First Approach toward On-line Evolution of Association Rules with Learning Classifier Systems

Albert Orriols-Puig[1], Jorge Casillas[2], and Ester Bernadó-Mansilla[1]

[1]Grup de Recerca en Sistemes Intel·ligents
Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, 08022, Barcelona, Spain
[2]Department of Computer Science and Artificial Intelligence.
University of Granada, 18071, Granada, Spain

aorriols@salle.url.edu, casillas@decsai.ugr.es, esterb@salle.url.edu

ABSTRACT

This paper presents CSar, a Michigan-style Learning Classifier System which has been designed for extracting 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 on-line and so it is prepared to adapt its knowledge to changes in the variable associations hidden in the stream of unlabeled data quickly and efficiently. Preliminary 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 for further investigating on CSar.

Categories and Subject Descriptors

I.2.6 [**Learning**]: concept learning, knowledge acquisition

General Terms

Algorithms

Keywords

Genetic algorithms, learning classifier systems, unsupervised learning, association rules.

1. INTRODUCTION

Association rule mining [2] aims at extracting interesting associations among the attributes 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, 11]. Later on, several researchers concentrated on extracting association rules from data described by continuous attributes [7, 20], which posed new challenges to the field. Several algorithms proposed to previously apply a discretization method to transform the original data to binary values [13, 15, 20, 22] and then use a binary association rule miner. This lead to further research on designing discretization procedures that avoided losing useful information. Other approaches mined interval-based association rules and permitted the algorithm to independently move the interval bound of each rule's variable [14]. Also fuzzy modeling was introduced to create fuzzy association rules (e.g., see [10, 12]).

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 on-line, so that data may be made available in form of streams [1, 16]. 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, 16]. However, few proposals of on-line binary association rule miners can be found in the literature, and they are only able to deal with problems with categorical attributes (e.g., see [21]).

In this paper, we address the problem of mining association rules from streams of examples on-line. We propose a Learning Classifier System (LCS) whose architecture is inspired by XCS [23, 24] and UCS [5], 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 tested on two real-life static scenarios; one containing categorical attributes and the other with continuous attributes. The results on the problem with categorical attributes indicate that CSar can evolve rules of similar interest as those created by Apriori [3], 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

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

categorical data). The results on the problem with continuous attributes denotes 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. All these results indicate that the method holds promise and leads us to further apply the method to domains with changing dynamics.

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 provides the results of the preliminary experiments, which support the suitability of the method. Finally, Section 5 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 $X \in I$. An itemset X has a support $supp(X)$ which is computed as

$$supp(X) = |X(T)|/|T|. \quad (1)$$

That is, the support is the number of transactions in the database which have the itemset X divided by the total number of transactions in the database. An itemset is said to be a *large 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.,

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

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.,

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

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 (see, for example, [6, 19]). All these algorithms used the same methodology as Apriori to mine association rules, which basically consisted of two different phases: (i) identification of all large itemsets (i.e., all itemsets whose support was greater than $minsupp$), and (ii) generation of association rules from these large 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 to reflect 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 [20] 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 [13, 15]. A completely different approach was taken in [14], where a GA-based algorithm 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, what 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 [18] 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 [14, 18], 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. In the next section, CSar is described in detail.

3. DESCRIPTION OF CSAR

CSar is a Michigan-style LCSs for mining interval-based association rules from data that contain both quantitative and categorical attributes. The learning architecture of CSar is inspired by UCS [5] and XCS [23, 24]. 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 on-line and evolves this rule set by means of a steady-state genetic algorithm that is applied to population niches. As follows, a detailed description of the system is provided.

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

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

in which 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. For quantitative attributes, a similar representation to the XCSR one is used [25], 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 defined value 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 $tcreate$.

3.2 Learning Interaction

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 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 (i) the covering operator, (ii) the procedures to create association set candidates, (iii) the association set subsumption mechanism, and (iv) the parameter update procedure. Next section explains in more detail the discovery component.

3.2.1 Covering Operator

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 is been 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., $l_i = e_i - rand(maxInt/2)$ and $u_i = e_i + rand(maxInt/2)$, 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.

3.2.2 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 [23] and classification [5] 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. That is, 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.

