Approximation of Optimal Two-Dimensional Association Rules for Categorical Attributes Using Semidefinite Programming*

Katsuki Fujisawa[1], Yukinobu Hamuro[2], Naoki Katoh[1], Takeshi Tokuyama[3], and Katsutoshi Yada[2]

[1] Department of Architecture and Architectural Systems, Kyoto University
Kyoto, Japan, email: {katsuki,naoki}@archi.kyoto-u.ac.jp
[2] Department of Business Administration, Osaka Industrial University
Daito, Osaka, Japan email: {hamuro,yada}@adm.osaka-sandai.ac.jp
[3] Tokyo Research Laboratory, IBM Japan
Yamato, Japan email: ttoku@trl.ibm.co.jp

Abstract. We consider the problem of finding two-dimensional association rules for categorical attributes. Suppose we have two conditional attributes A and B both of whose domains are categorical, and one binary target attribute whose domain is {"positive", " negative"}. We want to split the Cartesian product of domains of A and B into two subsets so that a certain objective function is optimized, i.e., we want to find a good segmentation of the domains of A and B. We consider in this paper the objective function that maximizes the confidence under the constraint of the upper bound of the support size. We first prove that the problem is NP-hard, and then propose an approximation algorithm based on semidefinite programming. In order to evaluate the effectiveness and efficiency of the proposed algorithm, we carry out computational experiments for problem instances generated by real sales data consisting of attributes whose domain size is a few hundreds at maximum. Approximation ratios of the solutions obtained measured by comparing solutions for semidefinite programming relaxation range from 76% to 95%. It is observed that the performance of generated association rules are significantly superior to that of one-dimensional rules.

1 Introduction

In recent years, data mining has made it possible to discover valuable rules by analyzing huge databases. Efficient algorithms for finding association rules have been proposed [1, 9, 10, 20, 24], and classification and regression trees that use these rules as branching tests have been extensively studied [19, 21, 22].

One of important application fields of data mining is marketing. In particular, we are interested in developing an effective strategy of direct mail distribution

* Research of this paper is partly supported by the Grant-in-Aid for Scientific Research on Priority Areas (A) by the Ministry of Education, Science, Sports and Culture of Japan.

based on customer purchase data accumulated in databases. Our research stems from the following real problem related to direct mail distribution. *Kao*, a leading manufacturer of household products in Japan, developed a new brand *Lavenus* in 1996 which covers several different categories of products ranging from shampoo & conditioners, hair care products, hair dye and so on. But, for the first one year, it was not well recognized by customers. Kao then decided to start a joint sales promotion with *Pharma*, which is a drugstore chain in Japan that has approximately three million members (Pharma has been maintaining detailed data of customers' purchase data for more than ten years. See [13] for data mining activities of Pharma). The sales promotion was done by sending direct mails to potential users of *Lavenus*. In order to establish an effective direct mail plan, it is crucial to select customers who have high potential to buy Lavenus in the near future.

For this purpose, Pharma investigated purchase data of Lavenus users, i.e., customers who belong to Pharma's member club and had already bought Lavenus, and identified commodities that are sold well for them. Under the hypothesis that those who frequently buy these commodities but have not yet purchased Lavenus are possibly likely to become Lavenus users in near future, Kao sent direct mails with free sample coupon to such customers. Kao and Pharma observed the effectiveness of this sales promotion. However, this result raises the following question: What is the best strategy to find customers to be targeted for direct mail promotion? This question motivates our study.

In view of this motivation, hoping that more refined rules may possibly find customers that have higher response rate to direct mails, we study the problem of finding two-dimensional association rules for categorical attributes. Association rules that classify a target attribute will provide us with valuable information which may in turn helps understand relationships between conditional and target attributes. Association rules are used to derive good decision trees. As pointed out by Kearns and Mansour [14], the recent popularity of decision trees such as C4.5 by Quinlan [22] is due to their simplicity and efficiency and one of the advantage of using decision trees is potential interpretability to humans.

One-dimensional association rules for categorical attributes can be efficiently obtained [20]. On the other hand, as will be shown in this paper, finding two-dimensional association rules for categorical attributes is NP-hard. Nevertheless we shall develop a practically efficient approximation algorithm for obtaining two-dimensional association rules for categorical attributes. One of the advantages of two-dimensional association rules over one-dimensional ones is that two-dimensional rules usually induce a decision tree of smaller size that has a higher classification ability.

