The Pennsylvania State University

The Graduate School

The Harold and Inge Marcus

Department of Industrial & Manufacturing Engineering

A SPEEDY ALGORITHM FOR DETERMINING PERSONALIZED DISCOUNTS FOR PRODUCTS APPROACHING THEIR END-OF-LIFE STAGE

A Thesis in

Industrial Engineering and Operations Research

by

Keerati Inochanon

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

December 2008

Abstract

Revenue maximizing entities in a supply chains such as suppliers, distributors, and retailers are often concerned with how they would quickly move products out of their inventory while still making profit. In this thesis, we look at a scenario in which a supplier is launching a new product and thus has a need to quickly empty out the inventory for the existing product that will be replaced, under the constraint that the revenue from the sales must at least cover the cost of the items. We present a method to determine an optimal discount for each customer to increase the likelihood that the customer will accept the offer and maximize the revenue. In order to achieve this, past purchase behaviors of customers are examined. Data mining techniques such as sequential pattern mining and clustering are used to mine customer behavior data. The amount of discount to be given for each customer is based on a price model that we created in combination with fuzzy modeling. Based on past purchase behaviors and past discount responses, fuzzy inference rules are created and a knowledge base is formed to make discount decisions. An example of how the algorithm is used with RFM (Recency, Frequency, and Monetary) features are presented. We show that the use of fuzzy modeling provides good results while also maintains unambiguity through its linguistic rules.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my parents for their moral support as well as financial support. Without them, I would not be who I am today and this thesis would not have materialized. They have always stood by me when I needed them the most.

I would like to thank my thesis advisor, Dr. Soundar Kumara for supervising my work on this thesis. His guidance and expertise in artificial intelligence are excellent foundations for the work within the thesis. I appreciate his helps in shaping the direction of my research as well as my personal career goals.

I would like to thank Dr. Jun Shu for his insights throughout my years at Penn State. I thank him for exposing me to the field of operations research. It was conversations with him that sparked my interest in the area.

I would also like to thank all my friends for their support. I would like to thank Noi and Charn for allowing me to bounce ideas off of them. I would like to especially thank Noi for her encouragements and for putting up with me. I would like to thank P'Oak and P'Chum for their advices and helps during the time that I was away from State College. I would like to thank Yu-tai for providing me a roof and delicious meals during my stays.

Chapter 1

Introduction

There are several reasons why it is easier for online stores to implement dynamic pricing policies. In general, online stores have more information about their customers. In addition to transaction data, online stores also have information on where customers have been and what items have been viewed before the customer finally makes a purchase. A page generated online can be easily made to target a specific customer with relevant marketing data. Online retailers can change prices easily and more often [1]. It is even possible for different customers to see different prices for the same product. It is not simple, however, for a brick-and-motar store to develop a similar strategy. Firstly, it takes time to update prices of items on shelves. Even if it is possible to frequently update the price tags (e.g. through electronic tags), it can create confusion for customers if prices change too frequently. In addition, it would still not be possible to provide customers with customized pricing. In [2], the authors discuss why it is difficult for brick-and-motar store to adjust prices. They argue that brick-and-motar stores lack the ability to identify the customers that come in and are not able to use their historical data.

Oftentimes, suppliers and retailers wish to move a certain product out of their inventory for various reasons. From conversations with management from Proctor & Gamble and Unilever, two of the world largest consumer goods conglomerate, we learned that when a supplier wants to release a newer version of a product (which could range from a simple packaging change to a totally new and innovative formulation), they have to quickly move the existing items out of the inventory [3][4]. This does not only include the supplier's own inventory, but also the inventory of retailers and stores down the supply chain. If these items do not clear the inventory by the time the new product is launched, then they are collected back incurring loss of sales and possible write-off costs. This might not be too much of a problem for modern trade retailers with sufficiently large bargaining power to ensure that such responsibility can be shared with the suppliers by returning the outdated products or by demanding promotional support from the suppliers to clear the inventory. However, it is not as simple for smaller and provisional convenience stores to do so, and they often have to absorb the loss. From a seller's point of view, wherever he/she is in the supply chain, it makes more business sense if he/she could quickly get the items out of their hands and in the meantime minimizing their loss.