3.2.3 Association Set Subsumption

A subsumption mechanism inspired by the one presented in [24] was designed with the aim of reducing the number of different rules that express the same 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 $u_i \geq u_j$).

3.2.4 Parameter's 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 as follows. Support is computed as:

$$supp = \frac{cm}{ctime - tcreate}, \tag{4}$$

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

$$conf = \frac{cm}{exp}. \tag{5}$$

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

$$F = (conf \cdot supp)^{\nu}, \tag{6}$$

where ν is a user-set parameter that permits to control 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.

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 *genetic algorithm* (GA) [9] 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 [8], where the probability of selecting a classifier k is

$$p^k_{sel} = \frac{F^k}{\sum_{i \in [A]} F^i}. \tag{7}$$

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's 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: (i) introduction/removal of antecedent variables (with probability $P_{I/R}$), (ii) mutation of variable's values (with probability P_μ), and (iii) 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 the rule or one of the antecedent variables has to be removed. 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. 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 [24] 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 the 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

$$p_k = \frac{d_k}{\sum_{\forall j\in[P]} d_j}, \tag{8}$$

where

$$d_k = \begin{cases} \frac{as\cdot num\cdot F_{[P]}}{F^k} & \text{if } exp^k > \theta_{del} \text{ and } F^k < \delta F_{[P]} \\ as\cdot num & \text{otherwise.} \end{cases} \tag{9}$$

Thus, the deletion algorithm balances the classifier's 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.

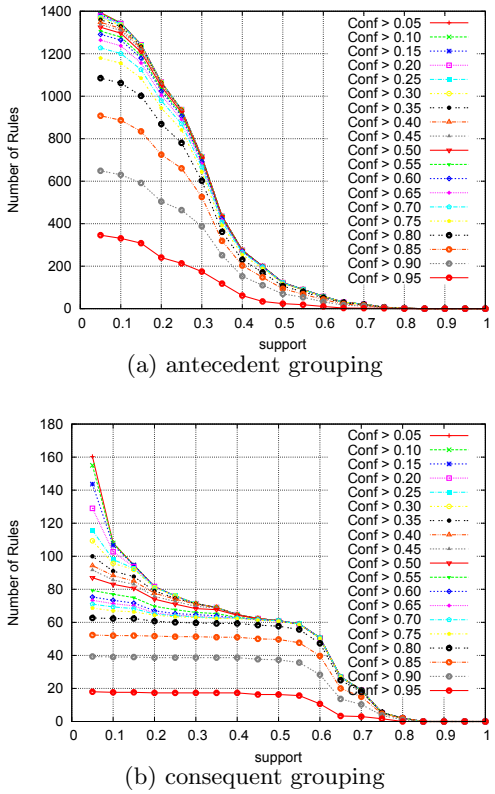
3.4 Rule Set Reduction

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.

4. EXPERIMENTS

This section presents some preliminary results that show the capabilities of CSar to evolve association rules with high support and confidence on-line. For this purpose, we selected two real-life problems from the UCI repository [4]: the *zoo* problem and the Wisconsin-breast cancer diagnosis problem (*wbcd*). The zoo problem is represented by fifteen binary attributes and two categorical attributes which can take more than two values (one of these attributes is the class of the problem). We selected a problem with only categorical attributes to be able to compare the results with Apriori [3], probably the most influential association rule miner, which only works for categorical data. The wbcd problem consists of nine continuous attributes and a nominal attribute that is the class of the problem. Therefore, we applied CSar to this problem to see its capabilities to create quantitative association rules. Although both data sets are classification problems, the information of the class is used as another input attribute to extract association rules. CSar was run on the whole data set with the following configuration parameters: 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. The two strategies for the formation of association sets, i.e., grouping by antecedent and grouping by consequent, were applied. Association set subsumption was activated in all runs.



(a) antecedent grouping

(b) consequent grouping

Figure 1: Number of rules evolved with minimum support and confidence for the *zoo* problem with (a) antecedent grouping and (b) consequent grouping strategies.