We assume that a database consists of only categorical attributes. Let $\mathcal{R}$ be a database relation. We treat one special attribute as a *target attribute*. Other attributes are called *conditional attributes*. We assume in this paper that the domain of a target attribute is $\{0, 1\}$, i.e., 1 means "positive" response and 0 "negative". Among conditional attributes, we focus on two particular attributes A and B. Let $\mathrm{dom}(A)$ and $\mathrm{dom}(B)$ denote the domain of A and B, respectively.

Let $n_A = |\mathrm{dom}(A)|$ and $n_B = |\mathrm{dom}(B)|$, and let $U \subseteq \mathrm{dom}(A)$, $V \subseteq \mathrm{dom}(B)$. For notational convenience, let $S = U \times V$ and $\bar{S} = \mathrm{dom}(A) \times \mathrm{dom}(B) - (U \times V)$, where $\mathrm{dom}(A) \times \mathrm{dom}(B)$ denotes the Cartesian product of $\mathrm{dom}(A)$ and $\mathrm{dom}(B)$. We then split $\mathrm{dom}(A) \times \mathrm{dom}(B)$ into $(S, \bar{S})$. Ideally, we want to find S for which all records $t \in \mathcal{R}$ with $(t[A], t[B]) \in S$ take value '1' in a target attribute, while other records $t \in \mathcal{R}$ with $(t[A], t[B]) \in \bar{S}$ take value '0'. Since such segmentation is impossible in general, we introduce a certain objective function $f(S, \bar{S})$ that evaluates the goodness of the segmentation.

We consider in this paper the problem that maximizes the confidence under the constraint that the support does not exceed a given threshold. We transform the problem into the one that finds a dense subgraph on weighted bipartite graphs appropriately defined. We first prove its NP-hardness by reduction from a *balanced complete bipartite subgraph* problem which is known to be NP-complete (see [11, 12]). We shall then focus on an approximation algorithm. We propose in this paper an approximation algorithm based on a semidefinite programming (SDP) relaxation. The idea of relaxation is similar to the one by Srivastav and Wolf [23] for the *densest subgraph* problem on general graphs. The densest subgraph problem has recently been studied by several researchers. For the case where weights satisfy the triangle inequality, Arora et al. [2] proposed a PTAS (polynomial-time approximation scheme) for $k = \Omega(n)$, where k is a problem parameter that constrains the lower bound on the vertex size of subgraphs we seek for. For general case, only $\tilde{O}(n^{0.3885\cdot})$ approximation ratio is known for general k [16]. For $k = \Theta(n)$, a few papers presented algorithms constant approximation ratios [3, 23]. After a solution for SDP relaxation is obtained, the conventional approach obtains a rounded solution based on random hyperplane cutting. On the other hand, we introduce a refined rounding technique by making use of the special structure of bipartite graphs. Although we have not yet obtained an improved approximation ratio for our problem, we have implemented the proposed algorithm and carried out computational experiments to see its effectiveness and efficiency. In our experiments, we have employed SDPA which is a software developed by one of the authors [7] for semidefinite programming problems. Although SDP relaxation is known to be powerful in approximately solving densest subgraph problems, there seems to be no report on computational experiments, as far as the authors know. Thus, this paper seems to be the first to report computational results for SDP-based approximation algorithms for such problems although our algorithm is limited to bipartite graphs.

Problem instances we have solved have sizes (i.e., $n_A \times n_B$) ranging from 1,600 to 100,000, all of which are obtained through Pharma from real sales data related to Lavenus sales promotion. We observe that the proposed algorithm efficiently produces good approximate solutions in general for both small and large problem instances. In fact, the average of ratios of the objective value of the obtained solutions to that of SDP solutions exceeds 85%. We also observe that the obtained two-dimensional association rules outperform one-dimensional rules in solution quality. Therefore, we believe that the proposed approach will

be effective for various data mining applications that deal with categorical attributes.