In this thesis, we propose a method to move a product out of the inventory quickly while still making profit. The method is based on personalized discount from information mined from transactional database. One very important point we took into consideration when designing the algorithm was that it should be feasible to implement and deploy. While an elegant optimization techniques could be used to solve this problem, it may not be feasible to implement because of the amount of computation involved. Because we would like to get rid of the items as quickly as possible, we need to give discounts to each customer that comes in. We use personalized price discount in order to incentivize customers to buy the product in question. However, in order to maintain a reasonable level of profitability, we may not want to give everyone the full discount. Our method can ensure that the amount of discount given to each customer will approximately maximize the expected revenue.

1.1 Statement of the Problem

As more purchase transactions are stored in the database, it makes business sense to use them to increase the return on investment. This thesis proposes a method that could be used to increase profitability to suppliers and retailers through a construction of a decision support system. The ultimate goal is to generate personalized prices for different customers as they enter into trade, in real-time in an attempt to maximize profit. We limit the scope to analyze only those products which are approaching their end-of-life stage. In order to reach our objective, we formulate a price model and develop an algorithm to facilitate a construction of a decision support system to solve the problem.

1.2 Thesis Outline

This thesis is organized as follows. Chapter 2 discusses the current literature and existing methods used to solve the problem. Chapter 3 presents our price model, which is the heart of this thesis, and provides an algorithm used to solve the model. Data mining techniques used will also be discussed. Chapter 4 presents an experiment in which we benchmark our proposed algorithm against a different algorithm, the results obtained, and discussions of the results. Finally, Chapter 5 concludes the thesis and provides the reader with future research directions.

Chapter 2

Background Literature

Considerable research work exists on dynamic pricing policies, however, there had not been much work done on how discounts should be generated. A quick search on the Internet, however, shows a number of patents pertaining to personalized coupon generation. In the following subsections, we will look at the existing lituratures and discuss them briefly.

2.1 Dynamic Pricing and Coupon Discounts

Dyanmic pricing is an act of adjusting price over a period of time a product is being sold. In [5], the authors discussed the inter-temporal pricing problem. They partitioned the customers into four groups based on their valuation of the product and their patience. They then derived optimal pricing policies based on the customer group that currently dominates the market. They found that if the market is dominated by high-value patient type or low-value impatient type, then the product price should increase over time. On the other hand, if the market is dominated by low-value patient type or high-value impatient type, then the product price should decrease over time.

Since the problem of giving sales discount is very similar to giving out coupons, we have also looked into this area. The ideas of price discrimination using coupons have been around for a long time. In 1984, Narasimhan [6], using a price theoric model, found that customers who use coupons are more price sensitive than those who do not. In addition, there are evidences that brands priced higher should be couponed at higher discounts and larger sizes should also be couponed at higher discounts overall but lower per unit.

There is a considerable of literature which looks into coupon redemptions and effectiveness of coupons. In [7], Shaffer and Zhang studied a model in which promotional coupons targeted at brand switchers can lead to a prisoner's dilemma, decreasing profit for all firms. Because targeted coupons allow certain customer to pay discounted prices while others pay full prices, coupon targeting is a form of price discrimination [7]. In [8], Reibstein and Traver studied factors that affect coupon redemption rate. They developed a model which can predict coupon redemption rates based on method of distrbution, coupon's face value, discount from the coupon, size of coupon drop, and brand's consumer franchise. The authors in [9] studied coupon attractiveness and coupon proneness in order to model coupon redemption based on Item Response Theoretic (IRT) approach. They define coupon proneness as an unobserved tendency of a customer to use coupons. The authors suggested that the model can be used to design coupons that would maximize customers' response while minimizing promotion costs if we can target lower-value coupons to customers who are more coupon-prone. Inman and McAlister in [10] found that expiration dates also affect coupon redemption. According to their research, coupon redemption increases toward the end of its life because consumers regret the loss of expired coupons.

There have been several attempts to combine pricing and other components of a business process together. [1] provides a good review of relatively current literature and research direction on pricing under inventory considerations. They categorize the problem into two different classes based on market types: (i) no inventory replenishment and independent demand over time and (ii) inventory replenishment, independent demand, and myopic customers. The authors define myopic customers as those who will immediately make a purchase if the price is below their valuation. In [11], Federgruen and Heching formulate both a finite horizon model and an infinite horizon model for a single item with periodic review where demand in each period is independent and stochastic with demand distribution that is dependent on the price of the item and follows a demand function. [12] studies a model to find optimal pricing strategies for perishable products. They found that the optimal price is a decreasing function of time.

2.2 Past Purchase Behaviors

Consumer purchasing behavior has been studied extensively over the past decade. Many researchers are interested in how past behaviors can foretell future purchases. The methods used include technniques from conventional regression, economic choice theory, and data mining.

