Transactions on Machine Learning
and Data Mining
Vol. 1, No. 1 (2008) 17 - 30
©ISSN:1865-6781 (Journal),
IBaI Publishing ISSN 1864-9734

www.ibai-publishing.org

# Classification Based on Consistent Itemset Rules

Yohji Shidara, Mineichi Kudo, and Atsuyoshi Nakamura

Graduate School of Information Science and Technology,
Hokkaido University,
Kita 14, Nishi 9, Kita-ku, Sapporo 060-0814, Hokkaido, Japan
{shidara, atsu, mine}@main.ist.hokudai.ac.jp

**Abstract.** We propose an approach to build a classifier composing consistent (100% confident) rules. Recently, associative classifiers that utilize association rules have been widely studied, and it has been shown that the associative classifiers often outperform traditional classifiers. In this case, it is important to collect high-quality (association) rules. Many algorithms find only rules with high support values, because reducing the minimum support to be satisfied is computationally demanding. However, it may be effective to collect rules with low support values but high confidence values. Therefore, we propose an algorithm that produces a wide variety of 100% confident rules including low support rules. To achieve this goal, we adopt a specific-to-general rule searching strategy, in contrast to previous approaches. Our experimental results show that the proposed method achieves higher accuracies in several datasets taken from UCI machine learning repository.

**Keywords:** Consistent Itemset Rules, Data Mining, Classification

## 1 Introduction

*Associative classifiers*, which integrate association mining and classification, have been widely studied as a new classification approach [1–6]. According to several reports [1–3], higher classification accuracies are achieved by associative classifiers than by using traditional classifiers such as C4.5 [7] and RIPPER [8]. The associative classifiers assume that the input dataset is a set of itemsets, that is, a transaction database (databases in table-form are converted into transaction databases beforehand.). Here, a transaction is a set of items. The pioneer of associative classification, CBA [1], builds a classifier from a set of association rules obtained by an association rule mining technique such as Apriori [9]. Here, only high quality-rules are selected to construct a ruleset (a classifier).

In many mining techniques, the rules are found in a general-to-specific manner. In this case, starting from a null item, the number of items used in a temporal rule condition increases during the rule narrowing process. Note that a rule with a null item in the condition part explains every instance. In such an approach, items are added to a rule condition in such a way that a rule maintains a minimum support. Such a minimum-support requirement brings the efficiency of the mining process. One of challenging issues is to extract more rules having lower supports (below a required minimum support) but sufficiently high confidences to improve association classifiers. This has not been carried so far due to the impractical increase in computational cost that is mainly consumed to examine many combinations of items while keeping a small value of support. We overcome this difficulty by taking the reverse strategy, that is, a specific-to-general approach. In this approach, confidence is given more priority than support. As a result, regardless of their values of support, more rules with a high confidence value can be found efficiently. Note that an individual instance can be regarded as a 100% confident rule itself, while the support value of the corresponding rule is quite low (such a rule might explain only one instance.). They are the most specific rules. We merge some instances by taking intersection (the set of common items among them) to make them more general. This merging is done only when the consistency (explaining instances of one class only) is maintained. As a consequence, we obtain the most general rules, in the set inclusion relation, keeping consistency. In addition, such rules are expected to be highly interpretable because they represent some unique patterns for a class.

In the proposed method, we consider combinations of instances instead of combinations of items. The latter is more popular than the former. This exchange enables us to have an algorithm that runs in a linear order with respect to the number of items. Although the naive version of the proposed algorithm still requires a combinatorial number of examinations of instances, we can reduce the computational cost in its randomized version.

Our contribution is summarized as follows: 1) a novel rule extraction method is proposed as an extension of the subclass method [10–12], so that a transaction database consisting of itemsets is now able to be dealt with, 2) a classification rule combining obtained consistent (association) rules is proposed, and 3) it is shown that the proposed classifier achieves higher classification accuracies in several datasets than do such as C4.5 [7], RIPPER [8], CBA [1], CMAR [2] and CPAR [3].

The initial version of the classifier is proposed in [13]. In this paper, we developed the classifier by simplifying rules while maintaining consistency. Experimental results show that this modification improves the average classification accuracy of the proposed classifier. Moreover, we discuss the effect of setting of discretization granularity for numerical attributes on the performance of our classifier.

## 2 Methodology

Our method is built on the basis of the subclass method [10–12], which is a classifier only applicable to numerical attributes. We expand the framework of the subclass method so as to deal with transaction databases.

