Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 9 (2014) pp. 235-243
© MIR Labs, www.mirlabs.net/jias/index.html

Classification Model Using Contrast Patterns and GRASP

Hiroyuki MORITA[1] **and Arthur Mahéo**[2]

[1]Osaka Prefecture University, College of Sustainable System Sciences
Sakai, Osaka, JAPAN
morita@eco.osakafu-u.ac.jp

[2]University of Nantes
Nantes, France
arthur.maheo@etu.univ-nantes.fr

Abstract: **The volume of historical purchasing data has become huge, and it includes many kinds of data attributes. Specifically, categorical data, such as product codes, are difficult to handle. If the product is purchased repeatedly, we can aggregate the data and use the product data as a numerical attribute. However, if the item was purchased only once, we can get only very basic information, such as whether it was purchased or not. To use the information more effectively, we can use a subset of these purchased items as a purchasing pattern within the set of items. Some classification predictive models that use these patterns were proposed, including the classification by aggregating contrast patterns (CACP). However, the model sometimes produces too many specific patterns. This is not a problem for predictions, but interpreting the model can become too complicated to implement efficiently.**
In this paper, we propose a method to decrease the number of patterns in the classification model for CACP. The proposed method uses the meta-heuristics algorithm known as greedy randomized adaptive search procedure (GRASP). A computational experiment shows that we can remove extra patterns and construct the model, while maintaining its performance level.
Keywords: contrast pattern; CACP; classification predictive model; GRASP

I. Introduction

A huge volume of historical purchasing data has been accumulated, and the organizations have gathered valuable knowledge from the data through various mining algorithms and methods.The association rule is a well-known mining method that enumerates interesting patterns from point of sale (POS) data. Other mining methods enumerate characteristic patterns for a class to develop a predictive model. Predictive models include classification by aggregating emerging patterns [Dong et al., 1999] (CAEP) and classification by aggregating contrast patterns [Morita and Mao, 2013] (CACP) which have been applied to practical data. However, in most cases, the CAEP and CACP models produce large numbers of patterns. These large numbers of patterns improve the accuracy of the model; however they also make analysis very complicated. In CAEP and CACP, rules can remove some extraneous patterns, but several hundred patterns still remain.
Our study focuses on removing extraneous contrast patterns in the CACP model while preserving the level of performance. Simply removing patterns is easy; however, maintaining the level of performance simultaneously is difficult. To do this, we focused on the number of times contrast patterns that could represent a transaction, and we searched for better combinations of the patterns by using greedy randomized adaptive search procedure (GRASP).
Section 2 discusses related works, and Section 3 explains the combinatorial optimization problem and GRASP. In Section 4, we describe the computational experiments we performed using a benchmark problem. Finally, Section 5 presents the conclusion and future works.

II. Related works

[Bay and Pazzani, 1999] proposed a characteristic pattern for a specific class as the contrast pattern by focusing on the difference between support values, as expressed by Equation 1. And [Morita and Mao, 2013] has proposed CACP using the contrast patterns. Given two different classes pos and neg, suppose that the databases D_{pos} and D_{neg} consist only of transactions that belong to the pos or neg classes, respectively.

$$d(x, D_{pos}) = \quad sup(x, D_{pos}) - sup(x, D_{neg}), \quad d(x, D_{pos}) > 0, \tag{1}$$

where $sup(x, D_{pos})$ denotes the support value defined as follows:

$$sup(x, D_{pos}) = \frac{cnt(x, D_{pos})}{|D_{pos}|}, \tag{2}$$

where $cnt(x, D_{pos})$ and $|D_{pos}|$ denote the number of transactions which match the pattern x in D_{pos} and the number of all transactions in D_{pos}, respectively. $sup(x, D_{neg})$ is defined the same way. As $d(x, D_{pos})$ or $d(x, D_{neg})$ becomes

larger, pattern x becomes more characteristic of the database. Then contrast pattern is defined as below:

Contrast pattern

Pattern x is a contrast pattern for c class, when $d(x, D_c) \geq \theta$, where θ is a predefined threshold value.

A range of $d(x, D_c)$ is from 0 to 1, and when $d(x, D_c) = 0$, the state is neutral. Because $d(x, D_c) = 0$ means that $sup(x, D_{pos}) = sup(x, D_{neg})$. On the contrary, when $d(x, D_{pos}) = 1$, x is the most characteristic contrast pattern for pos class. θ is given the rage from 0 to 1 depending upon data sets. The number of contrast patterns is depended upon the θ value. In short, smaller θ value enumerates larger number of contrast patterns and larger θ value enumerates smaller number of contrast patterns. In addition to this, when the data density is higher, larger number of contrast patterns are enumerated for the same θ value. So we can't know the number of contrast patterns before enumerating. It sometimes makes the performance of the model poor, and we have to search better θ values. To avoid such a situation, we use $topK$ parameter. The $topK$ parameter means that enumerates top K contrast patterns whose $d(x, D_c)$ values are larger. In the following, we use the $topK$ parameter to enumerate contrast patterns.

