelligence, 5579, Springer, Berlin, Heidelberg, 2009, pp. 767–776. Lecture Notes in Computer Science.
[49] M. Zihayat, A. An, Mining top-k high utility patterns over data streams, Inf. Sci. 285 (2014) 138–161.