### 2.1 Problem setting

Let $D$ be a dataset, $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ be the set of all items in $D$, and $C = \{c_1, c_2, \ldots, c_k\}$ be the set of the class labels. We refer to a pair of an itemset $X \subseteq \mathcal{I}$ and a class label $c \in C$ as an *instance*, that is, $(X, c) \in D$ denotes an *instance*. For simplicity, we also call $X$ *instance*. A rule $r$ is written in the form $r : A \to c$, where $A \subseteq \mathcal{I}$ and $c \in C$. If $A \subseteq X$ holds, we say that "itemset $A$ covers (explains) instance $X$" or "rule $r : A \to c$ covers (explains) instance $X$." Let $D_c$ denote the family of the itemsets of the instances whose class label is $c$ (*positive instances*), and let $D_{\overline{c}}$ be the itemsets of the remaining instances (*negative instances*). If itemset $A$ does not cover any instance of $D_{\overline{c}}$, then we say that itemset $A$ is *consistent (to $D_{\overline{c}}$)*. For an itemset family $S$, let $R(S)$ denote the set of the common items seen in every instance of $S$, that is, $R(S) = \bigcap_{X \in S} X$.

For class $c$, a *subclass* $S$ is defined as a subset of $D_c$ that satisfies the following:

1. $R(S)$ is consistent to $D_{\overline{c}}$.
2. $S$ is maximal, that is, no instance in $D_c$ can be added to $S$ while keeping consistency of $R(S)$.

Here, $R(S)$ is called *Consistent Common Itemset (CCI)* in class $c$ ($\in C$) if $S$ is a subclass of $c$. The family of the subclasses for class $c$ is denoted by $\Omega_c$ and the set of all subclasses over all classes is denoted by $\Omega$ ($= \bigcup_{c \in C} \Omega_c$). Then, what we want to obtain is the family of subclasses $\Omega$ as well as CCIs ($R(S)$'s for $S \in \Omega$) for all the classes.

**An example:** For an illustrative dataset (Table 1), there are two CCIs for **not-play** class and five CCIs for **play** class as follows:

1. {sunny, humid} → not-play
2. {rainy, windy} → not-play
3. {windless, normal-humidity} → play
4. {overcast} → play
5. {rainy, windless} → play
6. {mild-temperature, normal-humidity} → play
7. {sunny, normal-humidity} → play

Here, each rule corresponds to one subclass (e.g., #1 rule corresponds to $S = \{1, 2, 4\}$) and each itemset in the condition part shows one CCI (e.g., #1 rule shows $R(S) = \{\text{sunny, humid}\}$). The procedure of finding these rules will be described in the following section.

**Table 1.** An illustrative dataset.

| ID | $X$ | $c$ |
|---|---|---|
| 1 | {sunny, hot, humid, windless} | not-play |
| 2 | {sunny, hot, humid, windy} | not-play |
| 3 | {rainy, cool, normal-humidity, windy} | not-play |
| 4 | {sunny, mild-temperature, humid, windless} | not-play |
| 5 | {rainy, mild-temperature, humid, windy} | not-play |
| 6 | {overcast, hot, humid, windless} | play |
| 7 | {rainy, mild-temperature, humid, windless} | play |
| 8 | {rainy, cool, normal-humidity, windless} | play |
| 9 | {overcast, cool, normal-humidity, windy} | play |
| 10 | {sunny, cool, normal-humidity, windless} | play |
| 11 | {rainy, mild-temperature, normal-humidity, windless} | play |
| 12 | {sunny, mild-temperature, normal-humidity, windy} | play |
| 13 | {overcast, mild-temperature, humid, windy} | play |
| 14 | {overcast, hot, normal-humidity, windless} | play |

## 2.2 Rule extraction procedure

We employ a randomized algorithm [10] to obtain a subset $\Omega' \subseteq \Omega$ to economize the computational cost of enumerating all members of $\Omega$. According to a theoretical analysis [10], the suboptimal subclass family $\Omega'$ (a subset of CCIs) obtained by this randomized algorithm with a fixed iteration number has the following properties:

1. Larger subclasses (CCIs with larger coverage rates) are more likely to be found than smaller subclasses in the earlier iterations (The procedure is described later.).
2. Characteristic subclasses are also more likely to be found. Here, a "characteristic subclass" is one that includes instances covered by only a few subclasses.

Now let us explain the algorithm briefly. The algorithm executes multiple scans for all positive instances. The scanning is repeated $t$ times for a given $t$. Each scan is made according to a random order, that is, a permutation $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{|D_c|})$ randomly chosen. According to order $\sigma$, we merge the positive instances into $S$ (initialized by the empty set) in order, as long as the addition does not break the consistency of $R(S)$. Otherwise we skip the positive instance. Because of the fact that the merging process does not make $R(S)$ larger than before, in the set inclusion relation, and the fact that all positive instances are scanned, it is guaranteed that one subclass is necessarily found by one scan. Here, the dataset is assumed to be consistent in the weakest sense, that is, all of the positive instances themselves are assumed to be consistent to the negative instances. We may find the same subclass for different $\sigma$'s. Thus, the duplicated subclasses are removed in the last stage.