On the other hand, [Dong et al., 1999] defines an emerging pattern as one where the $growth\ rate$ defined by Equation 3 is greater than a predefined threshold value.

$$g(x, D_{pos}) = \begin{cases} \dfrac{sup(x, D_{pos})}{sup(x, D_{neg})}, & sup(x, D_{neg}) > 0 \\ \infty, & sup(x, D_{neg}) = 0 \end{cases} \quad (3)$$

Additionally, [Dong et al., 1999] proposed CAEP, which is a predictive model for classification using emerging patterns. In CAEP, a score is calculated for the emerging patterns in each class, based on the support value and the growth rate. The score for each class is increased for every transaction that matches the pattern. The class with the highest score is selected as the predictive class. CAEP is a simple and effective method, and it has been applied to some real data. However, in many cases, it requires a large number of emerging patterns to enhance the predictive performance. [Morita and Mao, 2013] proposed CACP, which is similar to CAEP, except it uses contrast patterns. Contrast patterns generally represent many more transactions than emerging patterns; therefore, fewer contrast patterns are required.

Before developing the CACP model, redundant contrast patterns are pruned. Given two contrast patterns x and y in the same class, if $sup(y, D_{pos}) \leq sup(x, D_{pos})$ and $d(y, D_{pos}) \leq d(x, D_{pos})$, then contrast pattern y is removed and x is kept. In the example in Figure 1, y was eliminated after being compared with x, but z was not, because $sup(x, D_{pos}) \leq sup(z, D_{pos})$. The pattern x is not removed, because $d(z, D_{pos}) \leq d(x, D_{pos})$. Therefore, contrast patterns x and z were kept, and only y was removed. Such pruning effectively decreases extraneous contrast patterns; however, in many cases, too many contrast patterns remain.

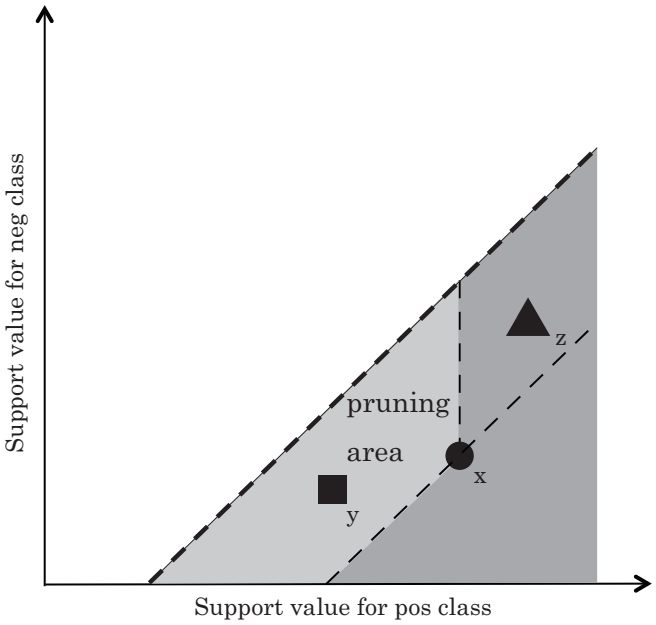

Figure. 1: Pruning area.

Given score of pattern for each class and a relationship between transactions and patterns as table 1 and table 2, respectively. Here, "tid" and "pid" denote "transaction id" and "pattern id", respectively. In the table 1, we can see that selected contrast patterns have a score for only one class. For example, in the case of "pid=1", it has 1.2 score value for *pos* class and no score value for *neg* class.

Table 1: Contrast pattenrs and their score for each class

pid	score for *pos* class	score for *neg* class
1	1.2	0.0
2	0.0	0.7
3	2.5	0.0
4	0.0	3.1
5	0.8	0.0

Table 2: Contrast pattenrs and transactions

tid	pid
1	1 2
2	1 2 5
3	1 3 4
4	3 5
5	3
6	2 5
7	1 4
8	2
9	2 4
10	2 3 4

And table 2 shows relationship between "tid" and "pid". For instance, "tid=1" has two contrast patterns, "pid=1" and "pid=2". "pid=1" has the score for *pos* class, but "pid=2" has the score for *neg* class. So these scores are aggregated for each class shown in figure 2. For all tids, the scores are aggregated in the same. Then, we can make table 3. In case of

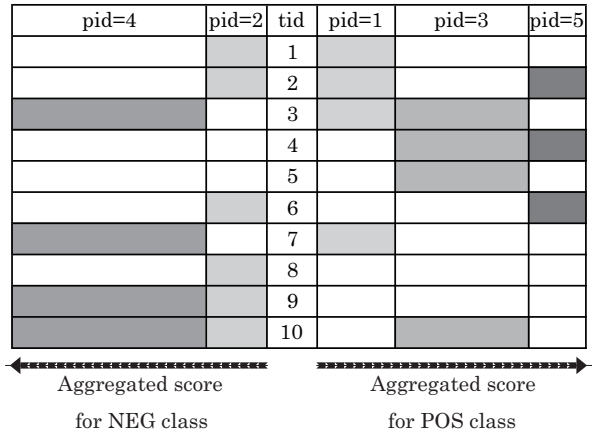

Figure. 2: Example of score aggregation.

