Evolution of Interesting Association Rules Online with Learning Classifier Systems

Albert Orriols-Puig[1] and Jorge Casillas[2]

[1] Grup de Recerca en Sistemes Intel·ligents
La Salle - Universitat Ramon Llull
Quatre Camins 2, 08022 Barcelona (Spain)
aorriols@salle.url.edu
[2] Dept. Computer Science and Artificial Intelligence
University of Granada
18071, Granada, Spain
casillas@ugr.es

Abstract. This paper presents CSar, a Michigan-style learning classifier system designed to extract quantitative association rules from streams of unlabeled examples. The main novelty of CSar with respect to the existing association rule miners is that it evolves the knowledge online and it is thus prepared to adapt its knowledge to changes in the variable associations hidden in the stream of unlabeled data quickly and efficiently. The results provided in this paper show that CSar is able to evolve interesting rules on problems that consist of both categorical and continuous attributes. Moreover, the comparison of CSar with Apriori on a problem that consists only of categorical attributes highlights the competitiveness of CSar with respect to more specific learners that perform enumeration to return all possible association rules. These promising results encourage us to further investigate on CSar.

1 Introduction

Association rule mining [2] aims at extracting interesting associations among the attributes—i.e., associations that occur with a certain frequency and strength—of repositories of unlabeled data. Research conducted on association rule mining was originally focused on extracting rules that identified strong relationships between the occurrence of two or more attributes or *items* on collections of binary data, e.g., "*if item X occurs then also item Y will occur*" [2,3,14]. Later on, several researchers concentrated on extracting association rules from data described by continuous attributes [10,22], which posed new challenges to the field. Several algorithms proposed to apply a discretization method in advance to transform the original data into binary values [16,18,22,24] and then use a binary association rule miner. This led to further research on designing discretization procedures that avoid losing useful information. Other approaches mined interval-based association rules and permitted the algorithm to independently move the interval

J. Bacardit et al. (Eds.): IWLCS 2008/2009, LNAI 6471, pp. 21–37, 2010.

bound of each rule's variable [17]. Also, fuzzy modeling was introduced to create fuzzy association rules (e.g., see [13,15]).

Association rules are widely used in various areas such as telecommunication networks, market and risk management, and inventory control. All these applications are characterized by generating data online, so that data may be made available in form of streams [1,19]. Nonetheless, all the aforementioned algorithms were designed for static collections of data. Learning from data streams has received a special amount of attention in the last few years, particularly in supervised learning [1,19]. However, few proposals of online binary association rule miners can be found in the literature, and most of them are only able to deal with problems with categorical attributes (e.g., see [23]).

In this paper, we address the problem of mining association rules from streams of examples online. We propose a *learning classifier system* (LCS) whose architecture is inspired by XCS [25,26] and UCS [6], which we address as *classifier system for association rule mining* (CSar). CSar uses an interval-based representation for evolving quantitative association rules from data with continuous attributes and a discrete representation for categorical attributes. The system receives a stream of unlabeled examples which are used to create new rules and to tune the parameters of the existing ones with the aim of evolving as many interesting rules as possible. CSar is first compared with Apriori [3] on a problem defined only by categorical attributes. The results on this problem indicate that CSar can evolve rules of similar interest as those created by Apriori, one of the most referred algorithms in the association rule mining realm, which considers all the possible combinations of attribute values to create all interesting association rules (notice that this approach can only be used in domains with categorical data). The experimentation is then extended by considering a collection of real-world problems and by analyzing the behavior of different configurations of CSar over these problems. The results denote that CSar is able to create highly supported and interesting interval-based association rules in which the intervals have not been prefixed by a discretization algorithm.

The remainder of this paper is organized as follows. Section 2 provides the basic concepts of association rules and reviews the main proposals in the literature for both binary and quantitative association rule mining. Section 3 describes in detail our proposal. Section 4 explains the methodology followed in the experiments, and Section 5 analyzes the results of these experiments. Finally, Section 6 summarizes, concludes, and gives the future work lines that will be followed.

2 Framework

Before proceeding with the description of our proposal, this section introduces some important concepts of association rules. We first describe the problem of extracting association rules from categorical data. Then, we extend the problem to mining association rules from data with continuous attributes and review different proposals that can be found in the literature.

2.1 Association Rule Mining

The problem of association rule mining was firstly defined over binary data in [2] as follows. Let $I = \{i_1, i_2, \ldots, i_\ell\}$ be a set of binary attributes called *items*. Let T be a set of transactions, where each transaction t is represented as a binary vector of length ℓ. Each position i of t indicates whether the item i is present ($t_i = 1$) or not ($t_i = 0$) in the transaction. X is an *itemset* if $X \subseteq I$. An itemset X has a support $supp(X)$ which is computed as

$$supp(X) = \frac{|X(T)|}{|T|}, \tag{1}$$

That is, the support is the number of transactions in the database which have the itemset X, X(T), divided by the total number of transactions in the database, $|T|$. An itemset is said to be a *frequent itemset* if its support is greater than a user-set threshold, typically addressed as $minsupp$ in the literature.