2 Problem Formulation

For simplicity, we assume $\mathrm{dom}(A) = \{1, 2, \ldots, n_A\}, \mathrm{dom}(B) = \{1, 2, \ldots, n_B\}$. Let $n = n_A + n_B$. Let s_{ij} denote the number of records t such that $t[A] = i$ and $t[B] = j$. Among such s_{ij} records, let h_{ij} denote the number of records such that $t[C] = 1$ (i.e., the value of a target attribute C is *positive*), and let $\bar{h}_{ij} = s_{ij} - h_{ij}$. For $R \subset \mathrm{dom}(A) \times \mathrm{dom}(B)$, let

$$s(R) = \sum_{(i,j)\in R} s_{ij}, h(R) = \sum_{(i,j)\in R} h_{ij}.$$

Suppose we are a given a two-dimensional association rule r defined over conditional attributes A and B and a target attribute C. Let S denote the set of $(i, j) \in \mathrm{dom}(A) \times \mathrm{dom}(B)$ such that a tuple t with $t[A] = i, t[B] = j$ satisfies the condition of rule r. Then the rule r can be identified with S. Then $s(S)$, the number of tuples satisfying the condition of r, is called the *support* of S.

$h(S)$, the number of tuples t that satisfies the condition of r as well as a target condition (i.e., $r[C] = 1$), is called the *hit* of r. The ratio $h(S)/s(S)$ is called the confidence of S, or *conf*(S).

We are interested in a rule r such that the corresponding subset S of $\mathrm{dom}(A) \times \mathrm{dom}(B)$ takes the form of $S = U \times V$ for some $U \subset \mathrm{dom}(A)$ and $V \subset \mathrm{dom}(B)$.

The problem we consider in this paper is to maximize $h(S)$ under the constraint of $h(S) \leq M$, and is formulated as follows.

$$\begin{aligned} P: \text{maximize} \quad & h(S) && (1)\\ \text{subject to} \quad & S = U \times V, U \subset \mathrm{dom}(A), V \subset \mathrm{dom}(B) && (2)\\ & s(S) \leq M, && (3) \end{aligned}$$

where M is a given positive constant. The practical role of the constraint (3) in terms of our application of direct mail distribution is to control the number of customers targeted for direct mail.

Lemma 1. *Problem P is NP-complete.*

Proof. Reduction is done from problem *balanced complete bipartite subgraph* which is known to be NP-complete [11, 12]. Given a bipartite graph $G = (U, V, E)$ and a positive integer m, it asks whether the graph contains a subgraph $K_{m,m}$ in G. Letting m be a prime integer satisfying $m^2 > n$, we construct an instance of problem P as follows. Letting $\mathrm{dom}(A) = U$ and $\mathrm{dom}(B) = V$, define $h_{ij} = 1$ if $e = (i, j) \in E$ for $i \in U, j \in V$ and $h_{ij} = 0$ otherwise. In addition, define $s_{ij} = 1$ for all i, j with $i \in U, j \in V$. We set $M = m^2$. Then, it is easy to see that the instance has a solution of objective value $M = m^2$ (i.e., confidence of one) if and only if the bipartite graph has a subgraph $K_{m,m}$.

From this lemma, we then focus on an approximation algorithm in the next section.

3 Approximation Algorithm

The problem of finding a densest subgraph in general graphs has been studied by several authors as mentioned in Section 1, and approximation algorithms have been proposed. Up to now, neither constant- nor logarithmic-approximation algorithms have been proposed. Since we are focusing on our attention to bipartite graphs, it may be possible to develop better approximation algorithms. Although we have not yet succeeded yet in such an attempt, we formulate the problem P as a semidefinite programming problem in a standard way used in [23], and we shall see its approximation ability through computational experiments.

In the formulation as SDP, we introduce an integer variable x_i for each $i \in \mathrm{dom}(A) = \{1, 2, \ldots, n_A\}$ and a variable y_j for each $j \in \mathrm{dom}(B) = \{1, 2, \ldots, n_B\}$. We interpret x_i as $x_i = 1$ if $i \in U$ and $x_i = -1$ otherwise. Similarly, y_j is interpreted as $y_j = 1$ if $j \in V$ and $y_j = -1$ otherwise.

Then the problem P can be rewritten as follows:

$$\begin{array}{rll} P: & \text{maximize} & \frac{1}{4}\sum_{i=1}^{n_A}\sum_{j=1}^{n_B} h_{ij}(x_0+x_i)(x_0+y_j) \\ & \text{subject to} & \frac{1}{4}\sum_{i=1}^{n_A}\sum_{j=1}^{n_B} s_{ij}(x_0+x_i)(x_0+y_j) \le M, \\ & & x_0 = 1. \end{array} \tag{4}$$