"tid=1" of the table, the aggregated scores are 0.7 and 1.2 for *neg* class and *pos* class, respectively. Then the class is predicted as *pos* class, because the aggregated *pos* score is larger than aggregated *neg* score. In the same way, for other tids, each score is aggregated and each transaction is predicted the class. Finally, the predicted classes are compared with actual classes, and result is judged as "TRUE", when the predicted class and the actual class are same. Otherwise, the result is judged as "FALSE". In the example, almost tids are predicted accurately, except in the case of "tid=6". When we observe

Table 3: Contrast pattenrs and their score for each class

tid	Aggregated *neg* score	Aggregated *pos* score	Predicted class	Actual class	Result
1	0.7	1.2	*pos*	*pos*	TRUE
2	0.7	2.0	*pos*	*pos*	TRUE
3	3.1	3.7	*pos*	*pos*	TRUE
4	0.0	3.3	*pos*	*pos*	TRUE
5	0.0	2.5	*pos*	*pos*	TRUE
6	0.7	0.8	*pos*	*neg*	FALSE
7	3.1	1.2	*neg*	*neg*	TRUE
8	0.7	0.0	*neg*	*neg*	TRUE
9	3.8	0.0	*neg*	*neg*	TRUE
10	3.8	2.5	*neg*	*neg*	TRUE

the figure 2 and table 3 carefully, we can notice that contrast pattern of "pid=5" is redundant and unnessesary. In short, we can remove it from the set of contrast patterns and, we can make the accuracy better. When we remove the "pid=5", table 3 is changed to table 4. In the table 4, *pos* score of "tid=6" is decreased form "0.8" to "0.0", because "pid=5" is removed. Any other scores are not changed in the case. Compared the score for *pos* class with the score for *neg* class in "tid=6" after removing, the *neg* score becomes larger by removing the pattern. Then predicted class is changed from "*pos*" to "*neg*" and the result becomes "TRUE". However, "pid=5" is not always redundant and unnessesary. In a case which we don't have "pid=1" and "pid=3", "pid=5" is nessesary, because it can be possible to predict two or theree tids.

Table 4: Aggregated scores after removing a redundunt contrast pattern

tid	Aggregated *neg* score	Aggregated *pos* score	Predicted class	Actual class	Result
1	0.7	1.2	*pos*	*pos*	TRUE
2	0.7	1.2	*pos*	*pos*	TRUE
3	3.1	3.7	*pos*	*pos*	TRUE
4	0.0	2.5	*pos*	*pos*	TRUE
5	0.0	2.5	*pos*	*pos*	TRUE
6	0.7	**0.0**	***neg***	*neg*	**TRUE**
7	3.1	1.2	*neg*	*neg*	TRUE
8	0.7	0.0	*neg*	*neg*	TRUE
9	3.8	0.0	*neg*	*neg*	TRUE
10	3.8	2.5	*neg*	*neg*	TRUE

So the combination of contrast pattern is important, and some patterns are always needed, and some patterns are sometimes nessesary and sometimes are not nessesary like "pid=5".
To further improve existing studies, we propose a method to remove such unnesessary contrast patterns using GRASP [Resende and Ribeiro, 2003] [Feo and Resende, 1995], which is a meta-heuristics algorithm for combinatorial optimization. GRASP consists of two phases: the construction phase and the local search phase as illustrated below.

Algorithm 1 procedure GRASP

Require: Input data(model made by CACP), $runs$ which is the number of the repetition, and RCL parameters shown in algorithm 3
Ensure: Approximate optimal solution
  Solution $\leftarrow \emptyset$;
  for $k = 1, ..., runs$ **do**
    Candidate $\leftarrow \emptyset$;
    Candidate $\leftarrow$ Candidate_Construction;
    Candidate $\leftarrow$ Local_Search(Candidate);
    Solution $\leftarrow$ Solution $\cup$ Candidate;
  end for
  return Solution;

The construction phase determines a number of feasible solutions and selects one as the initial tentative solution randomly and greedily. Starting from the initial tentative solution, the local optimum solution is found and the search process is terminated. Each local optimum solution is kept, but the best local optimum solution can also be tracked. In the CACP model, the set of contrast patterns for each class is enumerated and selected, and the goal is to find the best combination of contrast patterns from this set. The best combination is defined as the subset with the smallest number of patterns and with a predictive accuracy that is the same as or better than the existing CACP. To produce a simple model that is easy to interpret, we use GRASP to find better combinations of contrast patterns. Note that improving the accuracy level is not our objective in this study, although we aim to maintain the current level.

III. Problem definition and the proposed method

Algorithm 2 illustrates the flow of CACP and GRASP.Transaction data and class definition are pro-

vided as the input files, as well as some parameters for CACP and GRASP. For each class, contrast patterns are enumerated, and extra contrast patterns are pruned by the existing procedure. GRASP is added to the flow, and more extraneous contrast patterns are removed. Given the model and training data or test data, the criteria for the prediction, such as accuracy, are calculated.