Target marketing is a means to disperse promotional materials by targeting certain individuals or groups of individuals based on some information. [13] reported that household purchase histories can potentially improve profitability in direct marketing. They found that their strategy, which gives discount increment in multiples of 5 cents, could generate 50% more revenue than what would have been gained by a blanket strategy in which no targeting is done. Another common method for target selection is through RFM (Recency, Frequency, and Monetary) analysis. In this method, customers are scored based on recency, frequency, and value of their purchase. In [14], the author introduces a target selection method using fuzzy clustering on RFM variables. They have found that the predictive power of this method is better than logistic regression and other conventional techniques.

Methods like choice models are also being used to explain purchase behaviors. As more basket data are becoming available, choice models are becoming more widely used. Using the multinomial logit model (MNL), one can express the probability that a customer will buy something as a function of its utility and utilities of other alternatives [15]. The model is based on the assumption that individuals tend to maximize preferences, and that preferences may vary based on perceptions, attitudes, or other factors [16].

There are a number of papers which study segmentation of customers and their purhase behaviors using data mining methods. In 2003, Besanko et al. in [17], studied third-degree price discrimination by retailers. Their results show that even with little segment-level information, retailers can increase their profit from that information. [18] discusses four data mining models that can be used to help understand customers and to provide electronic websites and stores with competitive advantage. The models discussed were association rule and sequential pattern mining, clustering model, classification model, and prediction model. The authors in [19] proposed a method to detect changes in customer behaviors. Their method is able to detect changes in behaviors as well as characteristics of the customer group purchasing a product. They suggested that if a business knows that a trend is emerging or is changing, then they can be prepared for such changes.

2.3 Related Decision Support Systems

The authors in [20] proposed a decision support system (DSS) to provide personalized products at customized pricing for customers of online stores. They first cluster customers by demographic information and past purchase records. Then, they use various data mining techniques to determine sales promotion strategies. This includes determining previous purchasing behaviors of all customers, a cluster of customers, and individual customers. They also employed association rules mining [21] and sequential patterns mining [22] techniques to determine cross-selling products. However, they did not take into account the effectiveness of past promotion.

Our review of literature suggests that much work has been done in theorizing and modeling dynamic pricing policies and price discounts. However, we found little evidence on how it could be done in practice for suppliers, distributors, or brick-and-motar retail stores. In the next few chapters, we present a method of achieving this. The selection of products to undergo promotional discount is beyond the scope of this thesis. We will assume that a product to be discounted has been identified and is an input to the proposed system.

Chapter 3

Methodology

In this chapter we will discuss our approach in solving the problem. In the following sections, we will be presenting our model which includes the decision process and the price model. We will then outline a top-level view of our algorithm breaking them into subproblems, and describing how we solve the problems.

3.1 The Model

This section describes our model in detail. Assumptions behind the model are listed below:

1. The distribution of customers coming in, information whether we would offer them discount, and whether they would accept or reject the offer can be derived from past history. However, the order of customers coming in is not known. This is true because the process is random, and we cannot predict who is coming in before the product is sold out[1].

[1]With good database design and considerable large amount of historical data, it may be possible to make this prediction. However, the analysis is beyond the scope of this theis.

2. There is no volume discount regardless of the quantity a customer intends to buy.
3. A customer's reaction towards a discount is a function of his/her past purchase behavior and how he/she responded to previous offers.
4. Each customer has a budget they can spend per transaction.
5. There is no time constraint requirement for the items to be completely moved out of the inventory.

3.1.1 Decision Epoch

At the beginning of the decision process, the product that we need to move out of the inventory is identified and its remaining quantity is observed. As a customer comes in, his/her past transactions are examined and past buying patterns are extracted. A decision will then be made on the price to charge and the quantity to offer. The customer then accepts or rejects the offer with the probability $P_i(p_i)$.

3.1.2 The Price Model

We now present our price model. This is shown as Inequality 3.1.

$$\begin{aligned} & p_i P_i(p_i) q_{o_i}(p_i) \\ + \frac{P_i(p_i)}{|J|+1}\left(\sum_{j\in J}\pi_j + p_i\right)(q_t - q_s - q_{o_i}(p_i)) + \frac{1-P_i(p_i)}{|J|} & \sum_{j\in J}\pi_j(q_t - q_s) \\ & + \sum_{j\in J}\pi_j q_{o_j}(p_j) \quad \geq \quad c_t \end{aligned} \tag{3.1}$$

Variables	Meanings
c_t	Total purchase cost
q_t	Total number of units to be discounted
p_b	The base price of the product per unit
p_i	Unit price offerred to customer i
$P_i(p_i)$	Probability that customer i will accept price p_i
q_s	The number of items sold since promotion
$q_{o_i}(p_i)$	The quantity offered to customer i as a function of the price offered to customer i
J	The set of customers who has purchased the product since promotion either at full price or discount price
π_j	Unit price paid for the product by customer $j \in J$

Table 3.1. The variables and constants used in the models and their interpretations.