For the ease of exposition, by letting $x = (x_0, x_1, \ldots, x_{n_A})$ and $y = (y_1, y_2, \ldots, y_{n_B})$, we introduce an $(n+1)$-dimensional vector $z = (x, y)$ such that $z_i = x_i$ for i with $0 \le i \le n_A$ and $z_i = y_{i-n_A}$ for i with $n_A + 1 \le i \le n_A + n_B (= n)$. In addition, we define

$$h'_{ij} = \begin{cases} h_{i,j-n_A}/2 & \text{for } 1 \le i \le n_A \text{ and } n_A+1 \le j \le n_A+n_B, \\ h_{i-n_A,j}/2 & \text{for } n_A+1 \le i \le n_A+n_B \text{ and } 1 \le j \le n_A, \\ 0 & \text{otherwise.} \end{cases}$$

We then have the following formulation equivalent to (4).

$$\begin{array}{rll} P: & \text{maximize} & \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} h'_{ij}(z_0+z_i)(z_0+z_j) \\ & \text{subject to} & \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij}(z_0+z_i)(z_0+z_j) \le M, \\ & & z_0 = 1. \end{array}$$

In the same manner as is taken in existing semidefinite programming relaxations, we relax the integrality constraint and allow the variables to be vectors in the unit sphere in R^{n+1}. Letting B_1 be the unit sphere in R^{n+1}, the relaxation problem can be written as follows:

$$\begin{array}{rll} SDP_1: & \text{maximize} & \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} h'_{ij}(z_0+z_i)\cdot(z_0+z_j) \\ & \text{subject to} & \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij}(z_0+z_i)\cdot(z_0+z_j) \le M, \\ & & z_0 = (1, 0, 0, \ldots, 0). \end{array} \tag{5}$$

Introducing the variable v_{ij} with $v_{ij} = x_i \cdot x_j$ the above problem can be rewritten as follows:

$$\begin{array}{rll} SDP_2: & \text{maximize} & \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} h'_{ij}(1+v_{0i}+v_{0j}+v_{ij}) \\ & \text{subject to} & \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij}(1+v_{0i}+v_{0j}+v_{ij}) \le M, \\ & & v_{ii} = 1 \text{ for } i = 0, 1, \ldots, n, \\ & & Y = \{v_{ij}\}: \text{ symmetric and positive semidefinite.} \end{array} \tag{6}$$

This problem can be solved within an additive error δ of the optimum in time polynomial in the size of the input and $\log(1/\delta)$ by interior point algorithms. After obtaining an optimal solution v_{ij}^* for SDP_2, we can obtain an optimal solution z_i^* of SDP_1 by a Cholesky decomposition. We then round each unit vector z_i^* to $+1$ or -1. In the conventional method, this is done by choosing a random unit vector u on B_1 and by rounding z_i^* to 1 or -1 depending on the sign of the inner product of u and z_i^*. Let $\hat{z}$ denote the rounded solution so obtained.

Among several softwares for SDPs that are currently available, we use SDPA (Semi-Definite Programming Algorithm) [7] in our implementation. SDPA is a C++ implementation of a Mehrotra-type primal-dual predictor-corrector interior-point method [15, 18] for solving the standard form of SDP. The SDPA incorporates data structures for handling sparse matrices and an efficient method proposed by Fujisawa *et al.* [8] for computing search directions for problems with large sparse matrices.

In order to obtain reasonably good solutions from SDP solutions, we implement the following two algorithms.

Algorithm 1: Instead of using the conventional randomized rounding scheme, the algorithm uses a refined way to round an SDP solution. We do not choose a random unit vector, but instead we try all vectors z_k^* with $1 \leq k \leq n_A$. In fact, from our experimental results, it is worthwhile to try many different vectors in order to obtain better approximate solutions for P. Namely, for a fixed z_k^* with $1 \leq k \leq n_A$ (we call it a basis vector), we round z_i^* to 1 or -1 for other $i \neq k$ with $1 \leq i \leq n_A$, depending on the sign of the inner product of z_k^* and z_i^*. We round z_k^* to 1. Let x_i^k for i with $1 \leq i \leq n_A$ be the rounded solution so obtained.

For each rounded solution $\{x_i^k \mid 1 \leq i \leq n_A\}$ with $1 \leq k \leq n_A$, we obtain a rounded solution $\{y_j^k \mid 1 \leq j \leq n_B\}$ from z_i^* with $n_A + 1 \leq j \leq n$ based on the following lemma.