Then, an *association rule* R is an implication of the form $X \rightarrow Y$, where both X and Y are itemsets and $X \cap Y = \emptyset$. Typically, association rules are assessed with two qualitative measures, their support ($supp$) and their confidence ($conf$). The support of a rule is defined as ratio of the support of the union of antecedent and consequent to the number of transactions in the database, i.e.,

$$supp(R) = \frac{supp(X \cup Y)}{|T|}. \tag{2}$$

The confidence is computed as the ratio of the support of the union of antecedent and consequent to the support of the antecedent, i.e.,

$$conf(R) = \frac{supp(X \cup Y)}{supp(X)}. \tag{3}$$

Therefore, support indicates the frequency of occurring patterns, and confidence evaluates the strength of the implication denoted in the association rule.

Since the proposal of AIS [2], the first algorithm to mine association rules from categorical examples, several algorithms have been designed to perform this task. Agrawal et al. [3] presented the Apriori algorithm, probably the most influential categorical association rule miner. This work resulted in several papers which designed some modifications to the initial Apriori algorithm (e.g., see [8,21]). All these algorithms used the same methodology as Apriori to mine association rules, which basically consisted of two different phases: (1) identification of all frequent itemsets (i.e., all itemsets whose support was greater than $minsupp$), and (2) generation of association rules from these frequent itemsets.

2.2 Quantitative Association Rules

Early research in the realm of association rules only addressed the problem of extracting association rules from binary data. Therefore, these types of rules only permitted reflecting whether particular items were present in the transaction, but they did not consider their quantities. Later on, researchers focused

on algorithms that were able to extract association rules from databases that contained quantitative attributes.

Srikant and Agrawal [22] designed an Apriori-like approach to mine *quantitative association rules*. The authors used an equi-depth partitioning to transform continuous attributes to categorical attributes. Moreover, the authors identified the problem of the *sharp boundary* between discrete intervals, which highlighted that quantitative mining algorithms may either ignore or over-emphasize the items that lay near the boundary of intervals. Attempting to address this problem, several authors applied different clustering mechanisms to extract the best possible intervals from the data [16,18]. A completely different approach was taken in [17], where a genetic-algorithm-based technique was used to evolve interval-based association rules without applying any discretization procedure to the variables. The GA was responsible for creating new promising association rules and for evolving the intervals of the variables of the association rules. The problem associated to creating variables with unbounded intervals is that, in general, the support for small intervals is smaller than the support for large intervals, which makes the system create rules with large intervals, covering nearly all the domain. To avoid this, the system penalized the fitness of rules that had large intervals. In [20] a similar approach was followed. The authors proposed a framework in which finding good intervals from which interesting association rules could be extracted was addressed as an optimization problem.

As done in [17,20], CSar does not apply any discretization mechanism to the original data and interval bounds are evolved by the genetic procedure. The main novelty of our proposal is that association rules are not mined from static databases but from streams of examples. This characteristic guides some parts of the algorithm design, which is described in detail in the next section.

3 Description of CSar

CSar is a Michigan-style LCS for mining interval-based association rules from data that contain both quantitative and categorical attributes. The learning architecture of CSar is inspired by UCS [6] and XCS [25,26]. CSar aims at evolving populations of interesting association rules, i.e., rules with large support and confidence. For this purpose, CSar evaluates a set of association rules online and evolves this rule set by means of a steady-state genetic algorithm (GA) [11,12] that is applied to population niches. As follows, a detailed description of the system is provided, focusing on the differences in the knowledge representation and learning process with respect to those of XCS and UCS.

3.1 Knowledge Representation

CSar evolves a population of classifiers [P], where each classifier consists of a *quantitative association rule* and a set of parameters. The quantitative association rule is represented as

$$\textbf{if } x_i \in v_i \text{ and } \ldots \text{ and } x_j \in v_j \textbf{ then } x_k \in v_k,$$

where the antecedent is represented by a set of ℓ_a input variables $x_i, \ldots, x_j$ ($0 < \ell_a < \ell$, $0 \leq i < \ell$, and $0 \leq j < \ell$; where ℓ is the number of variables of the problem) and the consequent contains a single variable x_k. Note that we permit that rules have an arbitrary number of variables in the antecedent, but we only enable them to have a single variable in the consequent. Restricting the number of consequent variables to one aims at simplifying the creation of niches (see next subsection).

For quantitative attributes, a similar representation to the XCSR one is used [27], in which both antecedent and consequent variables are represented by the interval of values to which this variable applies, i.e., $v_i = [l_i, u_i]$. A maximum interval length $maxInt$ is set to avoid having large intervals that nearly contain all the possible values of a given variable; therefore, $\forall_i : u_i - l_i \leq maxInt$. Categorical attributes are represented by one of the possible categorical values x_{ij}, i.e., $v_i = x_{ij}$. A rule matches an input example if, for all the variables in the antecedent and consequent of the rule, the corresponding value of the example is either included in the interval defined for continuous variables or equal to the value defined for categorical variables.

Each classifier has seven main parameters: (1) the support *supp*, i.e., the occurring frequency of the rule; (2) the confidence *conf*, which indicates the strength of the implication; (3) the fitness *F*, which denotes the quality of the given rule; (4) the experience *exp*, which counts the number of times that the antecedent of the rule has matched an input instance; (5) the consequent matching sum *cm*, which counts the number of times that the whole rule has matched an input instance; (6) the numerosity *num*, which reckons the number of copies of the classifier in the population; and (7) the time of creation of the classifier *tcreate*. The next subsection explains how the classifiers are created and evolved and how their parameters are updated.