The inequality is a direct result of our statement of the problem. It should be intuitive to see that we are trying to make sure that the revenue from the product (LHS) at least as much as the cost of it (RHS). We would like to state that we are not necessarily trying to maximize the profit. Since the product of interest is at the end of its lifecycle, we are interested in doing our best to move the product out of the inventory so that we can cover our cost of procuring it. In addition, those items we cannot sell would incur storage and dispossal cost.

Interpreting the inequality is also straightforward. Table 3.1 lists the variables used in the model. The first term of inequality 3.1 represents the expected revenue from the current customer, which is the probability that the customer will accept the offered quantity and price multiplied by the offered quantity and price. The second term is the estimated revenue from the remaining sales. The second term is broken into two parts. The first part is the expected revenue from the remaining sale if the current customer accepts the offer. The second part is the expected revenue from the remaining sale if the current customer rejects the offer. Since we do not know who will be coming into the store in the future, it is difficult to compute the expected revenue from the remaining sales. Therefore, we approximate this value by using the average sale price accepted by customers since the beginning of the promotion. The third term is then just the revenue from sales already made. These must sum up to be greater than or equal to our total purchasing cost c_t.

It is evident from the price model that the quantity and the price to be offered need to be determined as well as the probability that customers will accept the offers. In order to compute for these values, various data mining techniques will be employed.

3.2 Mining the Database

Data mining is the discovery of meaningful information from large dataset. We have employed several data mining techniques on the transactional database containing customers' transactions.

3.2.1 Customer Clustering

Cluster analysis is a data mining technique which classifies data samples into different groups. The goal of clustering is to maximize the similarity of data samples within a group and dissimilarity between groups based on common characteristics (also called features). We used clustering to group customers with similar purchase behaviors together. For our purposes, we will be using the fuzzy c-means algorithm (FCM) [23][24]. As opposed to hard clustering, fuzzy clustering allows each data point to be included in more than one cluster. This is achieved by assigning degrees of membership to data points, specifying how definitely (or ambiguously) a data point belongs to a cluster.

As summarized in [25], the degree of membership of a data point $\mathbf{x_j}$ in a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, X \subseteq \Re^p$ to different clusters c can be given by a vector $\mathbf{u}_j$.

$$\mathbf{u}_j = (u_{1j}, \ldots, u_{cj})^T \tag{3.2}$$

A fuzzy partition matrix U is then a $c \times n$ matrix with n $\mathbf{u}$ column vectors:

$$U = (U_{ij}) = \begin{pmatrix} u_{1,1} & \cdots & u_{1,n} \\ \vdots & \ddots & \vdots \\ u_{c,1} & \cdots & u_{c,n} \end{pmatrix} \tag{3.3}$$

As will soon be evident, the fuzzy partition matrix resulting from the FCM algorithm will be used in Section 3.3 to construct a fuzzy rule-based system, which will help in making decisions on our offers. For more information on the FCM algorithm, the reader is referred to [23][24][25].

3.2.2 Sequential Pattern Mining of Basket Data

Another interesting data mining technique is sequential pattern recognition. Sequential pattern mining is the process in which chronologically frequently occuring patterns are found [26]. [22] describes a method to mine sequential patterns in basket data. The reader is also referred to [21] since [21] is an extension of [22]. A sequence is defined as an ordered list of itemsets. If a customer bought item A last week, and later bought item B this week, then {(A) (B)} is a valid sequence. If a a customer bought item A and item B last week, then bought item C this week, then the sequence is {(A B) (C)}. For example, in a retail basket data, it may be found that a customer who recently bought diapers is likely to come back to buy beer. If a customer's transactions contain this sequence, then it is said that the customer supports the sequence. A *support* is then defined as the fraction of customers whose transactions contain such sequence. The sequential pattern mining algorithm finds the maximal sequences from the set of sequences satisfying a user-defined minimum support [22].