If we test all possible $|D_c|!$ permutations, we can obtain the complete family $\Omega_c$ of the subclasses of class $c$. However, even for not so large $|D_c|$, this number becomes infeasible. So we terminate the iteration by a given iteration number $t$. As described already, we can expect that almost all important subclasses are found even for a moderate iteration number, say $t = 1,000$, for each target class. For the constant $t$, the randomized algorithm runs in $O(|D_c||D_{\overline{c}}||\mathcal{I}|)$ for each target class, where $|D_c|$ is the number of instances in a target class, $|D_{\overline{c}}|$ is the number of instances in non-target classes and $|\mathcal{I}|$ is the number of the items.

**An example:** Let us show how to obtain a rule "{sunny, humid} $\rightarrow$ not-play" from Table 1.

Assume that the permutation is decided as $\sigma = (1, 2, 3, 4, 5)$.

1. Instance #1 is put into $S$. Here, $S = \{1\}$ and $R(S) =$ {sunny, hot, humid, windless}. This $R(S)$ is consistent, because no negative instance (Nos. 6–14) has all of the items at once.
2. When instance #2 is put into $S$, $R(S)$ becomes {sunny, hot, humid}. $R(S)$ is still consistent.
3. If instance #3 is put into $S$, $R(S)$ becomes $\emptyset$, and the consistency is broken (because $\emptyset$ is included in any itemset). So we skip instance #3 for merging.
4. When instance #4 is put into $S$, $R(S)$ becomes {sunny, humid}. The consistency of the itemset is still kept.
5. If instance #5 is put into $S$, $R(S)$ becomes {humid}, and the consistency is again broken because the negative instances #6, #7 and #13 include {humid}. So we skip this instance too.

Through this scan we obtain a CCI: $R(S) =$ {sunny, humid} with the corresponding of $S = \{1, 2, 4\}$.

Note that another scan with a different $\sigma$ produces another rule. For example, with $\sigma = (3, 5, 1, 2, 4)$ we have another rule"{rainy, windy} $\rightarrow$ not-play." By repeating scanning with all 120(= 5!) permutations, all of the subclasses for class **not-play** are obtained. In this example, there are only two subclasses for class **not-play** and five subclasses for class **play**.

## 2.3 Classification

Once CCIs have been obtained in each class, we can proceed to build a classifier. We design a classifier relying on the following belief (note that all of the rules are 100% confident to the training set):

1. Rules with larger coverage rates are more reliable.
2. Class assignment by a larger number of rules is more reliable.

In order to satisfy both of these, we introduce a score to a rule and sum up the scores of rules that explain a given instance. Here, the score of a rule is measured by its coverage rate of positive instances. We classify an instance into a class with the highest score, sum of the rules covering the instance.

Let us assume that a (class unknown) instance is given with an itemset $A$. Next, let $\mathcal{S}_{A,c}$ be the set of subclasses $S$ $(\in \Omega_c)$ whose $R(S)$ is included in $A$, that is,

$$\mathcal{S}_{A,c} = \{S \in \Omega_c \mid R(S) \subseteq A\}.$$

Then our classification rule is written as

$$\hat{c} = \arg\max_{c \in C} \sum_{S \in \mathcal{S}_{A,c}} \frac{|S|}{|D_c|}.$$

Here, $|S|/|D_c|$ $(0 < |S|/|D_c| \leq 1)$ is the score of subclass $S$ (or equivalently CCI $R(S)$). Note that the score can be obtained without additional calculation during the rule generalization process by counting the number of instances put into the subclass.

A tie-break is resolved by assigning it to the class with the largest population. If none of the rules are matched, the largest class is also chosen. We call this combining way the "Consistent Common Itemsets Classifier (shortly, CCIC)" approach.

## 3 Experimental Results

We conducted an experiment to evaluate the performance of the CCIC approach.

### 3.1 Settings

According to the literature [1–3], we used 26 datasets from UCI Machine Learning Repository [14]. A summary of datasets is shown in Table 2.

Every instance in the dataset is converted into an itemset. Numerical attributes are discretized into 5 bins. Here, the intervals of specifying the bins are taken so as to make the populations of the attribute values equal in each attribute. The number of iterations is set to $t = 1,000$. The negative instances that break the consistency of any positive instance are removed in order to keep the consistency of the dataset in a weakest sense.