Figure 1 shows the number of rules created by CSar with minimum support and confidence with the two strategies for the formation of association set candidates. The results are averages of four runs with different random seeds. The

Table 1: Comparison of the number of rules evolved by CSar with antecedent and consequent grouping 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**	274.7	270.7	230.3	64.7	62.7	59.3	2613.0	2514.0	2070.0
	0.50	122.7	122.7	106.0	61.0	61.0	57.7	530.0	523.0	399.0
	0.60	57.7	57.7	50.7	50.7	50.7	47.3	118.0	118.0	93.0
	0.70	21.0	21.0	19.0	19.0	19.0	18.0	30.0	30.0	27.0
	0.80	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
	0.90	0.0	0.0	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	0.0	0.0

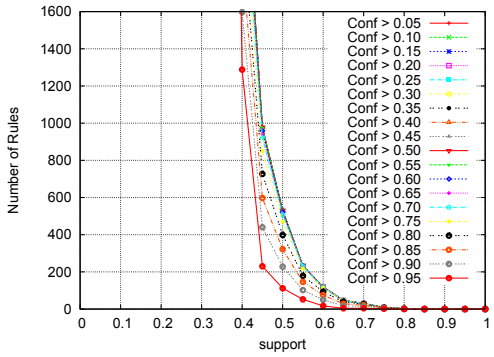


Figure 2: Number of rules created by Apriori with minimum support and confidence for the *zoo* problem. Lower confidence and support are not shown since Apriori creates all possible combinations of attributes, exponentially increasing the number of rules.

two graphics highlight the differences between the two types of association set formation. The populations evolved with the antecedent grouping strategy are larger than those built with the consequent grouping strategy. This behavior was expected since the antecedent grouping creates smaller association sets, and thus, maintains more diversity in the population. Nonetheless, it is worth noting that the number of rules with high confidence and support evolved by the two strategies is approximately the same. That is, antecedent grouping creates more rules but with low confidence and support. The most interesting rules, i.e., the rules that have higher support and confidence, are discovered by the two strategies. This can be considered as an advantage for the consequent grouping, since it avoids creating and maintaining uninteresting rules in the population, which implies a lower computational time to evolve the population.

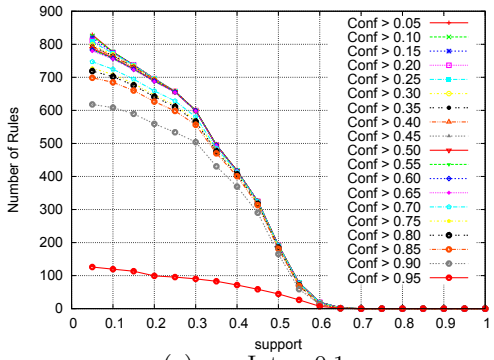
Figure 2 provides the number of rules created by Apriori [3] in the *zoo* problem. Apriori is a two-phase algorithm that exhaustively explores all the feature space, discovers all the itemsets with a minimum predefined support, and creates all the possible rules with these itemsets. Therefore, some of 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's antecedent and the rule's 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. Table 1 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}. Note that as the minimum confidence and support increase, the number of rules created by the three systems becomes more similar.

Next, we applied CSar to a problem with continuous attributes, *wbcd*, to analyze the capabilities of CSar to create quantitative rules. Figures 3 and 4 show the number of rules with minimum support and confidence evolved by CSar with antecedent grouping and consequent grouping strategies. For each of the two strategies, we ran the system letting the size of the variable's intervals be, at most, the ratio of 0.1, 0.5, and 0.9 the whole range of the variable. An example of the quantitative rules evolved by CSar is shown as follows:

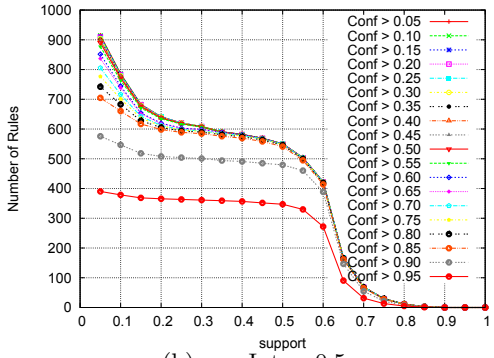
if $X7 \in [1,2] \rightarrow X8 \in [1,2]$ sup=0.668 , conf=0.975
if $X8 \in [1,2] \rightarrow X4 \in [2,3]$ sup=0.659 , conf=0.718

where X_i is the *ith* variable of the problem.