In order to use the sequential pattern mining algorithm described in [22], the database should be in the format {<u>*customer_id*</u>*, transaction_time, basket_items*} The algorithm divides sequential pattern mining problem into four phases: *sort phase, litemset phase, sequence phase, transformation phase* and *maximal phase.* In the *sort phase*, the database is sorted by customer_id then transaction_time. The *litemset phase* is similar to the algorithm for finding large itemset outlined in [21], but with a different meaning of *support.* Instead of counting every single occurence of each itemset, they are counted only once for each customer. The *sequence phase* and the *maximal phase* make passes through the set of litemsets to find sequences with minimum support. Any subsequences that are not maximal (i.e. a sequence S is not maximal if there exists a longer sequence containing sequence S) are eliminated. For more information on the algorithm, readers are referred to [22].

Understanding customers' purchase behaviors is extremely important. We do not want to give a discount to those customers we know would likely buy the product anyway. For example, a customer who recently bought a digital camera is perhaps more likely to buy a memory card the next time he/she comes back. If retailers can identify this type of pattern, then they can use this information in their marketing efforts to generate sales. In the above example, the retailer may want to send an advertising mailer to the customer to entice him/her to come back and buy a memory card for his new camera. For our purposes, we can use this information to decide whether or not to give discount. As a part of this thesis, the *AprioriAll* algorithm presented in [22] was implemented to detect sequential patterns in the test basket data.

3.3 Fuzzy Modeling

The idea of fuzziness was created to address vagueness in expressions and concepts [27]. Consider the phrase "that man is rich". How does one define rich? Using conventional set theory, we may define a set of rich people to contain anyone who has for example, at least a million dollar in a bank account. What if someone has 999,999 dollars? The person is not considered rich just because he has one dollar less than the cutoff for the set, even though in our mind we think that he/she is rich. By introducing fuzziness, we can give the person a degree of membership to describe the extent to which he belongs to the set. Figures 3.1 and 3.2 show the crisp set of rich people and the fuzzy set of rich people, respectively. The curves are called membership functions. They map the input (amount of money in our example) to the degree to which that input belongs to a given set. In Figure 3.2, the membership function is continuous with the degree of membership rising gradually as the amount of money increases.

Fuzzy modeling is a method in which fuzzy inference rules are used to describe input to output mapping through fuzzy logic. A fuzzy inference rule is of the form

If x is A, then y is B

where x is the input, and y is the output. A and B are called linguistic variables describing the fuzzy sets. For example, a rule could say:

If weather is hot, then fan speed is high

Fuzzy models can be constructed from fuzzy clustering [28]. Let $\mathbf{U}$ be a fuzzy partition matrix for the data points. $\mathbf{U}$ has dimension $C \times p$ where C is the number

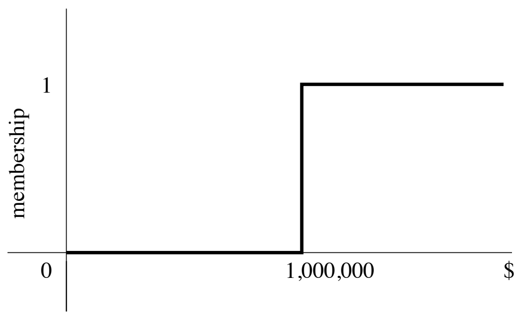

Figure 3.1. The crisp set of rich people.

of clusters and p is the dimension (number of features) of each data point. If we project each row of $\mathbf{U}$ into an individual input, then what we get is a membership function representing the degree of membership for that input. If we do this for each cluster, then we would get C membership functions for each input variable. These membership functions could be used to construct fuzzy inference rules. The problem with this method is that with large number of clusters, the projections often result in overlaps of membership functions, which need to be filtered out or merged together. [14] proposed a method to reduce the number of overlaps by projecting the clusters' centroids onto the input space and re-cluster them. This is allowed because a cluster centroid is a representative of data points in that cluster.

With inference rules, it is possible to build a control system or an expert system for making decisions. For our purpose, this is how we will obtain the probability that a customer accepts our offer.