Algorithm 2 procedure CACP and GRASP

Require: Input files(transactions and class definition) and parameters which are minimum length of the pattern, maximum length of the pattern, minimum support value, $topK$ as mentioned earlier, and GRASP parameters shown in algorithm 1
Ensure: Classification model by using CACP and GRASP
MODEL $\leftarrow \emptyset$;
while each class c **do**
CP$_c \leftarrow$ Enumerate_Contrast_Patterns;
CP$_c \leftarrow$ Pruning(CP$_c$);
MODEL $\leftarrow$ MODEL $\cup$ CP$_c$;
end while
MODEL $\leftarrow$ GRASP_for_CACP(MODEL);
return MODEL;

To confirm the possibility of removing a pattern, we investigated the number of times that a transaction is represented by a pattern. If a contrast pattern matches a transaction, the transaction is "represented" by the pattern; if no pattern matches the transaction, it is called "unrepresented". Some transactions are represented by too many contrast patterns, and some are represented by only a few or none at all. If most transactions are unrepresented, or represented by only a few contrast patterns, we cannot eliminate more contrast patterns, because we would not be able to predict unrepresented transactions. However, we can remove patterns if many other patterns represent the same transactions. In our preliminary experiments, the number of times that a transaction is represented is an important parameter; therefore, we define our combinatorial optimization problem as follows:

$$\begin{aligned} &\min. \quad \sum_{i=1}^{n} x_i \\ &s.t. \quad \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} c(t_j, x_i)}{m} \geq \beta, \\ &x_i = \begin{cases} 1, & \text{if pattern } x_i \text{ is selected,} \\ 0, & \text{otherwise,} \end{cases} \\ &c(t_j, x_i) = \begin{cases} 1, & \text{if } t_j \text{ is represented by } x_i, \\ 0, & \text{otherwise,} \end{cases} \end{aligned} \tag{4}$$

where n denotes the number of candidate patterns obtained using the existing CACP, m denotes the number of total transactions, and β denotes the minimum number of transactions that the patterns have to represent to avoid being eliminated. We select the approximate optimal solution from the candidate patterns that represent more transactions than the minimum β. We assumed that maintaining the degree of representation stabilizes the accuracy of the prediction.

Candidate_Construction in Algorithm 1 uses a restricted candidate list (RCL), as shown in Algorithm 3. At the first step of the GRASP phase, we select candidates from the RCL, and a feasible initial solution is generated for the local search phase, during which the candidate solution is improved.

Algorithm 3 procedure RCL

Require: A candidate map which is an ordered list based on the score of each pattern, and r_size which is the upper bound of size of the restricted candidate list
Ensure: A list of candidates less than a predetermined r_size
RCL $\leftarrow \emptyset$;
while size of RCL $\leq r\_size$ and iterator not at the end **do**
Determine num which the number of items to add to the RCL;
Fetch candidates at iterator's position;
if $num \leq$ size of candidates **then**
Add all candidates to RCL;
else
Select num candidates at random and add to RCL;
end if
Increment position of iterator;
end while
return RCL;

Algorithm 4 procedure candidate map

Require: A set of candidates and fitness function f.
Ensure: An ordered structure on the score of each pattern associated with the list of patterns.
for each x_i in candidates **do**
Calculate score : $f^k(x_i)$;
Add element in the candidate map using its score ;
end for
return candidate map;

To make a candidate map shown in Algorithm 4, we introduce three kinds of fitness functions as follows:

Cover function

This function is used at the beginning of the algorithm as the most important part is : do not leave unrepresented transactions. Unrepresented means that no selected pattern represents transactions. If we have unrepresented transactions, it is difficult to predict their class accurately. Hence, the fitness of a solution component using this function, is the number of unrepresented transactions it can represent. Here, we use the unrepresented count, because overlapping patterns are less interesting towards our goal.

$$f^0(x_i) = \sum_{j=1}^{m} c^0(t_j, x_i),$$

$$c^0(t_j, x_i) = \begin{cases} 1, \text{ if } t_j \text{ is unrepresented and} \\ \quad \text{is represented by } x_i, \\ 0, \text{ otherwise.} \end{cases} \tag{5}$$

Validation function

Next, another important score of an element is the number of transactions it validates. Here, "validate" means that including this pattern would lead predictive class of the transaction to be correct. The rationale behind this choice is that there is a larger chance of improving the quality of solution by adding good patterns to it. We rely on the improvement function to check whether there were worsening moves leading to better solutions in the end.

$$f^1(x_i) = \sum_{j=1}^{m} c^1(t_j, x_i),$$

$$c^1(t_j, x_i) = \begin{cases} 1, \text{ if } t_j \text{ is represented by} \\ \quad x_i \text{ precisely.} \\ 0, \text{ otherwise.} \end{cases} \tag{6}$$

Class function

Finally, this function is another heuristic based on the fact that a good pattern belongs to the class which still needs the most validation. In other words, if we still have more *neg* transactions than *pos* transactions to validate, it is better to use more patterns which lead to validate *pos* class. This function relies on the trend of the program. The trend is simply which class needs more coverage at the moment.

$$f^2(x_i) = \sum_{j=1}^{m} c^2(t_j, x_i),$$

$$c^2(t_j, x_i) = \begin{cases} 1, & \text{if } t_j \text{ which is } \textit{pos} \text{ class} \\ & \quad \text{and is unrepresented} \\ & \quad \text{is represented by } x_i, \\ -1, & \text{if } t_j \text{ which is } \textit{neg} \text{ class} \\ & \quad \text{and is unrepresented} \\ & \quad \text{is represented by } x_i, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Of these, cover function and class function have effect on improvement which represents unrepresented transactions. On the other hand, validation function has effect on the degree of representation precisely. We expext that candidate map is constructed by these functions and local search phase find better solution efficiently.