3.2 Learning Process Organization

At each learning iteration, CSar receives an input example (e_1, e_2, ..., e_ℓ). Then, the system creates the match set [M] with all the classifiers in the population that match the input example. If [M] contains less that θ_{mna} classifiers, the *covering operator* is triggered to create as many new matching classifiers as required to have θ_{mna} classifiers in [M]. Then, classifiers in [M] are organized in association set candidates following one of the two methodologies explained below. Each association set is given a probability to be selected that is proportional to the average confidence of the classifiers that belong to this association set. The selected association set [A] is checked for subsumption with the aim of diminishing the number of rules that express similar associations among variables. Then, the parameters of all the classifiers in [M] are updated. At the end of the iteration, a GA is applied to the selected association set if the average time since the last application of the GA to the classifiers of the selected association set is greater than θ_{GA} (θ_{GA} is a user-set parameter). Finally, for each continuous attribute, we maintain a list with no repeated elements that stores the last few values seen for the attribute (in our experiments we stored the last hundred different

values). This list is used by the mutation operator with the aim of preventing the existence of intervals that cover the same examples but are slightly different. As follows, we provide details about (1) the covering operator, (2) the procedures to create association set candidates, (3) the association set subsumption mechanism, and (4) the parameter update procedure. Next section explains in more detail the discovery component. It is worth noting that some of the operators are similar to those of several existing systems such as the ones described in [5,9].

Covering Operator. The purpose of the covering operator is to feed classifiers that denote interesting associations among variables into the population. Given the sampled input example e, the covering operator creates a new matching classifier as follows. Each variable is selected with probability $1 - P_{\#}$ to belong to the rule's antecedent, with the restriction that, at the end of this process, at least one variable has to be selected. The values of the selected variables are initialized differently depending on the type of attribute. For categorical attributes, the variable is initialized to the corresponding input value e_i. For continuous attributes, the interval $[l_i, u_i]$ that represents the variable is obtained from generalizing the input value e_i, i.e.,

$$l_i = e_i - rand(maxInt/2) \text{ and} \tag{4}$$

$$u_i = e_i + rand(maxInt/2), \tag{5}$$

where $maxInt$ is the maximum interval length. Finally, one of the previously unselected variables is randomly chosen to form the consequent of the rule, which is initialized following the same procedure. Note that the association rule created is supported by, at least, the sampled example.

Creation of Association Set Candidates. The aim of creating association set candidates or niches is to group rules that express similar associations to establish a competition among them and so let the best ones take over their niche. Whilst the creation of these niches of similar rules is quite immediate in reinforcement learning [25] and classification [6] tasks, several approaches could be used to form groups of similar rules in association rule mining. Herein, we propose two alternatives which are guided by different heuristics:

Grouping by antecedent. This strategy considers that two rules are similar if they have exactly the same variables in their antecedent, regardless of their corresponding values V_i. Therefore, this grouping strategy creates N_a association set candidates, where N_a is the number of rules in [M] with different variables in the antecedent. Each association set contains rules that have exactly the same variables in the antecedent. The underlying idea is that rules with the same antecedent may express similar knowledge. Note that, under this strategy, rules with different variables in the consequent can be grouped in the same association set.

Grouping by consequent. This strategy groups in the same association set the classifiers in [M] that have the same variable in the consequent with

equivalent values. We consider that two continuous variables are equivalent if their intervals are overlapped and that two categorical variables are equivalent if they have the same categorical value. For this purpose, the next process is followed. The rules in [M] are sorted ascendingly according to the variable that they have in their consequent. Given two rules r_1 and r_2 that have the same variable in the consequent, we consider that r_1 is smaller than r_2 if

$$\begin{cases} l_1 < l_2 \text{ or } (l_1 = l_2 \text{ and } u_1 > u_2) & \text{if continuous attribute} \\ ord(x_1) < ord(x_2) & \text{if categorical attribute} \end{cases}$$

where l_1, l_2, u_1, and u_2 are the lower bound and upper bound of the consequent variable of r_1 and r_2 for a continuous attribute, x_1 and x_2 are the values of the consequent variable for a categorical attribute, and $ord(x_i)$ maps each categorical value to a numeric value. It is worth noting that given two continuous variables with the same lower bound in the interval, we sort first the rule with the most general variable (i.e., the rule with larger u_i). We take this approach with the aim of forming association set candidates with the largest number of overlapping classifiers by using the procedure explained as follows.

Once [M] has been sorted, the association set candidates are built as follows. At the beginning, an association set candidate is created and the first classifier in [M] is added to this association set candidate. Then, the following classifier is added if it has the same variable in the consequent, and his lower bound is smaller than the minimum upper bound of the classifiers in the association set. This process is repeated until finding the first classifier that violates this condition. In this case, a new association set candidate is created, and the same process is applied to add new classifiers to this association set. The underlying idea of this association set strategy is that rules that explain the same region of the consequent may denote the same associations among variables.

The cost of both methodologies for creating the association sets are guided by the cost of sorting the population. We applied a quicksort strategy for this purpose, which has a cost of $O(n \cdot logn)$, where n is the match set size.