3.4 The Algorithm

We will now present an algorithm to determine an optimal offer to customers for a product approaching its end of life. The algorithm is based on the price model 3.1 and various data mining methods described in the sections above. Figure 3.3

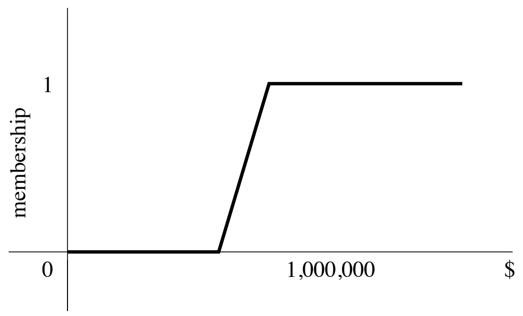

Figure 3.2. The fuzzy set of rich people.

illustrates the flow diagram of our algorithm. The algorithm can be divided into two parts depending on the time of execution. We call the first part the *offline* algorithm as it has to be run when the customers are not present. We call the second part the *online* algorithm as it has to run when a customer is present. The steps are described in next sections.

3.4.1 Offline

1) For a period of time, collect customers' response to discounts. During this training period, each customer coming into the store are given a discount to a product which they have not purchased recently. The amount of discount is randomly generated within a discrete range $R = [d_{min}, d_{max}]$. During this period, the amount of discount and the response whether or not the customer accepts the discount offer (1 for accept and 0 for reject) are recorded. This step is very crucial to our algorithm. The purpose of giving out initial discounts is to gauge customers' response toward different discounts. Using the data obtained, our algorithm will approximate the probability that a customer will accept a certain amount of discount.

2) Prepare the data and filter out any non-relevant data. We divide the data

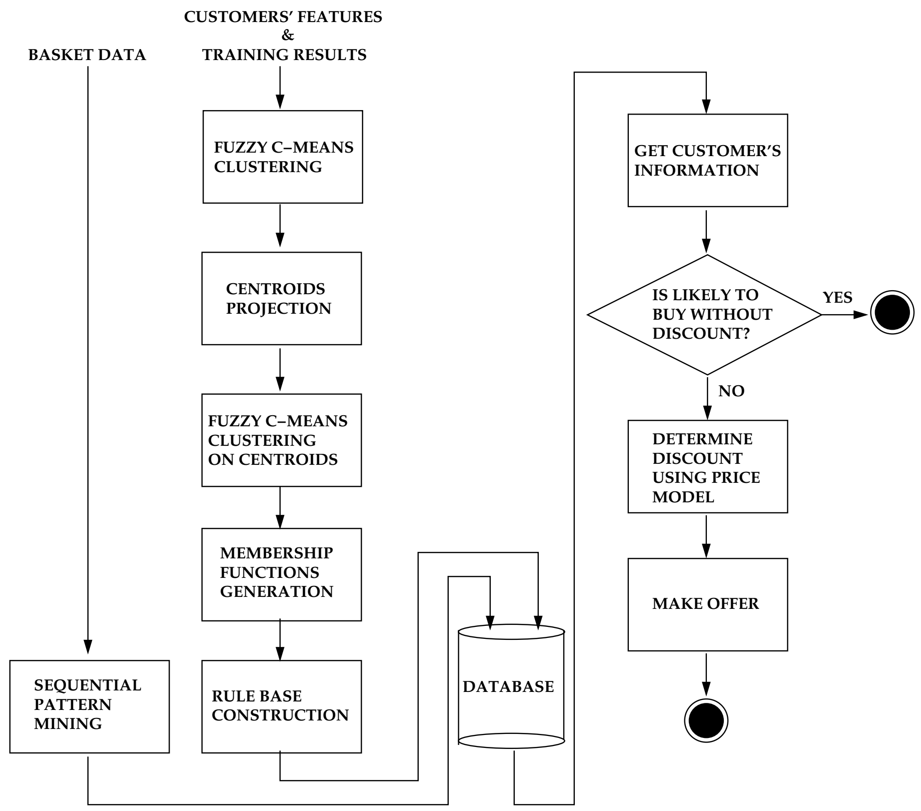
CUSTOMERS' FEATURES & TRAINING RESULTS
BASKET DATA
FUZZY C–MEANS CLUSTERING
CENTROIDS PROJECTION
FUZZY C–MEANS CLUSTERING ON CENTROIDS
MEMBERSHIP FUNCTIONS GENERATION
RULE BASE CONSTRUCTION
SEQUENTIAL PATTERN MINING
DATABASE
GET CUSTOMER'S INFORMATION
IS LIKELY TO BUY WITHOUT DISCOUNT?
YES
NO
DETERMINE DISCOUNT USING PRICE MODEL
MAKE OFFER

Figure 3.3. A flow chart describing the algorithm presented.