In the local search phase, neighborhood solutions are generated by the 01-exchange algorithm. The neighborhood solutions are generated by only changing 1s to 0s randomly, because we have to remove patterns to improve the objective function. We can improve the tentative solution whenever we can, then we repeat the phase to reach the local optimum solution. GRASP requires only three parameters, $runs$, the size of RCL, and β. In following section, we describe how we applied our improved method to a practical problem, and we examine the results.

IV. Computational experiments

In our experiment, we applied our method to a mushroom data set, which is a popular classification problem data set provided by the from UCI Machine Learning Repository [Bache and Lichman, 2013].The data has two classes (edible and poisonous) and 22 attributes that can be used as explanatory variables. We divided the data set into a training data set and test data set, as shown in Table 5.

Table 5: Number of transactions for each data set

	Training data	Test data	Total
#all transaction	5415	2709	8124
#edible transaction	2792	1416	4208
#poisonous transaction	2623	1293	3916

First, we applied the original CACP method to the data set and confirmed the results. As mentioned earlier, the $topK$ parameter means the number of contrast patterns which have larger $d(x, D_{pos})$ and $d(x, D_{neg})$ values and the value affects the performance of the model from the view points of the accuracy and the computational time. We experimented with three values of $topK$ (1000, 10000, and 100000) and compared the results.

Table 6: Number of contrast patterns and computational time

	Not pruned		Pruned	
$topK$	$\#cp$	CPUt(sec.)	$\#cp$	CPUt(sec.)
1000	2846	196.181	51	8.53
10000	19123	1536.50	103	29.44
100000	202089	29081.9	228	549.33

Table 6 shows a significant difference in the numbers of contrast patterns between non-pruned cases and pruned case on each $topK$, when using the original CACP model. Therefore, computational time increases in the non-pruned cases.

Table 7: Number of cp used in the CACP model and the corresponding computational time

CACP results for training data

	No pruned		Pruned	
$topK$	Accuracy	#unrepresented	Accuracy	#unrepresented
1000	0.897	0	0.896	0
10000	0.885	0	0.885	0
100000	0.911	0	0.893	0

CACP results for test data

	Not pruned		Pruned	
$topK$	Accuracy	#unrepresented	Accuracy	#unrepresented
1000	0.897	1	0.906	1
10000	0.885	1	0.893	1
100000	0.913	1	0.904	1

The performance of the CACP model is shown in Table 7. We define accuracy as the percentage of correct predictive transactions out of all the transactions in the data set. The accuracy increases as $topK$ increases, and the accuracy between pruned and non-pruned cases are similar. What is interesting to note is that the number of contrast patterns in the pruned case is very small, but the performance does not get worse. Therefore, we make the following two observations:

1. The original CACP is effective at pruning contrast patterns.

2. More contrast patterns can be removed by the proposed method, because the accuracy level is stable.

The limitation is currently unknown, but a model can use less than 50 contrast patterns. Figure 3 illustrates the distribution of the represented transactions from the training and test data sets in the pruned and $topK = 100000$ case. The difference between the training data set and the test data set is insignificant; therefore, pruned contrast patterns can represent the transactions for test data as effectively as for training data. Moreover, the extent of representation is greater than expected. For example, 10% of the transactions are represented by more than 80 patterns. Based on these observations, we can remove more contrast patterns by using GRASP with CACP and still maintain the performance level with the same data.

