OFFERING PATTERN MINING USING HIGH YIELD PARTITION TREES

Jianying Hu *Aleksandra Mojsilovic*

IBM T.J. Watson Research Center
1101 Kitchawan Road, Route 134, Yorktown Heights, NY 10598
{jyhu, aleksand}@us.ibm.com

ABSTRACT

Despite the wide use of data mining techniques in client segmentation and market analysis applications, so far there have been no algorithms that allow for the discovery of strategically important combinations of products (or offerings) – the ones that have the highest impact on the performance of the company. We present a novel algorithm for analyzing a multi-product environment and identifying strategically important combinations of offerings with respect to a predefined criterion, such as revenue impact, profit impact, inventory turnover etc. In contrast to the traditional association rule and frequent item mining techniques, the goal of the new algorithm is to find segments of data, defined through combinations of products (rules), which satisfy certain conditions as a group. We present a novel algorithm to derive specialized partition threes, called High Yield Partition Trees, which lead to such segments, and investigate different splitting strategies. The algorithm has been tested on real-world data, and achieved very good performance.

1. INTRODUCTION

Today the strategic and marketing decisions of most companies depend on customer segmentation, i.e. understanding the characteristics and behavioral patterns of their customers. As more and more companies diversify their operations and expand the spectrum of their products and services, it is becoming critical to understand cross-cohesion among different products or services (referred to collectively as offerings hence forth) and identify natural groupings that were not expected to exist or have not been addressed in the development phase of each individual offering. In particular, companies are interested in identifying groups of offerings that have the most significant impact on the their bottom line. Such knowledge could guide strategic decisions at the top levels of corporation, optimize the behavior of the entire enterprise by exploiting the linkages between different brands, institute new offerings by bundling the discovered combinations of offerings, and even open up new markets and new opportunities driven by the identified relationships.

This need gives rise to a new opportunity for pattern recognition as a support tool in strategic decision making process. The problem of identifying offering groups that collectively account for a significant portion of revenue (or other quantitative measures such as profit, marketing cost, etc.) is quite different from the problems addressed by the traditional market segmentation techniques, or so called "market basket" analysis techniques, because the latter only attempt to identify offerings that are frequently bought together with no regard to their total revenue contribution.

In this paper we present a novel algorithm for analyzing a multi-product environment and identifying "strategic offerings", (i.e. the combinations of offerings which represent significant amount of companys total revenue) using a new type of binary partition tree called *High Yield Partition Tree*.

2. RELATED WORK

There has been much work on techniques for "market basket" analysis in the data mining community. In particular, much focus has been given to fast algorithms for association rule mining and the related problem, frequent item set mining [1, 2, 3]. The former aims to discover rules such as "30% customers who bought A and B also bought C", while the latter aims to identify patterns such as "20% of all transactions contain A, B and C". The *offering pattern mining* problem we consider is different, since unlike former approaches we conduct "rule-discovery" not only with respect to the individual attributes, but also with respect to the overall criterion for the mined set. We are interested in patterns such as "offerings A, B and C account for 80% of revenue from customers who bought all three". More significantly, we aim to find groups of such patterns that combined account for a significant portion of the total revenue from all clients. Clearly, this is not a mere extension of the market basket analysis problem and an entirely new approach is required.

Binary partition trees have also been extensively used before, particularly in machine learning in the form of regression/classification/decision trees [4, 5], and in image analysis for image segmentation [6]. In both cases, the goal of partitioning is to identify homogeneous segments of data, with respect to some characteristics of individual data elements or image objects (i.e., belonging to the same class, having similar color or texture, etc.). In contrast, in the offering pat-

tern mining problem, the goal of partitioning is to identify segments of data that satisfy certain condition as a group. Furthermore, we are interested in finding partitions that combined satisfy certain criterion. As will be shown later, these different goals lead to very different strategies for both tree growing and pruning.