into two categories *(1) transaction data* and *(2) customers' features. transaction data* should contain only *customer_id*, *timestamp*, and *basket_data*. These will be used in the algorithm described in [22]. *customers' features* should contain relevant information about the customers which are likely to attribute to customers' decisions. Using RFM (Recency, Frequency, and Monetary) variables is a good start, since it is one of the most frequently used target selection methods [29]. The model says that a customer who has recently visited, who has been visiting recently, and who has been spending a decent amount are more likely to visit again. We will use these features in conjunction with the training data obtained in Step 1 to compute the probability that a customer will accept an offer. The algorithm used to compute

Table 3.2. A sample database table with RFM variables as features.

customer_id	last_visit	visit_frequency	purchase_avg	discount	accept
1	2008-08-08	5	62.98	0.05	0
2	2008-06-27	3	19.81	0.05	1
3	2008-04-27	6	28.91	0.10	1
4	2007-07-10	2	118.21	0.11	0
5	2007-08-29	1	72.12	0.08	1

the aforementioned probability is presented in the next section. Table 3.2 illustrates a sample database table for storing customers and their interesting features:

3) Perform sequential pattern mining on the *transaction data* and write the result into the database. The algorithm in [22] is a good start, however, other well-known sequential pattern mining algorithms should also work.

4) Perform fuzzy c-means clustering on *customers' features.* The benefit for clustering on these features is twofold. First, a cluster groups similar customers together based on the features. In making a decision, we can perform the same decision making steps on the indivual customer as well as on the cluster, which may provide additional meaningful insights. More importantly, fuzzy models can be constructed from these clusters.

5) Project the cluster centroids onto the input space. Let **O** be a matrix of clusters' centroids where each row represents a centroid, then projecting the centroid into each input variable is just a matter of picking a column out of the matrix **O**.

6) For each input variable, re-cluster the projected centroids using fuzzy c-means clustering. In many published papers, it is suggested that each cluster (as

opposed to centroids) should be directly projected onto the input variables. However, we found that in doing so, the membership functions overlap to a large degree. We decided to follow the proposed method in [14] which suggested one could project just the centroids and recluster them since the centroids are representatives of each of the data point in that cluster. This yielded a much better result which is easier to interpret.

7) Generate membership functions. For each input variable, these membership functions can be obtained directly from the fuzzy partition matrix **U**.

8) Generate Sugeno-type fuzzy inference rules from the membership functions. The antecedent of each rule is a combination of each of the membership functions for all the input variables. This can be done by using the fuzzy operator *AND* to join the input variables together into an antecedent. The consequents can then be generated by using the discount training data obtained in Step 1 of the offline algorithm. In this process, each rule is evaluated by the same number of customers in order to determine the RHS (consequent) of the rules. First, the weight of each rule is obtained by performing fuzzy *AND* operation on the variables in that rule. In fuzzy set theory, this is simply taking the minimum of the fuzzified output of each of the variable. The fuzzified output can be found by feeding the input from the customer's features into the membership function and obtain the corresponding membership value. After the rule weight is obtained, the consequent value is just the response indication of each customer weighted by the rule weight. Let w_{mk} be the rule weight for rule m, r_k be the response indication, which is a one if customer k accepts the amount of discount specified in rule m or zero otherwise. The consequent can then be calculated as

$$c_m = \frac{\sum_{k=1}^{N} w_{mk} r_k}{\sum_{k=1}^{N} w_{mk}} \tag{3.4}$$

The pseudocode of the algorithm used for the training is listed in Figure 3.4.

```
Input: customer's features, discount training data
for each inference rule $m$ do
    for each customer $k$ do
        for each variable $j$ in rule $m$ do
            Set $\mu_{jk}$ = membership value;
        end
        Set $w_k = min(\mu_{1k}, \ldots, \mu_{nk})$;
        Set $z_k = w_k r_k$
    end
    Set consequent for rule $i = \frac{\sum_{k=1}^{N} z_k}{\sum_{k=1}^{N} w_k}$
end
```

Figure 3.4. A pseudocode of the algorithm for training the fuzzy inference rules.

9) Store the membership functions and the rules into the database.

3.4.2 Online

1) Using customer's ID, retrieve the customer's transactions from the database.