Lemma 2. *For problem P, when a rounded solution $\{x_i^k \mid 1 \leq i \leq n_A\}$ with $k = 1, 2, \ldots, n_A$ is given, the optimal assignment of $\{y_j^k \mid 1 \leq j \leq n_B\}$ can be obtained by solving a knapsack problem with a single constraint.*

Proof. For problem P, introducing a new variable $y'_j = (y_0 + y_j)/2$ which takes 0 or 1, Problem P becomes an ordinary knapsack problem.

Based on this lemma, in order to solve P, we solve n_A distinct knapsack problems because we try n_A distinct basis vectors. Thus, we obtain n_A feasible solutions of P. In addition, by exchanging the role of x and y, we do the above task in the same way to get n_B feasible solutions as well. Thus, we obtain n feasible solutions of P in total. The algorithm outputs the one that maximizes the objective function of P. In our implementation, we do not exactly solve knapsack problems. Instead we use the heuristic developed by Kubo and Fujisawa [17] who have shown that the heuristic is fast and produces very good solutions.

Algorithm 2: In order to obtain a feasible solution from an SDP solution z_i^*, we adopt the conventional rounding method explained above. The solution $\hat{z}$ so

obtained may violated the constraint of (3) or it has enough room for improvement ((3) is satisfied and some $\hat{z}_i$ can be changed from -1 to 1 to increase the objective value without violating (3)).

(1) If $\hat{z}$ is feasible, we check whether changing $\hat{z}_i$ from -1 to $+1$ still preserves the constraint (3). Let I be the set of such i. For each $i \in I$, let δ_i^s and δ_i^h be the increase of the objective function of P and the left-hand side of the constraint of (3) according to the change of $\hat{z}_i$ from -1 to $+1$. The algorithm chooses $i^* \in I$ with the largest δ_i^h/δ_i^s and sets $\hat{z}_{i^*} = 1$. Deleting i^* from I, we repeat this process as long as the current solution is feasible. This is a greedy-type algorithm.

The solution obtained in this manner is then further improved in a manner similar to Algorithm 1. Namely, for a given solution $\hat{z}$, we fix its x-part (resp. y-part) while we treat the y-part (resp. x-part) as free variables in order to maximize the objective function of P. This problem is again a knapsack problem, and is also solved by the heuristic developed by Kubo and Fujisawa [17].

(2) If $\hat{z}$ is infeasible, we shall change some $\hat{z}_i$ from $+1$ to -1. This change is also made in a greedy manner as in (1). We also improve the obtained solution by applying the heuristic by [17] after formulating the problem as a knapsack problem.

In the implementation of Algorithm 2, we generate random unit vectors u on B_1 as many times as in Algorithm 1. Among the solutions obtained above, we choose the best one.

4 Computational Experiments

In order to see the effectiveness and efficiency of the proposed two algorithms, we have performed computational experiments. Problem instances are generated from sales data related to Lavenus sales promotion mentioned in Section 1. We have chosen in our experiments six stores different from those for which Pharma performed experiments, and used sales data for three months starting from March of 1997. We have focused on customers who belongs to Pharma's member club and visited those stores at least three times during those three months. We concentrate on the same 660 brands correlated to Lavenus products that Pharma identified. Those brands are classified into seven classes as follows:

Table 1. Seven categories of commodities

class #	description	# of brands
1	medical and pharmaceutical products	187
2	medical appliances and equipments	56
3	health foods	38
4	miscellaneous daily goods	252
5	cosmetics	76
6	baby articles	43
7	others	8

We consider that these seven classes are different attributes. The domain of an attribute is a set of brands that fall into the corresponding class. Let s_{ij} stand for the number of customers who bought brands i and j. Among such customers let h_{ij} denote the number of customers who also bought Lavenus products. We generate problem instances by choosing two distinct classes. We also generate problem instances of larger size by grouping seven classes into two disjoint sets. In this case, each set is regarded as a single attribute. The problem instances we generated for computational experiments are listed in Table 2. For each problem instance we have tested two or three different values of M.

In order to compare the performance of two-dimensional rules generated by our algorithms with one-dimensional ones, we have considered the following problems Q_A and Q_B. Let

$$s_{i\cdot} = \sum_{j=1}^{n_B} s_{ij}, s_{\cdot j} = \sum_{i=1}^{n_A} s_{ij}.$$

$h_{i\cdot}$ and $h_{\cdot j}$ are similarly defined.

$$Q_A: \text{ maximize } \{\sum_{i\in U} h_{i\cdot} \mid U \subset \mathrm{dom}(A), \sum_{i\in U} s_{i\cdot} \leq M\}. \tag{7}$$

$$Q_B: \text{ maximize } \{\sum_{j\in V} h_{\cdot j} \mid V \subset \mathrm{dom}(B), \sum_{i\in V} s_{\cdot j} \leq M\}. \tag{8}$$

Both problems are knapsack problems and are solved by applying the algorithm of [17]. We then choose the better one between the obtained two solutions.