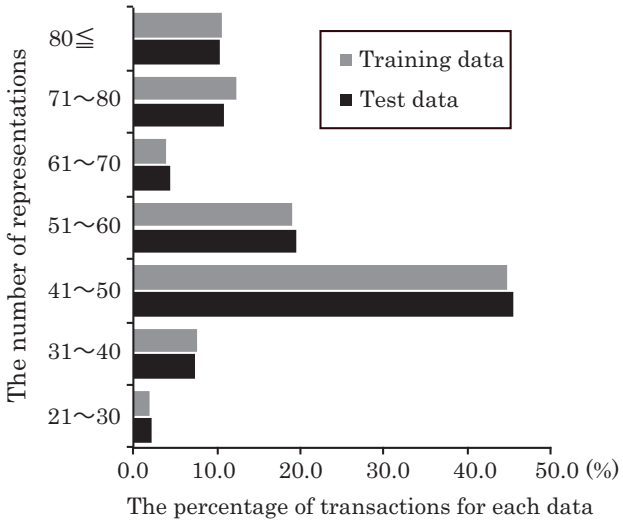

Figure. 3: Distribution of the number of representations

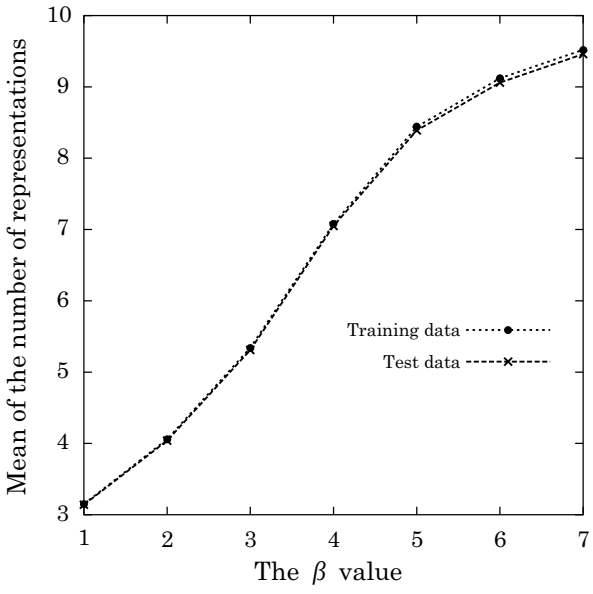

Figure. 4: Mean of the number of representations and the β value

In the following GRASP experiments, we changed the lower bound β, which is the mean of the number of representation,

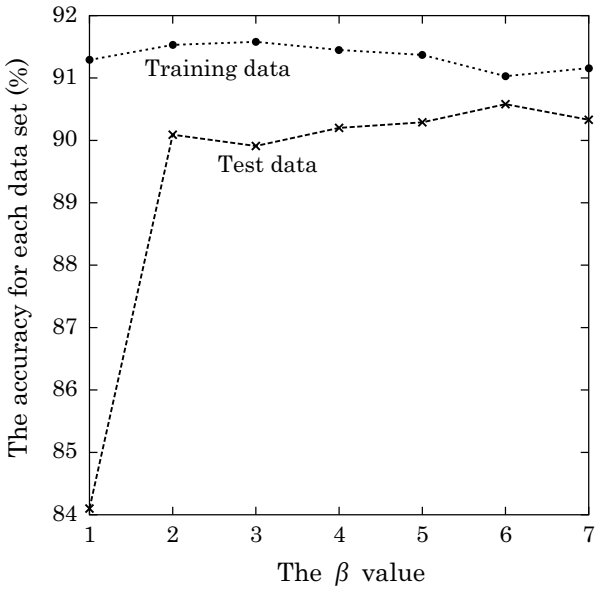

Figure. 5: Accuracy and the β value

to increase from 1 to 7. For each β, we applied GRASP 100 times. The size of RCL was set to 5.

Figure 4 illustrates the mean of the number of representations for each β on both data sets. The mean increases with β, but the results showed no difference between both the data sets. In short, it is very stable from the view point of the mean of representation, and we can expect that it is possible to use smaller β, such as $\beta = 3$.

Figure 5 shows the mean of accuracy for each β on both data sets. The accuracy with the training data is stable for every β; however, the accuracy with the test data is worse when $\beta = 1$ than when $\beta \geq 2$. Figure 6 shows the maximum

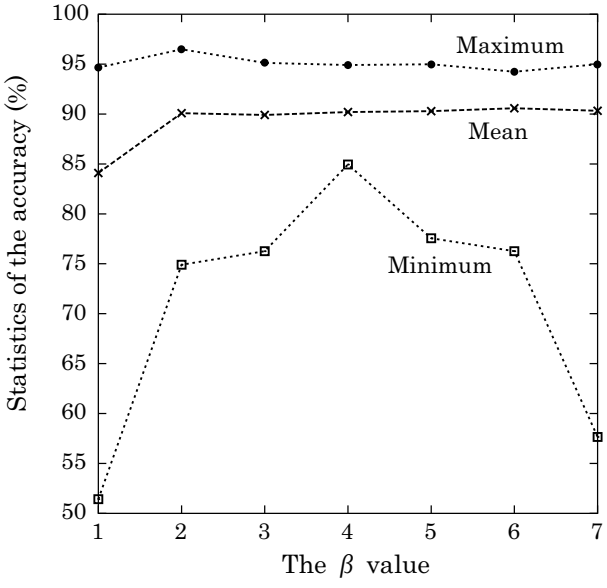

Figure. 6: Statistics of the accuracy and the β value

value, mean, and minimum value of the accuracy for each β with the test data. The minimum values for $\beta = 1$ and $\beta = 7$ are poor. When $\beta = 1$, the mean of the accuracy is also poor.

Therefore, we assume that $\beta = 1$ is not sufficient for the data set.

Table 8: Computational time (sec.) of GRASP

β	mean	standard deviation	maximum	minimum
1	2.82	1.38	7.92	0.65
2	2.73	1.51	8.32	0.92
3	2.39	1.14	7.61	0.76
4	2.74	1.27	8.37	0.91
5	2.62	1.08	7.27	1.05
6	2.43	1.00	5.99	1.05
7	2.47	1.12	8.09	1.04

Table 8 shows the statistics of CPU time(sec.) for 100 implementations for each β. The differences in the means are not significant, but we can see some deviations, because the computation time depends on the number of iterations in the local search. From the viewpoint of the accuracy level and its stability, the results for $\beta = 4$ are shown in more detail.