We also present the accuracies of C4.5 [7], RIPPER [8], CBA [1], CMAR [2] and CPAR [3] as competitors. All of their results are taken from reference [3]. This is allowed because the experimental conditions are almost the same.

### 3.2 Accuracy of CCIC

The results of CCIC are shown in Table 3. The average accuracy is obtained by 10-fold cross validation. In total, CCIC was the third in accuracy but was the first in the number of wins (8/26).

As can be seen from the results, the performance of the CCIC approach depends on the problem. Let us examine the reasons for this dependency. Since CCIC uses only consistent rules, the level of accuracy may decrease if a sufficient number of consistent rules is not found. Even if many consistent rules are found,

**Table 2.** Summary of the datasets. Three missing rates are: 1) the rate of attributes including missing values, 2) the rate of instances including missing values, and 3) the rate of missing values to all values.

| dataset | #attr. | | | #inst. | #classes | major class | missing | | |
|---|---|---|---|---|---|---|---|---|---|
| | | (cat.) | (num.) | | | (%) | (attr.) | (inst.) | (val.) |
| anneal | 38 | 32 | 6 | 898 | 6 | 0.76 | 0.763 | 1.000 | 0.650 |
| austra | 14 | 8 | 6 | 690 | 2 | 0.56 | - | - | - |
| auto | 25 | 10 | 15 | 205 | 7 | 0.33 | 0.280 | 0.224 | 0.012 |
| breast | 10 | - | 10 | 699 | 2 | 0.66 | 0.100 | 0.023 | 0.002 |
| cleve | 13 | 7 | 6 | 303 | 2 | 0.54 | 0.154 | 0.023 | 0.002 |
| crx | 15 | 9 | 6 | 690 | 2 | 0.56 | 0.467 | 0.054 | 0.006 |
| diabetes | 8 | - | 8 | 768 | 2 | 0.65 | - | - | - |
| german | 20 | 13 | 7 | 1000 | 2 | 0.70 | - | - | - |
| glass | 9 | - | 9 | 214 | 7 | 0.36 | - | - | - |
| heart | 13 | - | 13 | 270 | 2 | 0.56 | - | - | - |
| hepati | 19 | 13 | 6 | 155 | 2 | 0.79 | 0.789 | 0.484 | 0.057 |
| horse | 22 | 15 | 7 | 368 | 2 | 0.63 | 0.955 | 0.981 | 0.238 |
| hypo | 25 | 18 | 7 | 3163 | 2 | 0.95 | 0.320 | 0.999 | 0.067 |
| iono | 34 | - | 34 | 351 | 2 | 0.64 | - | - | - |
| iris | 4 | - | 4 | 150 | 3 | 0.33 | - | - | - |
| labor | 16 | 8 | 8 | 57 | 2 | 0.65 | 1.000 | 0.982 | 0.357 |
| led7 | 7 | 7 | - | 3200 | 10 | 0.11 | - | - | - |
| lymph | 18 | 15 | 3 | 148 | 4 | 0.55 | - | - | - |
| pima | 8 | - | 8 | 768 | 2 | 0.65 | - | - | - |
| sick | 29 | 22 | 7 | 2800 | 2 | 0.94 | 0.276 | 1.000 | 0.056 |
| sonar | 60 | - | 60 | 208 | 2 | 0.53 | - | - | - |
| tic-tac | 9 | 9 | - | 958 | 2 | 0.65 | - | - | - |
| vehicle | 18 | - | 18 | 846 | 4 | 0.26 | - | - | - |
| waveform | 21 | - | 21 | 5000 | 3 | 0.34 | - | - | - |
| wine | 13 | - | 13 | 178 | 3 | 0.40 | - | - | - |
| zoo | 16 | 16 | - | 101 | 7 | 0.41 | - | - | - |

**Table 3.** Accuracy comparison. The best score is indicated in boldface. The column #CCIs is the average number of CCIs found by the proposed method. The column 'drop' is the average ratio of test instances that are not matched with any rule. The bottom row #bests shows the number of datasets for which the method recorded the best accuracy.