Association Set Subsumption. A subsumption mechanism inspired by the one presented in [26] was designed with the aim of reducing the number of different rules that express the same or similar knowledge. The process works as follows. Each rule of the selected association set is checked for subsumption with each other rule in the same association set. A rule r_i is a candidate subsumer of r_j if it satisfies the following three conditions: (1) r_i has higher confidence and it is experienced enough (i.e., $conf_i > conf_0$ and $exp_i > \theta_{exp}$, where $conf_0$ and θ_{exp} are user-set parameters); (2) all the variables in the antecedent of r_i are also present in the antecedent of r_j and both rules have the same variable in the consequent (r_j can have more variables in the antecedent than r_i); and (3) 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, each categorical variable of r_i has the same value as the corresponding variable in r_j, and the interval $[l_i, u_i]$ of each continuous variable in r_i includes the interval $[l_j, u_j]$ of the corresponding variable in r_j (i.e., $l_i \leq l_j$ and $u_i \geq u_j$).

Parameter Update. At the end of each learning iteration, the parameters of all the classifiers that belong to the match set are updated. First, we increment the experience of the classifier. Next, we increment the consequent matching estimate *cm* if the rule's consequent also matches the input example. These two parameters are used to update the support and confidence of the rule i as follows. Support is computed as:

$$supp_i = \frac{cm_i}{ctime - tcreate_i}, \tag{6}$$

where *ctime* is the time of the current iteration and $tcreate_i$ is the iteration in which the classifier i has been created. Then, the confidence is computed as

$$conf_i = \frac{cm_i}{exp_i}. \tag{7}$$

Lastly, the fitness of each rule i in [M] is updated with the following formula

$$F_i = (conf_i \cdot supp_i)^{\nu}, \tag{8}$$

where ν is a user-set parameter that permits controlling the pressure toward highly fit classifiers. Note that with this fitness computation, the system makes pressure towards the evolution of rules with not only high confidence but also high support. We empirically tested to compute the fitness only from *conf*, but preliminary experiments indicated that CSar could obtain a larger variety of interesting association rules if support was included in the fitness computation.

Finally, the association set size estimate of all rules that belong to the selected association set is updated. Each rule maintains the average size of all the association sets in which it has participated.

3.3 Discovery Component

CSar uses a steady-state niched GA to discover new promising rules. The GA is applied to the selected association set [A]. Therefore, the niching is intrinsically provided since the GA is applied to rules that are similar according to one of the heuristics for association set formation.

The GA is triggered when the average time from its last application upon the classifiers in [A] exceeds the threshold θ_{GA}. It selects two parents p_1 and p_2 from [A] using proportionate selection [11], where the probability of selecting a classifier k is

$$p^k_{sel} = \frac{F^k}{\sum_{i \in [A]} F^i}. \tag{9}$$

The two parents are copied into offspring ch_1 and ch_2, which undergo crossover and mutation if required.

The system applies uniform crossover with probability P_χ. First, it considers each variable in the antecedent of both rules. If only one parent has the variable, one child is randomly selected and the variable is copied to this child. If both parents contain the variable, this variable is copied to each offspring. The procedure controls that, at the end of the process, each offspring has, at least, one input variable. Then, the rule consequent is crossed by adding to the first offspring the consequent of one of the parents (which is randomly selected) and adding to the remaining offspring the consequent of the other parent.

Three types of mutation can be applied to a rule: (1) introduction/removal of antecedent variables (with probability $P_{I/R}$), (2) mutation of variable's values (with probability P_μ), and (3) mutation of the consequent variable (with probability P_C). The first type of mutation chooses randomly whether a new antecedent variable has to be added to or one of the antecedent variables has to be removed from the rule. If a variable has to be added, one of the non-existing variables is randomly selected and added to the rule. This operation can only be applied if the rule does not have all the possible variables. If a variable has to be removed, one of the existing variables is randomly selected and removed from the rule. This operation can only be applied if the rule has at least two variables in the antecedent. The second type of mutation selects one of the existing variables of the rule and mutates its value. For continuous variables, two random amounts ranging in [-m_0, m_0] are added to the lower bound and the upper bound respectively, where m_0 is a user-set parameter. If the interval surpasses the maximum length or the lower bound becomes greater than the upper bound, the interval is repaired. Finally, the lower and the upper bounds of the mutated variable are approximated to the closest value in the list of seen values for this variable. This process is applied to avoid having rules in the population with very similar interval bounds in its variables, since having all them may not only provide no additional knowledge, but also hinder human experts from reading the whole population. For categorical variables, a new value for the variable is randomly selected. The last type of mutation randomly selects one of the variables in the antecedent and exchanges it with the output variable.

After crossover and mutation, the new offspring are introduced into the population. First, each classifier is checked for subsumption [26] with their parents. To decide if any parent can subsume the offspring, the same procedure explained for association set subsumption is followed. If any parent is identified as a possible subsumer for the offspring, the offspring is not inserted and the numerosity of the parent is increased by one. Otherwise, we check [A] for the most general rule that can subsume the offspring. If no subsumer can be found, the classifier is inserted into the population.