Computational results for Problem P are shown in Table 3. Experiments have been carried out on DEC Alpha 21164 (600MHz). The first column indicates the problem No. and the problem size ($n_A \times n_B$). The second column indicates the overall average of the confidence, i.e., $\sum h_{ij} / \sum s_{ij}$ where the sum is taken over all pairs of brands (i, j) that belong to $\mathrm{dom}(A) \times \mathrm{dom}(B)$. The third column shows the ratio of M to $\sum s_{ij}$, where the sum is taken over all pairs of brands that belong to $\mathrm{dom}(A) \times \mathrm{dom}(B)$. The fourth column represents the ratio of the objective value for an approximate solution to the upper bound of the optimal objective value obtained by SDP. Here the approximate solution indicates the better one between those obtained by Algorithms 1 and 2. The fifth column indicates CPU time spent by SDPA software. The sixth, the seventh and the eighth columns indicate the confidence of two-dimensional rules derived by Algorithms 1 and 2, and of one-dimensional rule, respectively. The asterisk * indicates that it exhibits the better performance than the one without *. We see from the table that the proposed algorithms produce good approximate solutions in a reasonable amount of time. The time spent for obtaining one rounded solution from SDP solution by both Algorithms 1 and 2 ranges from 1 to 13 seconds depending the problem size (most of the time is spent for solving knapsack problems). Since we obtain n different candidate solutions for both Algorithms 1 and 2, the time required for obtaining the best approximate solution from an SDP solution is, on the average, three times larger than that for obtaining an SDP solution. Notice that for the cases of $M/\sum s_{ij} = 1/8$, we see a significant difference between

the confidence of two-dimensional association rules and that of one-dimensional rules.

As for comparison of Algorithms 1 and 2, it is observed that Algorithm 2 produces better solutions for many cases while the difference of performances between Algorithms 1 and 2 gets closer as $\sum h_{ij}/\sum s_{ij}$ becomes smaller. In particular, for problems 8 and 9 with $M/\sum s_{ij} = 1/16$, Algorithm 1 exhibits better performance.

Notice that for two attributes A and B in our problem instance, $s(R)$ for $R = U \times V$ with $U \subset \mathrm{dom}(A)$ and $V \subset \mathrm{dom}(B)$ does not represent, in general, the number of customers who bought at least one brand in U and at least one brand in V because a single customer may have bought many pairs of brands $(i, j) \in R$ and such a customer is counted the same many times in $s(R)$. Therefore, in order to evaluate the practical effectiveness of the rules generated, we need to calculate the number of customers that satisfy the condition of the association rule obtained for each problem instance, and the number of customers that satisfy the target condition (i.e., bought Lavenus products). We call the former number c-support, and the latter c-hit ("c-" is attached in order to distinguish the support and the hit defined in Section 2). We call the ratio of c-hit to c-support a *hit ratio* of the rule.

We have computed the c-supports, c-hits and hit ratios of the four two-dimensional rules obtained by our algorithm for the last two problem instances, i.e., the rules found in the second last problem instance for $M/\sum s_{ij} = 1/8$ and $1/4$ (denoted by Rules 1 and 2, respectively), and the rules found in the last problem instance for $M/\sum s_{ij} = 1/8$ and $1/4$ (denoted by Rules 3 and 4, respectively). Results are summarized in Table 4. In the second last and last problem instances, numbers of customers that bought at least one brand from A and at least one brand from B are 4822 and 5190, respectively, and the numbers of Lavenus users among them are 184 (3.8%) and 198 (3.8%). Therefore, as seen from Table 3, our algorithm found a good customer segmentation.

As mentioned in Section 1, Kao and Pharma, in their experiments, adopted the rule to select the customers for direct mail distribution such that they bought at least three distinct brands for a certain time period from among 660 ones that Pharma identified. Let us call the rule *at-least-3 rule*. We can generalize this rule to *at-least-k rule*. For comparison, we consider k with $3 \leq k \leq 8$. We have computed the c-supports, c-hits and hit ratios for such rules. The results are summarized in Table 3. We see from the table that Rules 2 and 4 are a bit worse than at-least-6 or 7 rule, while Rules 1 and 3 are better than at-least-8 rule.