| dataset | C4.5 | RIPPER | CBA | CMAR | CPAR | CCIC | #CCIs | drop |
|---|---|---|---|---|---|---|---|---|
| anneal | 0.948 | 0.958 | 0.979 | 0.973 | **0.984** | 0.966 | 128.0 | 0.02 |
| austra | 0.847 | 0.873 | 0.849 | 0.861 | 0.862 | **0.877** | 714.6 | 0.00 |
| auto | 0.801 | 0.728 | 0.783 | 0.781 | **0.820** | 0.787 | 334.2 | 0.07 |
| breast | 0.950 | 0.951 | 0.963 | **0.964** | 0.960 | **0.964** | 266.5 | 0.00 |
| cleve | 0.782 | 0.822 | **0.828** | 0.822 | 0.815 | **0.828** | 584.9 | 0.00 |
| crx | 0.849 | 0.849 | 0.847 | 0.849 | 0.857 | **0.875** | 716.8 | 0.00 |
| diabetes | 0.742 | 0.747 | 0.745 | **0.758** | 0.751 | 0.723 | 833.6 | 0.01 |
| german | 0.723 | 0.698 | 0.734 | **0.749** | 0.734 | 0.748 | 1635.6 | 0.00 |
| glass | 0.687 | 0.691 | 0.739 | 0.701 | **0.744** | 0.705 | 193.0 | 0.09 |
| heart | 0.808 | 0.807 | 0.819 | 0.822 | 0.826 | **0.837** | 548.0 | 0.00 |
| hepati | 0.806 | 0.767 | 0.818 | 0.805 | 0.794 | **0.827** | 270.3 | 0.01 |
| horse | 0.826 | **0.848** | 0.821 | 0.826 | 0.842 | 0.845 | 601.3 | 0.02 |
| hypo | **0.992** | 0.989 | 0.989 | 0.984 | 0.981 | 0.972 | 183.4 | 0.01 |
| iono | 0.900 | 0.912 | 0.923 | 0.915 | **0.926** | 0.923 | 999.7 | 0.00 |
| iris | **0.953** | 0.940 | 0.947 | 0.940 | 0.947 | 0.933 | 35.0 | 0.02 |
| labor | 0.793 | 0.840 | 0.863 | **0.897** | 0.847 | 0.833 | 77.0 | 0.04 |
| led7 | 0.735 | 0.697 | 0.719 | 0.725 | **0.736** | 0.729 | 153.1 | 0.00 |
| lymph | 0.735 | 0.790 | 0.778 | **0.831** | 0.823 | 0.810 | 260.6 | 0.05 |
| pima | **0.755** | 0.731 | 0.729 | 0.751 | 0.738 | 0.732 | 829.2 | 0.01 |
| sick | **0.985** | 0.977 | 0.970 | 0.975 | 0.968 | 0.941 | 438.1 | 0.01 |
| sonar | 0.702 | 0.784 | 0.775 | 0.794 | 0.793 | **0.836** | 1655.2 | 0.00 |
| tic-tac | 0.994 | 0.980 | **0.996** | 0.992 | 0.986 | 0.989 | 268.9 | 0.00 |
| vehicle | **0.726** | 0.627 | 0.687 | 0.688 | 0.695 | 0.703 | 1715.2 | 0.00 |
| waveform | 0.781 | 0.760 | 0.800 | **0.832** | 0.809 | 0.802 | 2944.7 | 0.02 |
| wine | 0.927 | 0.916 | 0.950 | 0.950 | 0.955 | **0.961** | 407.8 | 0.00 |
| zoo | 0.922 | 0.881 | 0.968 | **0.971** | 0.951 | 0.891 | 8.8 | 0.11 |
| average | 0.8334 | 0.8293 | 0.8469 | **0.8522** | 0.8517 | 0.8476 | | |
| #bests | 5 | 1 | 2 | 7 | 5 | **8** | | |

their coverage rates might be low, that is, the consistent rules may explain only a part of the positive instances. When an instance is not matched by any rule, it is assigned to the largest population class in our classifier. This is one possible reason for the performance degradation. Indeed, in some datasets such as auto, glass and zoo, more than 5% of the test instances were not matched with any rule (see column 'drop' in Table 3).

## 4 Removing redundant attributes/items and adjusting the granularity

### 4.1 Model selection

So far, we considered CCIs as consistent and maximal itemsets. In other words, a CCI holds the common properties seen in a maximal set of positive instances but not seen in any negative instance. However, in practice, some redundant items can be included in such a CCI. This is mainly due to the lack of sufficient number of positive instances. When the number of possible items is large and the number of instances explained by a CCI is small, there are, in general, some redundant items in the CCI. By the word "redundant", we mean that some of items can be removed while keeping consistency. It is widely known in pattern recognition that it is better to remain only necessary attributes as long as the consistency is kept. Note that, in our framework, only a single item is chosen from an attribute in each instance. That is, the number of attributes/items should be appropriately chosen according to the number of instances. So, we will carry out removal of redundant attributes in the following.

In our CCIC, there is one principal parameter that affects much the results. That is the number of bins used for discretizing a numerical variable into a set of binary variables (a set of items). In this way, a numerical value is translated into $B$ binary values of which only one value takes one, according to the bin in which the value is. Then the chosen bin is represented as an item. In the previous experiment, we set $B$ to 5 in common to all numerical variables. It should be noted that more bins cause less commonality of instances, that is, the probability of falling in a same bin becomes smaller. As a result, in a large value of $B$, to keep/increase the commonality among positive instances, more attributes are needed. Therefore, $B$ is the key parameter in our approach.