If the population is full, excess classifiers are deleted from [P] with probability proportional to their association set size estimate as. Moreover, if a classifier k is sufficiently experienced ($exp^k > \theta_{del}$) and its fitness F^k is significantly

lower than the average fitness of the classifiers in [P] ($F^k < \delta F_{[P]}$ where $F_{[P]} = \frac{1}{N}\sum_{i\in[P]} F^i$), its deletion probability is further increased. That is, each classifier has a deletion probability p_k of

$$p_k = \frac{d_k}{\sum_{\forall j\in[P]} d_j}, \tag{10}$$

where

$$d_k = \begin{cases} \frac{as\cdot num\cdot F_{[P]}}{F^k} \text{ if } exp^k > \theta_{del} \text{ and } F^k < \delta F_{[P]} \\ as \cdot num \quad \text{otherwise.} \end{cases} \tag{11}$$

Thus, the deletion algorithm balances the classifier allocation in the different association sets by pushing toward the deletion of rules belonging to large correct sets. At the same time, it favors the search toward highly fit classifiers, since the deletion probability of rules whose fitness is much smaller than the average fitness is increased.

3.4 Rule Set Reduction

At the end of the learning process, the final rule set is processed to provide the user with only interesting rules. For this purpose, we apply the following reduction mechanism. Firstly, we remove all rules whose experience is smaller than θ_{exp} (θ_{exp} is a user-set parameter). Then, each rule is checked against each other for subsumption following the same procedure used for association rule subsumption but with the following exception: now, a rule r_i is a candidate subsumer for r_j if r_i and r_j have the same variables in their antecedent and consequent, r_i is more general than r_j, and r_i has higher confidence than r_j. Note that, during learning, the subsumption mechanism requires that the confidence of r_i be greater than $conf_0$.

After applying the rule set reduction mechanism, we make sure that the final population consists of different rules. Other policies can be easily incorporated to this process such as removing rules whose support and confidence are below a predefined threshold. Nonetheless, in our experiments we return all the experienced rules in the final population that are not subsumed by any other.

The overall section has described the mechanisms that CSar uses to evolve a population of interesting association rules online. Differently from other quantitative association-rule miners, CSar is characterized for having a maximum population size that limits the number of different interesting association rules that can exist in the final population. Then, the system organizes rules in different association sets and uses a GA to make rules in the same association set compete. Therefore, CSar does not aim at returning all the possible association rules, but at providing the user with a population of limited size with "phenotypically" different and interesting association rules.

4 Experimental Methodology

After having carefully described the system, now we are in position to experimentally analyze the behavior of CSar. The aim of the experimental analysis

Table 1. Properties of the data sets. The columns describe: the identifier of the data set (Id.); the name of the data set (dataset); the number of instances (#Inst); the total number of features (#Fea); the number of real features (#Re); the number of integer features (#In); and the number of nominal features (#No).

Id.	dataset	#Inst	#Fea	#Re	#In	#No
adl	Adult	48841	15	0	6	9
ann	Annealing	898	39	6	0	33
aud	Audiology	226	70	0	0	70
aut	Automobile	205	26	15	0	11
bpa	Bupa	345	7	6	0	1
col	Horse colic	368	23	7	0	16
gls	Glass	214	10	9	0	1
h-s	Heart-s	270	14	13	0	1
irs	Iris	150	5	4	0	1
let	Letter recognition	20000	17	0	16	1
pim	Pima	768	9	8	0	1
tao	Tao	1888	3	2	0	1
thy	Thyroid	215	6	5	0	1
wdbc	Wisc. diagnose breast-cancer	569	31	30	0	1
wne	Wine	178	14	13	0	1
wpbc	Wisc. prognostic breast-cancer	198	34	33	0	1

was to (1) study whether CSar could actually evolve a set of interesting association rules, (2) examine the behavior of the system under different configurations. With these objectives in mind, we did the following two experiments.

As our first concern was to analyze whether CSar could evolve the most interesting association rules regardless of having a fixed population size. Therefore, we compared CSar with Apriori [3], probably the most influential association rule miner, on the *zoo* problem [4]. We selected the *zoo* problem for this analysis since Apriori only works on problems described by categorical attributes and the zoo problem satisfies this requirement. More specifically, the *zoo* problem is defined by (1) fifteen binary attributes which indicate whether the animal has a total of fifteen characteristics such as whether it has tail or hair and (2) two categorical attributes that can take more than two values and which represent the number of legs and the type of animal.

Secondly, we studied the impact of using the two different procedures to create association rule candidates and of using progressively bigger maximum intervals. For this purpose, we ran CSar (1) with both antecedent- and consequent-grouping strategies to create association sets candidates and (2) with different maximum interval lengths on a collection of real-world problems extracted from the UCI repository [4] and from local repositories [7]. The characteristics of these problems are reported in Table 1.