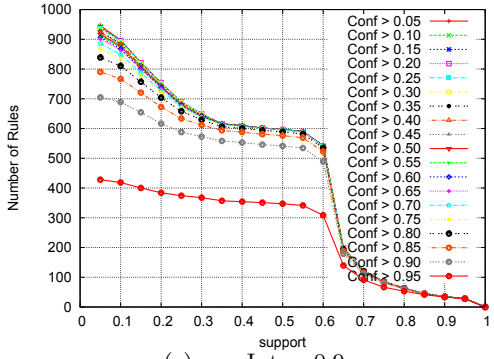
Several observations can be drawn from these results. First, the results permit to extend the conclusions extracted for categorical attributes to continuous attributes. For a given maximum interval size, antecedent grouping results in a higher number of rules, most of which are rules with low confidence and support. Moreover, for antecedent grouping, the number of rules decreases for $maxInt = 0.9$ since few rules with high confidence and support take over the population. That is, larger association sets with highly general and supported rules are created which may impair the population diversity. Second, note that the support of the rules is lower for low values of the maximum interval size. This behavior can be easily explained as follows. Larger maximum interval sizes 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. This highlights the importance of setting appropriate maximum interval sizes for each particular problem.



(a) maxInt = 0.1

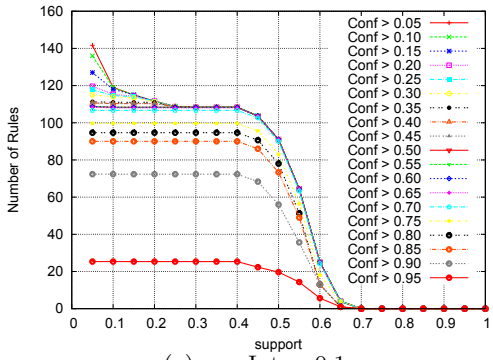


(b) maxInt = 0.5

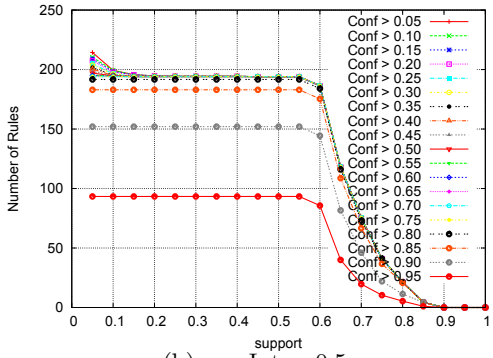


(c) maxInt = 0.9

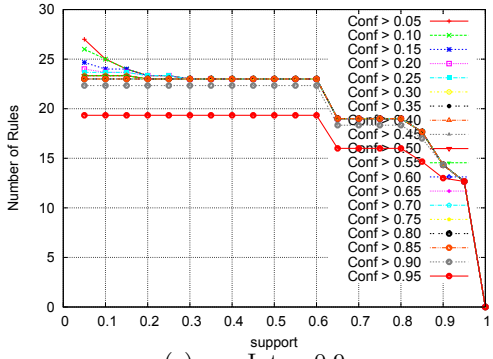
Figure 3: Number of rules evolved with minimum support and confidence for the *wbcd* problem with antecedent grouping strategy and setting the maximum interval size to (a) maxInt = 0.1, (b) maxInt = 0.5, and (c) maxInt = 0.9.



(a) maxInt = 0.1



(b) maxInt = 0.5



(c) maxInt = 0.9

Figure 4: Number of rules evolved with minimum support and confidence for the *wbcd* problem with consequent grouping strategy and setting the maximum interval size to (a) maxInt = 0.1, (b) maxInt = 0.5, and (c) maxInt = 0.9.

5. SUMMARY, CONCLUSION, AND FURTHER WORK

In this paper, we presented CSar, a Michigan-style Learning Classifier System designed to evolve quantitative association rules. The preliminary experiments done in this paper show that the method holds promise for on-line 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 the *wbcd* problem also pointed out the capabilities of CSar to extract quantitative association rules. These results encourage us to further investigate on the system with the aim of applying CSar to mine quantitative association rules from new challenging real-life 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 on-line architecture can extract knowledge similar to that obtained by other approaches that go several times through the learning data set. Actually, the on-line architecture of CSar makes

the system suitable for mining association rules from changing environments with concept drift [1], which we think that is common in many real-life problems in 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. In addition to this, new innovations could be introduced to the system, such as the use of a linguistic fuzzy representation as done in Fuzzy-UCS [17] with the aim of evolving fuzzy association rules. All these innovations and analyses point toward the refinement of the system to apply it to new challenging real-life problems.