3. PROBLEM DEFINITION

Let $C = c_1, c_2, \ldots, c_N$ represent a set of clients and $O = o_1, o_2, \cdots, o_M$ a set of offerings. Each client c_i is represented by a vector $R_i = [r_{i,1}, r_{i,2}, \ldots, r_{i,M}]$ where $r_{i,j}$ is the revenue from client c_i on offering o_j. An offering pattern is a set of conditions represented as an integer vector $P = [p_1, p_2, \ldots, p_M]$ where for each component p_i, a value of 1 indicates non-zero revenue from offering o_j, -1 indicates zero revenue, and 0 indicates "don't care". A client c_i is said to satisfy a pattern P if it satisfies the following two conditions:

$$r_{ij} > 0, \forall j | p_j = 1; \tag{1}$$

and

$$r_{ij} = 0, \forall j | p_j = -1. \tag{2}$$

Given a pattern P_k, the subset of C which consists of all clients that satisfy P_k is called the *subscription set* of P_k and represented as $C_k = \{c_{k1}, c_{k2}, \ldots, c_{kn_k}\}$. The *set revenue* of C_k, denoted $R(C_k)$, is simply the sum of all revenues from all clients in the subset. The *in-pattern revenue* of C_k is the sum of revenues corresponding to the 1 components in pattern P_k:

$$R_p(C_k) = \sum_{i=1}^{n_k} \sum_{j=1}^{M} p_{kj} \times r_{ki,j}. \tag{3}$$

A pattern P_k is called a *defining pattern* if it satisfies $R_p(C_k)/R(C_k) \geq \sigma$ where σ is a high ratio (e.g., over 0.8). It is called a *Significant Defining Pattern* (SDP) if it further satisfies: $R(C_k)/R(C) \geq \gamma$ where γ is a lower bound (e.g., 0.01). SDPs are particularly useful in offering portfolio analysis, since each SDP accounts for a significant portion of the total revenue of the represented clients, and can thus be considered to define the common purchasing behavior of this subset of clients. At the same time, the subscription set of each pattern is significant in the sense that they collectively represent a significant portion of total revenue.

Given a client set C and the associated revenue vectors, the goal of offering pattern mining is to identify a small set of SDPs whose combined in-pattern revenue is a significant portion of the total revenue of C. To avoid multiple counting of revenue, it is further required that the patterns result in non-overlapping subscription sets. However, the union of all subscription sets does not have to contain all clients. We shall refer to any set of non-overlapping SDPs as a *pattern set* for short.

For any given data, there likely exist many different pattern sets. In the trivial case, the patterns derived directly from the revenue vectors of clients with significant revenues are themselves SDPs, and together form a pattern set. However, this trivial pattern set is likely very large and not very interesting because it does not provide any insight as to the grouping of either clients or offerings. What we are interested in are pattern sets that include as few patterns as possible while at the same time represent as large a portion of the total revenue as possible. Thus, the quality of a pattern set is measured by two criteria: *yield*, which is defined as the ratio between total in-pattern revenue and total revenue, and *size*, representing the number of patterns in the set. Note that these measures are clearly data dependent and can only be used to compare pattern sets derived from the same data.

The goal of offering pattern mining can now be restated as: for any given size, identify the pattern set of the specified size that has the highest yield. Ideally, we would like to examine all possible pattern sets and compare them using the above two criteria. However, this presents a complex combinatorial problem with a large search space that quickly becomes intractable as M increases. Thus, for offering pattern mining to be of practical value we need to find efficient algorithms that lead to suboptimal and yet useful solutions.

4. HIGH YIELD PARTITION TREE

Because of the non-overlapping property, identifying the SDPs of interest is equivalent to identifying clusters of clients where clients in each cluster all satisfy a particular pattern. One efficient way to represent this problem is to use a binary partition tree, where each tree node represents a particular pattern and includes all of its subscribing clients. In this section we present a novel algorithm to derive specialized partition trees that lead to pattern sets with high yields. We call these trees *High Yield Partition Trees*.

The tree is initialized by assigning all clients to the root node, representing the most general pattern: $[p_i = 0; 1 \leq i \leq M]$. We now pick an offering o_j using a particular *offer selection criterion* (which will be discussed in detail later) and partition the clients into two groups, those with revenue from o_j and those without. The first group is assigned to the left child of the root, representing the pattern $[p_j = 1; p_i = 0 \text{ for } i \neq j]$ and the second group is assigned to the right child of the root, representing the pattern $[p_j = -1; p_i = 0 \text{ for } i \neq j]$. This "splitting" procedure is recursively carried out at each node, producing more and more specific patterns and partitioning the clients into smaller and smaller groups. As the patterns get more specific, the in-pattern revenue ratio $R_p(C_k)/R(C_k)$ becomes larger while the set revenue $R(C_k)$ becomes smaller. At some point, each leaf node in the tree will satisfy at least one of the following two conditions: 1) $R(C_k) < \gamma$, when the node becomes insignificant, and 2) $R_p(C_k)/R(C_k) > \sigma$, when the node be-

comes a defining node.