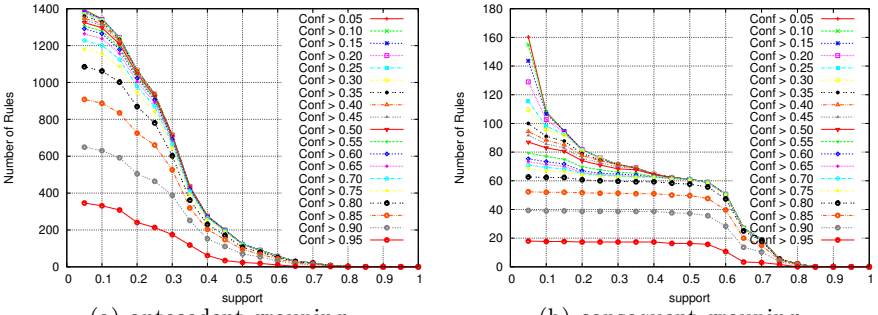
In all runs, CSar employed the following configuration: $num\_iterations = 100\,000$, popSize $= 6\,400$, $conf_0 = 0.95$, $\nu = 10$, $\theta_{mna} = 10$, $\{\theta_{del}, \theta_{GA}\} = 50$, $\theta_{exp} = 1000$, $P_\chi = 0.8$, $\{P_{I/R}, P_\mu, P_C\} = 0.1$, m_0=0.2. Association set subsumption was activated in all runs.

5 Analysis of the Results

With the aim of the experiments in mind, in what follows we discuss about the experimental results.

5.1 Ability of CSar to Discover Interesting Rules

In order to study the ability of CSar to extract interesting association rules, we first compared the system with Apriori on a problem with only categorical attributes, the *zoo* problem. CSar was ran with both antecedent-grouping and consequent-grouping strategies. As we wanted to analyze the interestingness of the rules created by the systems, we report the number of rules with different minimum supports and confidences obtained by CSar with the two grouping strategies (see Figure 1). The same information is reported for Apriori in Figure 2; however, in this case, the resulting rules of Apriori have been filtered.



(a) antecedent grouping (b) consequent grouping

Fig. 1. Number of rules evolved with minimum support and confidence for the *zoo* problem with (a) antecedent-grouping and (b) consequent-grouping strategies. The curves are averages over five runs with different random seeds.

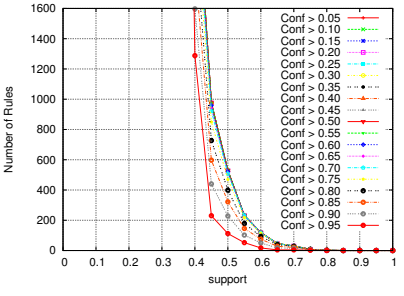


Fig. 2. Number of rules created by Apriori with minimum support and confidence for the *zoo* problem. Lower confidence and support are not shown since Apriori creates all possible combinations of attributes, exponentially increasing the number of rules.

Table 2. Comparison of the number of rules evolved by CSar with antecedent- and consequent-grouping strategies to form the association set candidates with the number of rules evolved by Apriori at high support and confidence values

		Confidence								
		antecedent grouping			**consequent grouping**			**A-priori**		
		0.4	**0.6**	**0.8**	**0.4**	**0.6**	**0.8**	**0.4**	**0.6**	**0.8**
Support	**0.40**	275 ± 30	271 ± 27	230 ± 23	65 ± 10	63 ± 9	59 ± 9	2613	2514	2070
	0.50	123 ± 4	123 ± 4	106 ± 3	61 ± 8	61 ± 8	58 ± 8	530	523	399
	0.60	58 ± 2	58 ± 2	51 ± 4	51 ± 8	51 ± 8	47 ± 7	118	118	93
	0.70	21 ± 1	21 ± 1	19 ± 1	19 ± 2	19 ± 2	18 ± 2	30	30	27
	0.80	2 ± 0	2 ± 0	2 ± 0	2 ± 0	2 ± 0	2 ± 0	2	2	2
	0.90	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0	0	0
	1.00	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0	0	0

That is, Apriori is a two-phase algorithm that exhaustively explores all the feature space, discovers all the itemsets with a minimum predefined support, and creates all the possible rules with these itemsets. Therefore, some of the rules supplied by Apriori are included in other rules. We consider that a rule r_1 is included in another rule r_2 if r_1 has, at least, the same variables with the same values in the rule antecedent and the rule consequent as r_2 (r_1 may have more variables). In the results provided herein, we removed from the final population all the rules that were included by other rules. Thus, we provide an upper bound of the number of *different rules* that can be generated.

Two important observations can be made from these results. Firstly, the results clearly show that Apriori can create a higher number of rules than CSAr (for the sake of clarity, Table 2 specifies the number of rules for support values ranging from 0.4 to 1.0 and confidence values of {0.4,0.6,0.8}). This behavior was expected, since CSar has a limited population size, while Apriori returns all possible association rules. Nevertheless, it is worth noting that CSAr and Apriori found exactly the same number of highly interesting rules; that is, both systems discovered two rules with both confidence and support higher than 0.8. This highlights the robustness of CSar, whose mechanisms guide the system to discover the most interesting rules.

Secondly, focusing on the results reported in Figure 1, we can see that the populations evolved with the antecedent-grouping strategy are larger than those built with the consequent-grouping strategy. This behavior will be also present, and discussed in more detail, in the extended experimental analysis conducted in the next subsection.