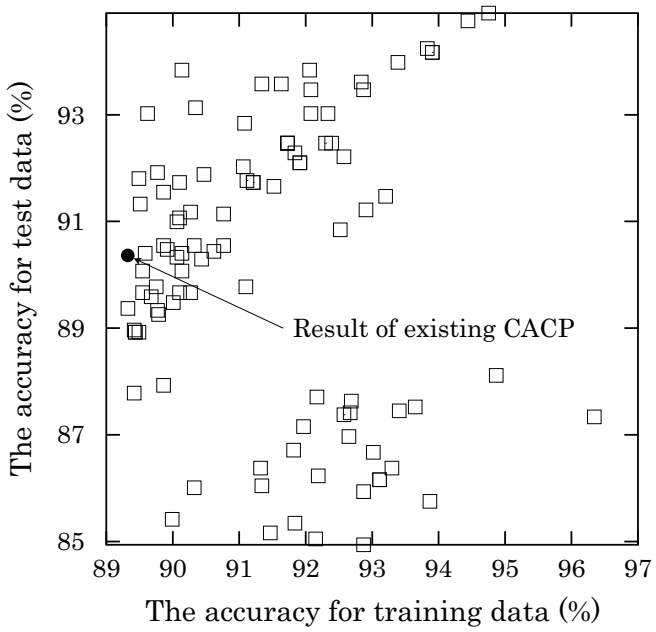

Figure. 7: Comparison of original CACP result with the new results

Figure 7 illustrates the accuracy of each model with each data set. Here, each model is constructed by giving $\beta = 4$ and it is implemented 100 times by changing random seed. So one point in figure 7 illustrates one model, and it has each the accuracy for training data and test data, respectively. All square points denote the accuracy after implementing GRASP, and black circle denotes the accuracy when using CACP without GRASP. In about 60% of 100 runs in the test data set, the accuracy of CACP with GRASP is better than that of CACP alone; however, in the remaining about 40% the accuracy is poor.

Table 9 shows a comparison of the accuracy between the best case and the worst case, considering the number of representations. As shown, Case 33 is the worst case and Case 95 is the best case, based on accuracies in the test data set. The two cases differ by only one pattern, in terms of numbers of contrast patterns used in the model, and the means of the number

Table 9: Best case and worst case

		training data		test data	
case	#pattern	accuracy	represented	accuracy	represented
33	12	92.872	6.750	84.939	6.667
95	11	94.755	6.472	94.906	6.419

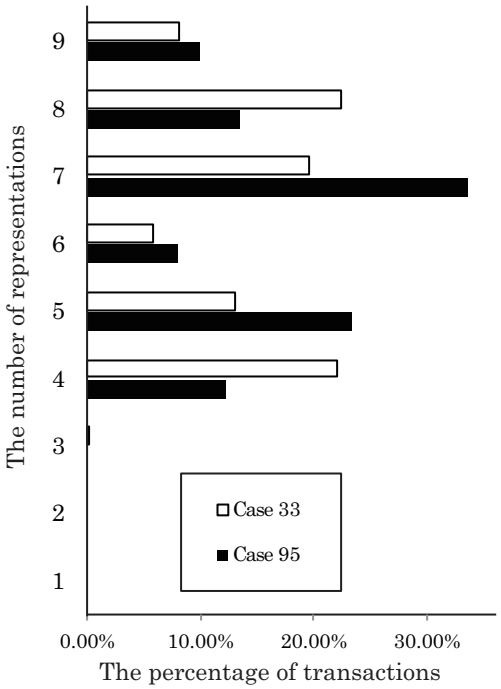

Figure. 8: Distribution for each representation using GRASP

of representations for transactions are similar. It is very important and interesting, because the results shows that it is possible to make sufficient model by using small number of contrast patterns, and that the combination of the patterns is important. The difference in accuracies with the training data is approximately 2%, but with the test data, the gap increases to approximately 10%.

Figure 8 shows the ratio of transactions with different numbers of representations. By comparing Case 33 with Case 95, we can see that the distribution is similar. Therefore, the number of contrast patterns and the distribution of the representations are not the cause of the differences in performance. Conversely, selected patterns and their combination are more important, so we have to observe that in more detail. Figures 9 and 10 show the scatter plots for the Case 95 and the Case 33, respectively. In both figures, pattern x_1 is same, however the other patterns are different. And in both cases, we can see that each class has only $5 \sim 7$ patterns. In the best case, some of the contrast patterns, such as pattern x_2 in figure 9, are near the ideal point, where the support value for one class is 100% and that for another class is 0%. We believe that the support value is one of the factors that influence the performance level. To clear the above point, we have to make clear that how near from the ideal point the patterns are needed and what combination of such the patterns is effective as our future work.