5 Conclusion

We have proposed an approximation algorithm based on SDP relaxation for finding optimal two-dimensional association rules for categorical attributes. From computational experiments, it was observed that the proposed algorithm finds a good segmentation in a reasonable amount of time. We finally observed how

effective the rules obtained are in terms of customer segmentation by applying the algorithm to real sales data.

There are still several tasks remaining for future research.

(1) First, we want to improve an approximation ratio of problem P, i.e., the densest subgraph problem for bipartite graphs. The current best ratio is the same as the one obtained for general graphs.
(2) The proposed algorithm can be extended to the problems with entropy gain or interclass maximization in a straightforward manner. However, it requires more computation time. So, we need to further improve the practical efficiency of the algorithm.
(3) We would like to conduct an experiment to see the robustness of the obtained two-dimensional association rules in terms of the ability of future forecasting.
(4) From the viewpoint of promotion sales through direct mail distribution, we would like to carry out the experiments to see the difference between response rates of customers selected by the rules obtained in our algorithm and of those by other rules.

Acknowledgments

Computational experiments have been carried out with assistance of Mr. Yoshihiro Kanno, who is a graduate student in Department of Architecture and Architectural Systems, Kyoto University. His efforts are gratefully acknowledged. We also express our gratitude to Pharma G&G Corp. for kindly providing us with their sales data.

References

1. R. Agrawal, T. Imielinski, and A. Swami, Mining association rules between sets of items in large databases, *Proc. of the ACM SIGMOD Conference on Management of Data*, 207-216, 1995.
2. S.Arora, D. Karger, and M.Karpinski, Polynomial time approximation schemes for dense instances of NP-hard problems, *Proc. 27th ACM Symposium on Theory of Computing*, 284-293, 1995.
3. Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama, Greedily finding a dense subgraph, *Proc. of the 5th Scandinavian Workshop on Algorithm Theory (SWAT)*, LNCS 1097, 136-148, Springer, 1996.
4. T. Asano, D. Chen, N. Katoh, and T. Tokuyama, Polynomial-time solutions to image segmentation problems, *Proc. of 7th ACM/SIAM Symposium on Discrete Algorithms*, pp. 104-113, 1996.
5. U. Feige and M. Seltser, On the densest k-subgraph problems, Technical Report, Dept. of Applied Mathematics and Computer Science, The Weizmann Institute, September, 1997.
6. A. Frieze and M. Jerrum, Improved algorithms for Max K-cut and Max bisection, *Algorithmica*, 18 (1997), 67-81.
7. K. Fujisawa, M. Kojima and K. Nakata, SDPA (Semidefinite Programming Algorithm) –User's Manual–, Tech. Report B-308, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Japan, 1998.