The recursive procedure stops at a node when the first condition is reached, because the offsprings of an insignificant node are also insignificant and thus not of interest.

The situation is more complex when the second condition is reached. While we could stop here since an SDP has been found, splitting further could potentially produce more specific patterns that lead to higher in-pattern revenue. The strategy we have adopted is as follows. After an offering has been chosen using the selection criterion, the corresponding potential children are examined. If both children are defining and significant, by definition their combined in-pattern revenue is larger than the parent's in-pattern revenue, thus both nodes are created and each is subjected to the recursive splitting procedure again. A child that is either non-defining or insignificant is not created. If only one child is defining and significant, then this child is created only if its in-pattern revenue is larger than the parent's. This strategy insures two properties of the tree. The first property is *continuity*: any path of defining nodes is continuous, i.e., a child of a defining node is also a defining node. The second property is *monotonicity*: the total in-pattern revenue of the child/children of a defining node is always larger than the parent's in-pattern revenue. As will become clear later, both properties are important in the pruning stage. The strategy also makes intuitive sense: we only continue splitting when it results in a larger total in-pattern revenue and thus a higher yield.

When the above recursive splitting procedure is completed, all insignificant leaves are removed. The remaining leaves each represent an SDP and together form a pattern set that has the highest possible yield for the chosen offering selection criterion. However, it may very well contain more patterns than the desired size, thus a pruning procedure is required to "trim" the tree to the desired size.

For pruning purpose, we identify a class of nodes called *fringe nodes*. Each fringe node has one or two associated pruning operations, with each operation reducing the size of the pattern set by one and incurring a loss of total in-pattern revenue.

A defining node is a fringe node if there are exactly two leaves in the subtree rooted at the node. The pruning operation at such a node is to remove all of its offsprings and the incurred loss is the difference between the combined in-pattern revenue of the two leaves minus the in-pattern revenue of the fringe node. The effect of this pruning operation is to replace two SDPs with one ancestor SDP.

A non-defining node is a fringe node if it has at least one leaf child. In this case there is a pruning operation associated with each leaf child, which is simply to remove the child with the incurred in-pattern revenue loss being the in-pattern revenue of the child. The effect of this pruning operation is to remove one SDP directly from the set.

The pruning procedure is carried out iteratively. During each iteration, all fringe nodes and the associated pruning operations are identified. The operation with minimum incurred in-pattern revenue loss is selected and carried out. Finally any newly "exposed" non-defining leaves are removed so that all leaves again represent SDPs. The iteration stops when the desired number of patterns is reached.

5. SPLITTING CRITERIA

We now turn our attention to the selection criterion used in recursive splitting. We have investigated four different strategies for offering selection. These strategies all attempt to compute a "predicted gain" of in-pattern revenue for each offering, and then select the offering with the highest predicted gain. They differ in how the predicted gain is computed. In the following description, the offerings corresponding to 0s in the associated pattern at each node are called *available offerings*, since they are the only ones still "available" for selection.

Yield Greedy (YG). For each available offering, the predicted gain is computed as the accumulated revenue from all clients in the current set on this offering.

YG with one look ahead. For each available offering, perform a hypothetical split on this offering; then select offering at each child using the YG criterion. Add revenues from selected offering to the predicted gain as defined in the first criterion.

Positive correlation. For each available offering, perform a hypothetical split on this offering. For each of the remaining available offerings, compute the ratio between the accumulated revenue on the left side and the accumulated revenue on the right side. Select κ offerings with highest ratio and add the left side accumulated revenues of these offerings to the predicted gain as defined in the first criterion.

Positive and negatived correlation. Starting with predicted gain as defined in criterion 3, further select κ offerings with lowest ratio and add the right side accumulated revenue of these offerings.

The first strategy gives the most "short sighted" estimate using only the offering itself. The other three strategies are different attempts at "looking further" and thus getting potentially better estimates. The second strategy hypothetically performs one more level of split and aggregate the gain from splits at both levels as the estimated gain. Strategies 3 and 4 are inspired by observations from domain experts that certain offerings have a pulling effect (positive correlation) or an inhibiting effect (negative correlation). Thus strategy 3 incorporates positively correlated revenues and strategy 4 incorporates both positively and negatively correlated revenues into the estimated gain. Parameter κ controls the number of correlated revenues to incorporate and was set to 2 in our empirical evaluations.