In total, the number of attributes and the number of bins affect both to the performance of our CCI approach. In the viewpoint of model selection, we should select appropriately both the numbers depending on datasets. It is desirable to attain this from the basic statistics of a dataset, e.g. from the number of instances per attribute. However, there are many factors affecting the performance other than such a size. For example, the possible best classification rate is the most useful information, but it is hardly obtained. As a result, we confirm experimentally the effectiveness of removing redundant attributes and of selecting the number of bins.

## 4.2 CCIC with a minimal itemset

We carried out removing of redundant attributes. We check every attribute as a candidate for removal in a random order. If the removal does not hurt the consistency, we remove it. Otherwise we leave the attribute. A processed CCI is referred to as an eCCI (an economized CCI).

**Table 4.** Comparison of the accuracies of CCIC and eCCIC. The figure in parentheses denotes the number of bins in each attribute. The column 'drop' is the average ratio of test instances that are not matched with any rule.

| | accuracy | | | | drop | | |
|---|---|---|---|---|---|---|---|
| dataset | CCIC(5) | eCCIC(5) | eCCIC($B$) | | CCIC(5) | eCCIC(5) | eCCIC($B$) |
| anneal | 0.966 | **0.969** | **0.969** | 5 | 0.02 | 0.02 | 0.02 |
| austra | 0.877 | 0.871 | **0.884** | 3 | 0.00 | 0.00 | 0.00 |
| auto | 0.787 | 0.802 | **0.806** | 10 | 0.07 | 0.01 | 0.00 |
| breast | **0.964** | **0.964** | **0.964** | 5 | 0.00 | 0.00 | 0.00 |
| cleve | 0.828 | **0.834** | **0.834** | 3 | 0.00 | 0.00 | 0.00 |
| crx | **0.875** | **0.875** | **0.875** | 5 | 0.00 | 0.00 | 0.00 |
| diabetes | 0.723 | 0.717 | **0.740** | 3 | 0.01 | 0.01 | 0.00 |
| german | 0.748 | 0.745 | **0.755** | 3 | 0.00 | 0.00 | 0.00 |
| glass | **0.705** | 0.682 | 0.691 | 10 | 0.09 | 0.02 | 0.03 |
| heart | 0.837 | 0.841 | **0.852** | 10 | 0.00 | 0.00 | 0.00 |
| hepati | 0.827 | **0.839** | **0.839** | 5 | 0.01 | 0.01 | 0.01 |
| horse | 0.845 | 0.845 | **0.850** | 3 | 0.02 | 0.02 | 0.01 |
| hypo | 0.972 | 0.974 | **0.984** | 10 | 0.01 | 0.00 | 0.00 |
| iono | **0.923** | 0.920 | 0.920 | 5 | 0.00 | 0.00 | 0.00 |
| iris | 0.933 | 0.933 | **0.940** | 3 | 0.02 | 0.02 | 0.01 |
| labor | 0.833 | 0.850 | **0.907** | 3 | 0.04 | 0.04 | 0.00 |
| led7 | **0.729** | 0.728 | 0.728 | - | 0.00 | 0.00 | 0.00 |
| lymph | 0.810 | 0.831 | **0.865** | 3 | 0.05 | 0.00 | 0.00 |
| pima | **0.732** | 0.729 | 0.729 | 5 | 0.01 | 0.01 | 0.01 |
| sick | 0.941 | 0.942 | **0.977** | 10 | 0.01 | 0.01 | 0.01 |
| sonar | 0.836 | **0.841** | **0.841** | 5 | 0.00 | 0.00 | 0.00 |
| tic-tac | **0.989** | **0.989** | **0.989** | - | 0.00 | 0.00 | 0.00 |
| vehicle | **0.703** | 0.690 | 0.695 | 10 | 0.00 | 0.00 | 0.00 |
| waveform | 0.802 | 0.803 | **0.821** | 3 | 0.02 | 0.01 | 0.01 |
| wine | 0.961 | **0.966** | **0.966** | 5 | 0.00 | 0.00 | 0.00 |
| zoo | 0.891 | **0.933** | **0.933** | - | 0.11 | 0.02 | 0.02 |
| average | 0.8476 | 0.8505 | **0.8598** | | | | |

The result is shown in Table 4 (see the column named eCCIC(5)). On average, eCCIC (the classifier using eCCIs) is better than CCIC at 0.29%. It can also be seen that more test instances are matched by the rules of eCCIC compared with those of CCIC (compare the 'drop' ratio). This means that original CCIs were reduced and the corresponding rules became more general. In almost all cases in which such a generalization is recognized, the accuracy has been improved (see, auto, hypo, lymph, waveform and zoo).