8. K. Fujisawa, M. Kojima and K. Nakata, Exploiting Sparsity in Primal-Dual Interior-Point Methods for Semidefinite Programming, Mathematical. Programming, Vol. 79, pp. 235-253, 1997.
9. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Constructing efficient decision trees by using optimized association rules. *Proc. of 22nd VLDB Conference*, 146-155, 1996.
10. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. "Data Mining Using Two-Dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization", Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Canada, pages 13-23, June 1996, ACM Press.
11. M.R. Garey and D.S. Johnson, *Computers and Intractability: A guide to the Theory of NP-completeness*, Freeman, 1979.
12. D.S. Johnson, The NP-completeness column: An ongoing guide, *Journal of Algorithms*, Vol.8 (1984), 438-448.
13. Y. Hamuro, N. Katoh, Y. Matsuda and K. Yada, Mining Pharmacy Data Helps to Make Profits, *Data Mining and Knowledge Discovery*. Vol.2, No.4, (1998), pp.391-398.
14. M. Kearns and Y. Mansour, On the boosting ability of top-down decision tree learning algorithms, *Journal of Computer and System Sciences*, 58 (1999) 109-128.
15. M. Kojima, S. Shindoh and S. Hara, Interior-point methods for the monotone semidefinite linear complementarity problems, SIAM Journal on Optimization, Vol. 7, pp. 86-125, 1997.
16. G. Kortsarz and D. Peleg, On choosing a dense subgraph, *Proc. of 34th IEEE Symp. on Foundations of Computer Sci.,* 692-701, 1993.
17. M. Kubo and K. Fujisawa, The Hierarchical Building Block Method and the Controlled Intensification and Diversification Scheme – Two New Frameworks of Metaheuristics –, unpublished manuscript, 1999.
18. S. Mehrotra, On the implementation of a primal-dual interior point method, SIAM Journal on Optimization, Vol 2, pp. 575–601, 1992.
19. Y. Morimoto, T. Fukuda, S. Morishita and T. Tokuyama, Implementation and evaluation of decision trees with range and region splitting, *Constraint*, 2(3/4), (1997), 163-189.
20. Y. Morimoto, T. Fukuda, H. Matsuzawa, K. Yoda and T. Tokuyama, "Algorithms for Mining Association Rules for Binary Segmentations of Huge Categorical Databases", Proceedings of VLDB 98, New York, USA, August 1998.
21. J.R. Quinlan, Induction of decision trees, *Machine Learning*, 1 (1986), 81-106.
22. J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.
23. A. Srivastav and K. Wolf, Finding dense subgraphs with semidefinite programming, *Approximation Algorithms for Combinatorial Optimization*, LNCS 1444, 181-191, Springer, 1998.
24. K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Computing Optimized Rectilinear Regions for Association Rules, Proceedings of Knowledge Discovery and Data Mining 1997 (KDD '97), AAAI, Newport Beach, USA, August 1997, AAAI Press.

Table 2. Problem instances generated for numerical experiments

problem #	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
classes of A	2	3	5	2	5	5	1	1,2	1,5,6
classes of B	1	4	1	4	4	1,2,3	4	3,4,5,6,7	2,3,4,7

Table 3. Computational results for problem instances tested

problem # & size $n_A \times n_B$	ave. confidence ($\sum h_{ij} / \sum s_{ij}$)	$\frac{M}{\sum s_{ij}}$	approx. ratio	time (sec.)	confidence (2-D rule) Algo. 1	confidence (2-D rule) Algo. 2	confidence (1-D rule)
(1)	6.3%	1/8	88%	27.5	*25.3%	23.0%	19.9%
56×187	(132/2085)	1/4	89%	25.7	15.5%	*16.5%	15.5%
(2)	7.4%	1/8	82%	47.7	36.9%	*38.2%	30.0%
38×252	(93/1256)	1/4	91%	45.0	23.9%	*25.5%	21.3%
(3)	7.8%	1/8	83%	39.7	26.3%	*30.0%	24.7%
76×187	(154/1976)	1/4	91%	36.2	21.7%	*21.9%	19.4%
(4)	9.3%	1/8	89%	63.0	*28.7%	*28.7%	23.0%
56×252	(477/5149)	1/4	95%	54.0	20.7%	*20.9%	17.6%
(5)	7.9%	1/8	85%	76.9	22.5%	*22.9%	19.4%
76×252	(491/6194)	1/4	90%	72.7	16.0%	*17.0%	15.1%
(6)	7.3%	1/8	84%	101.1	27.3%	*29.5%	23.9%
76×281	(187/2578)	1/4	91%	87.1	18.8%	*20.8%	18.8%
(7)	6.9%	1/8	90%	211.9	22.3%	*22.5%	20.4%
187×252	(1476/21513)	1/4	91%	184.2	14.4%	*20.8%	14.4%
(8)	6.9%	1/16	77%	739.7	*29.3%	25.6%	24.6%
243×417	(2409/35133)	1/8	86%	848.3	21.0%	*21.2%	19.8%
		1/4	94%	847.3	15.1%	*16.2%	15.1%
(9)	6.4%	1/16	76%	814.8	*27.0%	25.1%	24.2%
306×354	(2545/39470)	1/8	87%	851.7	*20.0%	*20.0%	18.7%
		1/4	95%	757.8	14.4%	*15.2%	14.4%

Table 4. Comparison of two-dimensional rules and conventional rules in terms of c-support, c-hit and hit ratio

rules	Rule 1	Rule 2	Rule 3	Rule 4	at-least-k rules					
					$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
c-support	916	1602	1126	1921	5806	4009	2775	1961	1375	959
c-hit	83	112	95	114	253	214	169	137	103	75
hit ratio	9.1 %	7.0%	8.4%	5.9%	4.4%	5.3%	6.1%	7.0%	7.5%	7.8%