By applying the GRASP algorithm over CACP, we can remove many extraneous contrast patterns. As shown in this experiment, we were able to reduce 228 contrast patterns

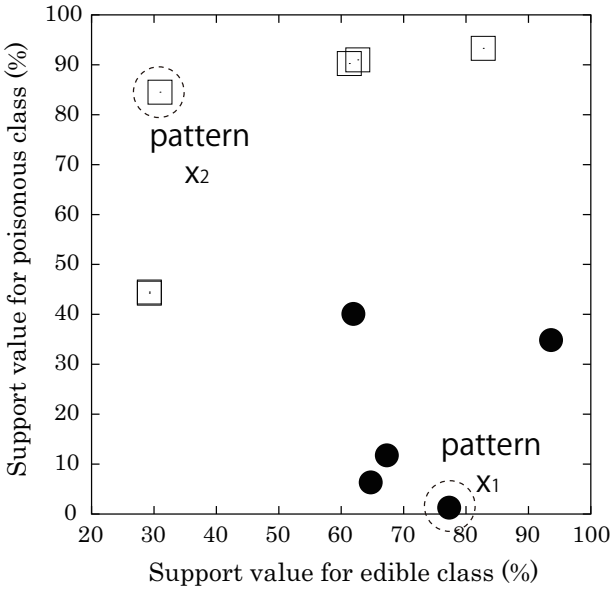

Figure. 9: Contrast patterns for each class in Case 95

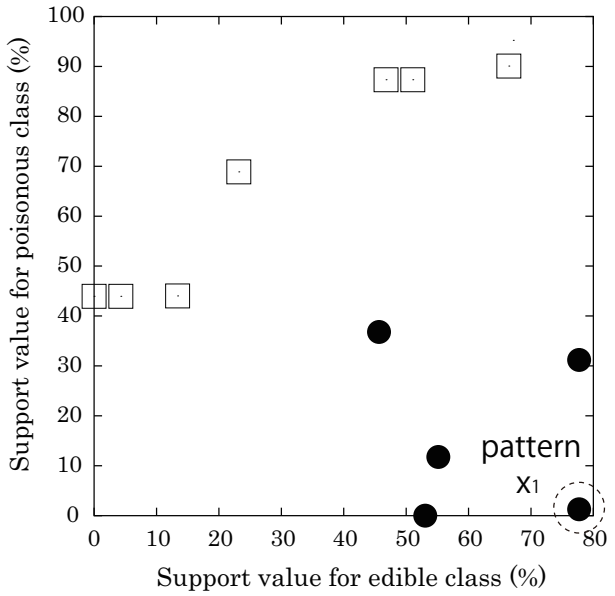

Figure. 10: Contrast patterns for each class in Case 33

to only 11 patterns, and the proposed model performs better than the existing model, which uses many more contrast patterns. Although it is the result only for one case, we guess that it is a valuable result to show the reduction of selected patterns.

V. Conclusion and future works

In this study, we improved CACP by adding GRASP, and we showed the computational results that prove we can use fewer contrast patterns while maintaining the accuracy level. Our method needs to be proven with other data, but we can showed that it performs well with mushroom data and that it can remove more contrast patterns.
Through our experiments, we can see that it is not easy to make the best model. Future works we have to do are mainly two points:

1. Identifying neccesary contrast patterns
   Although it is not easy which pattern is more important, the location of the contrast patterns on the two-dimensional space of support values for each class seem to be important. We are presently unable to find these criteria, but we would like to make clear in a future work. By using the criteria, we might improve the method to find better combination of the patterns.

2. Selecting better model
   We can run our method many times on training data, and we can get many candidate solutions shown in figure 7. However, it is not easy to select better model from the solutions. Although it is important to select the model which have better accuracy for test data, the models which have better accuracy for training data are not always better for test data. For example, in figure 7, better model for training data is most right one, however it has poor accuracy for test data. The reason is not clear, but it may be a kind of over fitting. It is not so fatal problem for this data, because the models have acceptable accuracy. If we can select better one, we can enhance our method from the view point of implementation.

Applying other data, we would like to improve above points and propose more sophisticated method. Especially, for real business data, we can expect to construct an effective classification model, because we need small number of essential patterns.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 25350343. And some computational codes provided by http://research.nii.ac.jp/∼uno/index.html and http://www.nysol.jp/ are used in our experiment.

References

[Bache and Lichman, 2013] Bache, K. and Lichman, M. (2013). Uci machine learning repository.

[Bay and Pazzani, 1999] Bay, S. D. and Pazzani, M. J. (1999). Detecting change in categorical data: Mining contrast sets. In *In Proceedings of the Fifth International*

Conference on Knowledge Discovery and Data Mining, pages 302–306. ACM Press.

[Dong et al., 1999] Dong, G., Zhang, X., Wong, L., and Li, J. (1999). Caep: Classification by aggregating emerging patterns. In Arikawa, S. and Furukawa, K., editors, *Discovery Science*, volume 1721 of *Lecture Notes in Computer Science*, pages 30–42. Springer Berlin Heidelberg.

[Feo and Resende, 1995] Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133.

[Morita and Mao, 2013] Morita, H. and Mao, N. (2013). Classification model using contrast patterns. In *Proceedings of the 15th International Conference on Enterprise Information Systems*, volume 1, pages 336–341.

[Resende and Ribeiro, 2003] Resende, M. G. C. and Ribeiro, C. C. (2003). Greedy randomized adaptive search procedures. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publisher.