As shown in Table 4, some instances are still not explained by any rule even after the removing prepossessing. On the other hand, as seen in Table 4, CCIC is better than eCCIC for datasets for which many consistent rules are found and almost all instances are covered by them. Therefore, a more systematic way other than a random removal would be worth studying.

## 4.3 Adjustment of bin size

Next, we examined the effect of the bin size $B$ for discretizing numerical variables. Discretization has been studied in a wide range of fields including quantization, data compression, representation of data, rough sets, and so on. Indeed, discretization of numerical values is often discussed in frequent itemset mining (for example, see [15]). In addition, in rough sets, such a trial is called "granularity" [16, 17] and has been well studied.

We compared the classification accuracies of eCCIC for $B = 3, 5, 10$ ($B = 5$ in the previous experiments). The result is shown in Figure 1. Figure 1 shows the classification accuracy vs. the relative data size, that is, the ratio of the number of instances per class to the number of numerical attributes. We plotted those for 11 datasets with numerical attributes only. We can see roughly such a tendency that a rough discretization (a smaller number of bins) works better for large datasets. On the contrary, a finer discretization is better for small datasets.

To examine how degree the number of bins affects the number of attributes used in eCCIs, we investigated the relationship between three values of $B$ and the accuracy. Figure 2 shows the average ratio of used attributes in eCCIC rules. In Figure 2, all datasets are displayed but the accuracy does not change for datasets with categorical attributes only. From Figure 2, we see that the number of attributes increases as the number of bins decreases. This implies that if the number of bins is smaller, then a larger number of attributes is necessary for finding the commonalty among many positive instances. Therefore, the result of Figure 1 is consistent with our basic strategy to use more attributes for a larger dataset and less attributes for a smaller dataset.

It is not easy to determine automatically an appropriate value of $B$, because it depends on the problem. For showing the possibility, the best accuracy achievable using one of three values of $B$ is shown in Table 4. The average accuracy was increased to 0.8598 and the number of wins was 12 of 26 among 5 competitors plus this eCCIC classifier.

## 4.4 Scalability

All of the experiments were performed on a 2 GHz Intel Core Duo PC (running Mac OS X 10.5.1) with 2 GB of main memory. The implementation was not multithreaded. The most time-consuming dataset was waveform; its execution time of 10-fold cross validation (including both training and test) with eCCIC was about 529 seconds. However, the algorithm is easily parallelized to reduce the running time with less overheads because each iteration is completely independent.