Acknowledgements

The authors would like to thank the *Ministerio de Educación y Ciencia* for its support under projects TIN2005-08386-C05-01 and TIN2005-08386-C05-04, and *Generalitat de Catalunya* for its support under grants 2005FI-00252 and 2005SGR-00302.

6. REFERENCES

[1] C. Aggarwal. *Data streams: Models and algorithms*. Springer, 2007.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington D.C., May 1993.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, pages 487–499, Santiago, Chile, September 1994.

[4] A. Asuncion and D. J. Newman. *UCI Machine Learning Repository: [http://www.ics.uci.edu/∼mlearn/MLRepository.html]*. University of California, 2007.

[5] E. Bernadó-Mansilla and J. Garrell. Accuracy-based Learning Classifier Systems: Models, analysis and applications to classification tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

[6] C. H. Cai, A. W.-C. Fu, C. H. Cheng, and W. W. Kwong. Mining association rules with weighted items. In *International Database Engineering and Application Symposium*, pages 68–77, 1998.

[7] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. In *PODS '96: Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 182–191, New York, NY, USA, 1996. ACM.

[8] D. E. Goldberg. *Genetic algorithms in search, optimization & machine learning*. Addison Wesley, 1 edition, 1989.

[9] D. E. Goldberg. *The design of innovation: Lessons from and for competent genetic algorithms*. Kluwer Academic Publishers, 1 edition, 2002.

[10] T. P. Hong, C. S. Kuo., and S. C. Chi. 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.

[11] M. Houtsma and A. Swami. Set-oriented mining of association rules. Technical Report RJ 9567, Almaden Research Center, San Jose, California, October 1993.

[12] M. Kaya and R. Alhajj. Genetic algorithm based framework for mining fuzzy association rules. *Fuzzy Sets and Systems*, 152(3):587–601, 2005.

[13] B. Lent, A. N. Swami, and J. Widom. Clustering association rules. In *Procedings of the IEEE International Conference on Data Engineering*, pages 220–231, 1997.

[14] J. Mata, J. L. Alvarez, and J. C. Riquelme. An evolutionary algorithm to discover numeric association rules. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 590–594, New York, NY, USA, 2002. ACM.

[15] R. J. Miller and Y. Yang. Association rules over interval data. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 452–461, New York, NY, USA, 1997. ACM.

[16] M. Núñez, R. Fidalgo, and R. Morales. Learning in environments with unknown dynamics: Towards more robust concept learners. *Journal of Machine Learning Research*, 8:2595–2628, 2007.

[17] A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. Fuzzy-UCS: a michigan-style learning fuzzy-classifier system for supervised learning. *IEEE Transactions on Evolutionary Computation*, in press.

[18] A. Salleb-Aouissi, C. Vrain, and C. Nortet. Quantminer: A genetic algorithm for mining quantitative association rules. In M. M. Veloso, editor, *Proceedings of the 2007 International Join Conference on Artificial Intelligence*, pages 1035–1040, 2007.

[19] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st VLDB Conference*, pages 432–443, Zurich, Switzerland, 1995.

[20] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 1–12, Montreal, Quebec, Canada, 1996.

[21] C.-Y. Wang, S.-S. Tseng, T.-P. Hong, and Y.-S. Chu. Online generation of association rules under multidimensional consideration based on negative border. *Journal of Information Science and Engineering*, 23:233–242, 2007.

[22] K. Wang, S. H. W. Tay, and B. Liu. Interestingness-based interval merger for numeric association rules. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, KDD*, pages 121–128. AAAI Press, 1998.

[23] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[24] S. W. Wilson. Generalization in the XCS classifier system. In *3rd Annual Conf. on Genetic Programming*, pages 665–674. Morgan Kaufmann, 1998.

[25] S. W. Wilson. Get real! XCS with continuous-valued inputs. In *Learning Classifier Systems. From Foundations to Applications*, LNAI, pages 209–219, Berlin, 2000. Springer-Verlag.