5.2 Study of the Behavior of CSar

After showing that CSar can create highly interesting association rules in a case-study problem characterized by categorical attributes, we now extend the experimentation by running the system on 16 real-world data sets. We ran the system with (1) antecedent-grouping and consequent-grouping strategies and (2)

Table 3. Average (± standard deviation of the) number of rules with support and confidence greater than 0.60 created by CSar with antecedent- and consequent-grouping strategies and with maximum interval sizes of MI={0.10, 0.25, 0.50}. The average and standard deviation are computed on five runs with different random seeds.

	antecedent			**consequent**		
	MI=0.10	***MI=0.25***	***MI=0.50***	***MI=0.10***	***MI=0.25***	***MI=0.50***
adl	135 ± 3	294 ± 15	567 ± 66	46 ± 1	74 ± 3	147 ± 23
ann	1736 ± 133	1765 ± 79	1702 ± 135	478 ± 86	525 ± 112	489 ± 34
aud	2206 ± 80	2017 ± 147	1999 ± 185	1014 ± 12	982 ± 100	880 ± 215
aut	84 ± 14	192 ± 7	710 ± 106	25 ± 6	58 ± 3	188 ± 6
bpa	11 ± 4	174 ± 15	365 ± 42	17 ± 2	100 ± 4	123 ± 22
col	134 ± 14	188 ± 7	377 ± 64	180 ± 13	191 ± 7	198 ± 8
gls	33 ± 4	160 ± 17	694 ± 26	23 ± 2	89 ± 6	205 ± 23
H-s	28 ± 1	61 ± 4	248 ± 32	13 ± 1	29 ± 1	92 ± 13
irs	0 ± 0	0 ± 0	50 ± 5	0 ± 0	0 ± 0	28 ± 8
let	0 ± 0	113 ± 17	991 ± 40	0 ± 0	103 ± 6	205 ± 13
pim	4 ± 1	93 ± 9	570 ± 51	3 ± 0	53 ± 5	154 ± 25
tao	0 ± 0	0 ± 0	8 ± 1	0 ± 0	0 ± 0	5 ± 2
thy	46 ± 2	152 ± 4	350 ± 27	29 ± 2	80 ± 3	160 ± 2
wdbc	0 ± 0	419 ± 43	1143 ± 131	0 ± 0	145 ± 17	304 ± 16
wne	116 ± 9	273 ± 48	536 ± 34	26 ± 3	65 ± 9	137 ± 17
wpbc	0 ± 0	0 ± 0	740 ± 234	0 ± 0	0 ± 0	264 ± 34

allowing intervals of maximum length $maxInt = \{0.1, 0.25, 0.5\}$ for continuous variables. Note that by using different grouping strategies we are changing the way how the system creates association set candidates; therefore, as competition is held among rules within the same association set, the resulting rules can be different in both cases. On the other hand, having an increasing larger interval length for continuous variables enables the system to obtain more general rules.

Table 3 reports the number of rules, with confidence and support greater than or equal to 0.6, created by the different configurations of CSar. All the reported results are averages of five runs with different random seeds.

Comparing the results obtained with the two different grouping schemes, we can see that the antecedent-grouping strategy yielded larger populations than the consequent-grouping strategy, on average. This behavior was expected since the antecedent grouping creates smaller association sets, and thus, maintains more diversity in the population. Nonetheless, a closer examination of the final population indicates that the difference in the final number of rules decreases if we only consider the rules with the highest confidence and support. For example, considering all the rules with confidence and support greater than or equal to 0.60, the antecedent-grouping strategy results in populations 2.16 bigger than those of the consequent-grouping strategy. However, considering only the rules with confidence and support greater than or equal to 0.85, the average difference in the population length gets reduced to 1.12. This indicates a big proportion of the most interesting rules are discovered by the two strategies. It is worth

highlighting therefore that the lower number of rules evolved by the consequent-grouping strategy can be considered as an advantage, since the strategy avoids creating and maintaining uninteresting rules in the population, which implies a lower computational time to evolve the population.

Focusing on the impact of varying the interval length, the results indicate that for lower maximum interval lengths CSar tends to evolve rules with less support. This behavior can be easily explained as follows. Large maximum interval length enable the existence of highly general rules, which will have higher support. Moreover, if both antecedent and consequent variables are maximally general, rules will also have high confidence. Taking this idea to the extreme, rules that contain variables whose intervals range from the minimum value to the maximum value for the variable will have maximum confidence and support. Nonetheless these rules will be uninteresting for human experts. On the other hand, small interval lengths may result in more interesting association rules, though too small lengths may result in rules that denote strong associations but have less support. This highlights a tradeoff in the setting of this parameter, which should be adjusted for each particular problem. As a rule of thumb, similarly to what can be done with other association rule miners, the practitioner may start setting small interval lengths and increase them in case of not obtaining rules with enough support for the particular domain used.

6 Summary, Conclusion, and Further Work

In this paper, we presented CSar, a Michigan-style LCS designed to evolve quantitative association rules. The experiments conducted in this paper have shown that the method holds promise for online extraction of both categorical and quantitative association rules. Results with the *zoo* problem indicated that CSar was able to create interesting categorical rules, which were similar to those built by Apriori. Experiments with a collection of real-world problems also pointed out the capabilities of CSar to extract quantitative association rules and served to analyze the behavior of different configurations of the system. These results encourage us to study the system further with the aim of applying CSar to mine quantitative association rules from new challenging real-world problems.