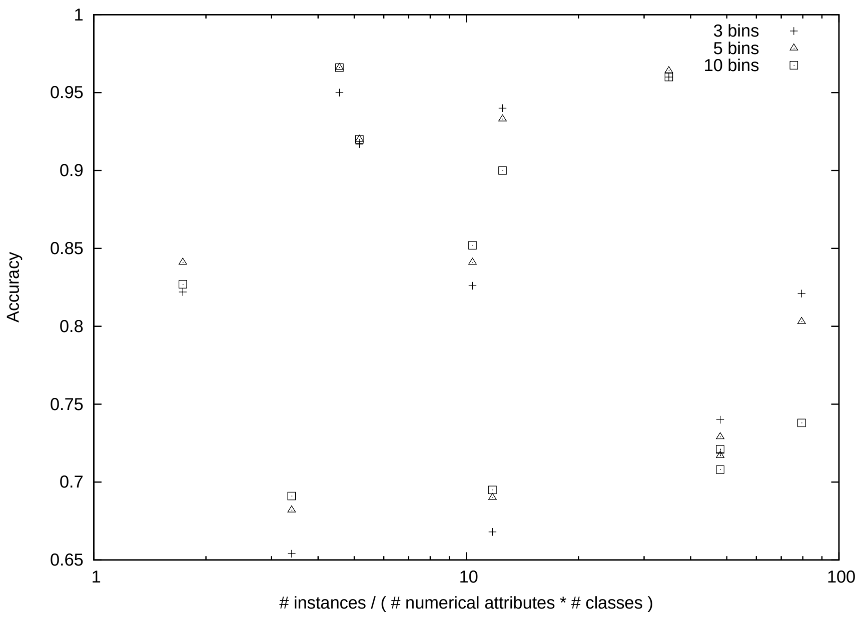

**Fig. 1.** Accuracy change in eCCIC for three different bin numbers. Only numerical datasets are displayed.

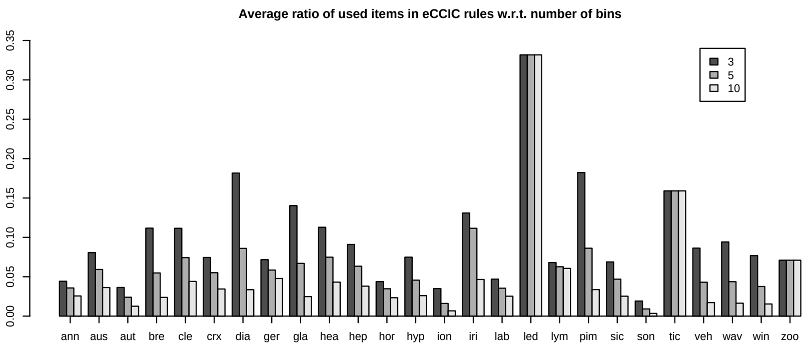

**Fig. 2.** Average ratio of used attributes/items in eCCIC rules.

## 5 Discussion

We have proposed a novel way to collect consistent rules and showed the effectiveness of a classifier using them. At the current stage, we consider only consistent (100% confidence) rules which explain only positive instances. However, in many practical situations, it would be preferable to incorporate with consistency-relaxed rules. When noisy measurement or wrong labelling can happen, such a relaxation is sometimes adequate. To achieve such a relaxation, possible approaches are usage of multiple subsets of instances systematically/randomly selected, such as boosting and bagging, and relaxing the exclusiveness against negative instances in our algorithm.

## 6 Conclusion

A classifier, called CCIC, consisting of consistent rules and its redundancy-removed version eCCIC have been proposed. CCIC and eCCIC combine many consistent itemsets for classification. Thus, they can be seen associative classifiers. Experimental results showed that CCIC and eCCIC outperformed other classifiers in several datasets.

We have also discussed on how degree discretization affects the performance of the proposed classifier. The results suggest that an appropriate choice of discretization ways has a large possibility to improve the performance of the proposed classifier.

In future works, we will consider different ways of combining consistent rules and adoption of consistency-relaxed rules to improve the performance of the classifier. In addition, rule selection should be considered in order to reduce redundancy of the ruleset.

## References

1. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Knowledge Discovery and Data Mining. (1998) 80–86
2. Li, W., Han, J., Pei, J.: CMAR: accurate and efficient classification based on multipleclass-association rules. In: Proceedings of IEEE International Conference on Data Mining (ICDM2001). (2001) 369–376
3. Yin, X., Han, J.: CPAR: Classification based on predictive association rules. In: 3rd SIAM International Conference on Data Mining (SDM'03). (2003)
4. Zaiane, O.R., Antonie, M.L.: Classifying text documents by associating terms with text categories. In: Proceedings of the 13th Australasian database conference. (2002) 215–222
5. Wang, Y., Wong, A.K.C.: From association to classification: Inference using weight of evidence. IEEE Transactions on Knowledge and Data Engineering **15**(3) (March 2003) 764–767
6. Dong, G., Zhang, X., Wong, L., Li, J.: CAEP: Classification by aggregating emerging patterns. In: Discovery Science. (1999) 30–42

7. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc. (1993)
8. Cohen, W.W.: Fast effective rule induction. In: Proceedings of the 12th International Conference on Machine Learning. (1995) 115–123
9. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases (VLDB). (1994) 487–499
10. Kudo, M., Yanagi, S., Shimbo, M.: Construction of class regions by a randomized algorithm: A randomized subclass method. Pattern Recognition **29**(4) (1996) 581–588
11. Kudo, M., Shimbo, M.: Feature selection based on the structual indices of categories. Pattern Recognition **26**(6) (1993) 891–901
12. Kudo, M., Shimbo, M.: Analysis of the structure of classes and its applications – subclass approach. Current Topics in Pattern Recognition Research **1** (1994) 69–81
13. Shidara, Y., Nakamura, A., Kudo, M.: CCIC: Consistent common itemsets classifier. In Perner, P., ed.: Machine Learning and Data Mining in Pattern Recognition. Volume 4571 of Lecture Notes in Computer Science., Leipzig, Germany, Springer (July 2007) 490–498
14. Asuncion, A., Newman, D.: UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html` (2007)
15. 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 (4–6 1996) 1–12
16. Lin, T.Y.: Granular computing: From rough sets and neighborhood systems to info rmation granulation and computing in words. In: European Congress on Intelligent Techniques and Soft Computing. (1997) 1602–1606
17. Lin, T.Y.: Granular Computing on Binary Relation I : Data Mining and Neighborhood Systems, II : Rough Set Representations and Belief Functions. Physica-Verlag (1998)