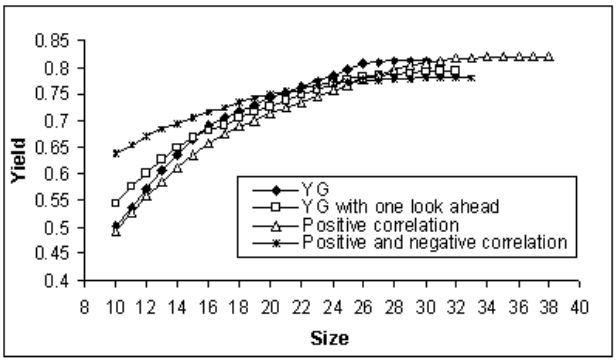

Fig. 1. Yield-size plots for data set 1.

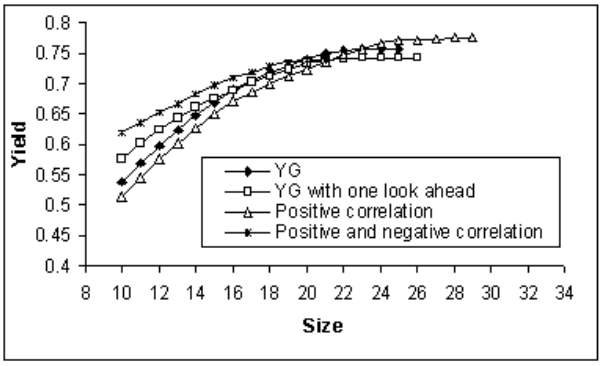

Fig. 2. Yield-size plots for data set 2.

6. EMPIRICAL EVALUATIONS

The high yield partition tree algorithm was evaluated using the purchasing data for IBM corporation. We considered two data sets. The first one contains 2003 revenue from 737 largest clients, broken across 21 products and lines of business (i.e., offerings). The second set contains 2004 revenue from 711 largest clients, broken across the same 21 offerings. For each data set, a tree was built using each selection criterion, then pruned one pattern at a time until there were 10 patterns left. Figures 1 and 2 show the yield vs. size curve for pattern sets derived using the four selection criteria for each data set.

As seen in the plots, the algorithm successfully identifies pattern sets of varying sizes with yield up to 0.82. The four different selection criteria lead to different tradeoffs between yield and size at different points of the curve, and their comparison appears to be more or less consistent in both data sets. When compared to the basic YG criterion, both "YG with one look ahead" and "Positive and negative correlation" provide improved yield at smaller sizes (with the latter providing more significant improvement) but perform worse at larger sizes. On the other hand, the "positive correlation" criterion performs slightly worse than YG at smaller sizes, while providing a small improvement at larger sizes.

Given that the selection criteria have complementary behaviors at different sizes, a reasonable strategy is to create four different trees on a given data set, each using a different criterion, then select the pattern set with the highest yield for each given size. Using this strategy, the algorithm achieves a yield of over 0.6 at 10 patterns and the highest yield of over 0.8 for both data sets.

7. CONCLUSIONS

Despite the wide used of data mining techniques in client segmentation and market analysis applications, so far there have been no algorithms that allow for the discovery of strategically important combinations of products or offerings, i.e. the ones that account for the highest percentage of company revenue (or other business performance metric), or have the highest impact on the company performance. We have presented a novel algorithm for analyzing a multi-product environment and identifying the offering combinations that have the highest impact on companys bottom line. Although we used the revenue coverage as a clustering criterion, the algorithm can be easily adapted to work by maximizing other more complex objective functions. The algorithm has been tested on real-world data with promising results. It is currently being used as a decision support tool by the IBM Market Intelligence to investigate impact and cohesion of different product lines, and architect new market strategies.

8. REFERENCES

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. of the 20th Int'l Conference on Very Large Databases*, Sept. 1994.

[2] W. Wang, J. Yang, and P.S. Yu, "Efficient mining of weighted association rules (WAR)," in *Proc. of SIGKDD 2000*.

[3] F. Tao, F. Murtagh, and M. Farid, "Weighted association rule mining using weighted support and significance framework," in *Proc. of SIGKDD 2003*.

[4] L.Breiman, J.J. Friedman, R.A. Olsen, and C.J. Stone, *Classification and regression trees*, Wadsworth, Belmont, CA, 1984.

[5] J.R. Quinlan, *C4.5: Programs for machine learning*, Morgan Kauffman, 1993.

[6] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation and information retrieval," *IEEE Trans. Image Processing*, pp. 561–575, April 2000.