Several future work lines can be followed in light of the present work. Firstly, we aim at comparing CSar with other quantitative association rule miners to see if the online architecture can extract knowledge similar to that obtained by other approaches that go several times through the learning data set. Actually, the online architecture of CSar makes the system suitable for mining association rules from changing environments with concept drift [1]; and we think that the existence of concept drift may be a common trait in many real-world problems to which association rules have historically been applied such as profile mining from customer information. Therefore, it would be interesting to analyze how CSar adapts to domains in which variable associations change over time.

Acknowledgements

The authors thank the support of *Ministerio de Ciencia y Tecnología* under projects TIN2008-06681-C06-01 and TIN2008-06681-C06-05, *Generalitat de Catalunya* under Grant 2005SGR-00302, and *Andalusian Government* under grant P07-TIC-3185.

References

1. Aggarwal, C. (ed.): Data streams: Models and algorithms. Springer, Heidelberg (2007)
2. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Washington D.C, pp. 207–216 (May 1993)
3. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, Santiago, Chile, pp. 487–499 (September 1994)
4. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository, University of California (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
5. Bacardit, J., Krasnogor, N.: Fast rule representation for continuous attributes in genetics-based machine learning. In: GECCO 2008: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, pp. 1421–1422. ACM, New York (2008)
6. Bernadó-Mansilla, E., Garrell, J.M.: Accuracy-based learning classifier systems: Models, analysis and applications to classification tasks. Evolutionary Computation 11(3), 209–238 (2003)
7. Bernadó-Mansilla, E., Llorà, X., Garrell, J.M.: XCS and GALE: A comparative study of two learning classifier systems on data mining. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 2001. LNCS (LNAI), vol. 2321, pp. 115–132. Springer, Heidelberg (2002)
8. Cai, C.H., Fu, A.W.-C., Cheng, C.H., Kwong, W.W.: Mining association rules with weighted items. In: International Database Engineering and Application Symposium, pp. 68–77 (1998)
9. Divina, F.: Hybrid Genetic Relational Search for Inductive Learning. PhD thesis, Department of Computer Science, Vrije Universiteit, Amsterdam, the Netherlands (2004)
10. Fukuda, T., Morimoto, Y., Morishita, S., Tokuyama, T.: Mining optimized association rules for numeric attributes. In: PODS 1996: Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 182–191. ACM, New York (1996)
11. Goldberg, D.E.: Genetic algorithms in search, optimization & machine learning, 1st edn. Addison-Wesley, Reading (1989)
12. Holland, J.H.: Adaptation in natural and artificial systems. The University of Michigan Press (1975)
13. Hong, T.P., Kuo, C.S., Chi, S.C.: Trade-off between computation time and number of rules for fuzzy mining from quantitative data. International Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems 9(5), 587–604 (2001)

14. Houtsma, M., Swami, A.: Set-oriented mining of association rules. Technical Report RJ 9567, Almaden Research Center, San Jose, California (October 1993)
15. Kaya, M., Alhajj, R.: Genetic algorithm based framework for mining fuzzy association rules. Fuzzy Sets and Systems 152(3), 587–601 (2005)
16. Lent, B., Swami, A.N., Widom, J.: Clustering association rules. In: Procedings of the IEEE International Conference on Data Engineering, pp. 220–231 (1997)
17. Mata, J., Alvarez, J.L., Riquelme, J.C.: An evolutionary algorithm to discover numeric association rules. In: SAC 2002: Proceedings of the 2002 ACM Symposium on Applied Computing, pp. 590–594. ACM, New York (2002)
18. Miller, R.J., Yang, Y.: Association rules over interval data. In: SIGMOD 1997: Proceedings of the 1997 ACM SIGMOD International Conference on Management of data, pp. 452–461. ACM, New York (1997)
19. Núñez, M., Fidalgo, R., Morales, R.: Learning in environments with unknown dynamics: Towards more robust concept learners. Journal of Machine Learning Research 8, 2595–2628 (2007)
20. Salleb-Aouissi, A., Vrain, C., Nortet, C.: Quantminer: A genetic algorithm for mining quantitative association rules. In: Veloso, M.M. (ed.) Proceedings of the 2007 International Join Conference on Artificial Intelligence, pp. 1035–1040 (2007)
21. Savasere, A., Omiecinski, E., Navathe, S.: An efficient algorithm for mining association rules in large databases. In: Proceedings of the 21st VLDB Conference, Zurich, Switzerland, pp. 432–443 (1995)
22. Srikant, R., Agrawal, R.: Mining quantitative association rules in large relational tables. In: Jagadish, H.V., Mumick, I.S. (eds.) Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, pp. 1–12 (1996)
23. Wang, C.-Y., Tseng, S.-S., Hong, T.-P., Chu, Y.-S.: Online generation of association rules under multidimensional consideration based on negative border. Journal of Information Science and Engineering 23, 233–242 (2007)
24. Wang, K., Tay, S.H.W., Liu, B.: Interestingness-based interval merger for numeric association rules. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, KDD, pp. 121–128. AAAI Press, Menlo Park (1998)
25. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation 3(2), 149–175 (1995)
26. Wilson, S.W.: Generalization in the XCS classifier system. In: 3rd Annual Conf. on Genetic Programming, pp. 665–674. Morgan Kaufmann, San Francisco (1998)
27. Wilson, S.W.: Get real! XCS with continuous-valued inputs. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 1999. LNCS (LNAI), vol. 1813, pp. 209–219. Springer, Heidelberg (2000)