2) Determine whether the customer is likely to buy the product without discount or not, then make a decision whether or not to offer a discount. In order to make a decision whether or not to give discount to a customer, we will have to check his/her past behaviors. It is rational to assume that the more frequent a customer purchases a product, the more chance it is that he/she will buy it in the future. For this reason, we could look at past purchases when deciding whether or not a discount should be given. If purchase history suggests that a customer has been buying a product frequently, then we should probably

not give discount for that product as it is likely that he/she will purchase it at the full price anyway. On the other hand, if the customer has rarely purchased the product, then a price discount would be necessary to entice him/her to buy the product. In addition, we could check to see whether we could infer the likelihood of purchasing from the previous purchase of other items.

The pseudocode to describe this process is given in Figure 3.5:

```
Input: customer_id,time_window,target_product,seq_weight,count_weight,
       threshold
read transaction t for customer_id during time_window;
sort t by the most recent entry first;
get sequential pattern list $S$;
isFound = 0;
count = select count(*) from t where t.basket LIKE '%target_product%';
total = select count(*) from t;
fraction = count/total;
for each sequential pattern s in S that contains target_product do
    if any subsequence of s that comes before target_product $\in$ t then
        set isFound = 1;
        break;
    end
end
if isFound * seq_weight + fraction * count_weight > threshold then
    set discount = false;
end
else
    set discount = true;
end
Output: discount
```

Figure 3.5. Making a decision to give appropriate discount.

3) If we decided to discount, then determine the price and quantity to offer. The reason why $q_{oi}(p_i)$ is a function of price is because we would like to factor in customers' budget. We do not know the exact value of customers' budget,

but we will approximate the value by taking the average of the amount they spend during each visit.

First and foremost, the amount of discount given must be such that Inequality 3.1 holds. From the inequality and Table 3.1, we can see that the only variables are p_i and $P_i(p_i)$, which are the price that we will offer to this customer and the probability that the customer will accept the price respectively. In other words, we are looking to find:

$$\begin{aligned}
z \;=\; & \operatorname*{argmax}_{\theta} \{\theta P_i(\theta) q_o(\theta) \\
& + \frac{P_i(\theta)}{|J|+1}\left(\sum_{j\in J}\pi_j + \theta\right)(q_t - q_s - q_o(\theta)) \\
& + \frac{1-P_i(\theta)}{|J|}\sum_{j\in J}\pi_j(q_t - q_s)\ \} \\
s.t. \quad & zP_i(z)q_o(z) \\
& + \frac{P_i(z)}{|J|+1}\left(\sum_{j\in J}\pi_j + z\right)(q_t - q_s - q_o(z)) \\
& + \frac{1-P_i(z)}{|J|}\sum_{j\in J}\pi_j(q_t - q_s) \\
& + \sum_{j\in J}\pi_j q_{o_j}(p_j) \qquad (3.5)\\
& \geq c_t \\
& \theta = \{x | x \in \text{Cartesian product of } p_b \text{ and } R\} \qquad (3.6)
\end{aligned}$$

We can see that a solution may or may not exist for some customers. If that is the case, then we will not give a discount to that customer. We will assume that customers will keep coming in, and that eventually we will be able to satisfy the constraints. That is, we will assume that customers' utility from the product is uniformly distributed from zero to the maximum price of the product and that they will buy if the price is lower than the utility received.

Determining the probability that a customer will accept a price is difficult. We will approximate this value through fuzzy modeling. Recall that we have stored membership functions and fuzzy inference rules in the database. We can directly use them to infer the quantity and the amount of discount we should offer to each customer using (3.5). In order to find the probability $P_i(\theta)$, the algorithm shown in Figure 3.6 is used at each discount point d_a in R, where $\theta = p_b d_a$.

```
Input: customer's features, $d_a$
for each inference rule $m$ do
    for each variable $j$ in the antecedent of $m$ do
        Set $\mu_j$ = membership value;
    end
    Set $w_m = min(\mu_1, \ldots, \mu_n)$;
    Set $z_m$ = consequent of rule $m$;
end
Let $N$ be the number or rules;
Set $P_i(\theta) = \frac{\sum_{m=1}^{N} w_m z_i}{\sum_{m=1}^{N} w_i}$;
Output: $P_i(d_a)$
```

Figure 3.6. An algorithm for finding $P_i(\theta)$.

3.5 Benchmarking the Performance

In order to benchmark our algorithm, we will compare the results from the aforementioned algorithm to a base case in which a different algorithm is applied. To simulate customer decision to buy or not to buy, we will try different common probability distributions in combination with information from the test database.

The experiment and its results are discussed in details in the next chapter.

Chapter 4

Results and Discussions

To assess the effectiveness of the algorithm, we test the algorithm on a sample database obtained by a consumer packaged goods supplier in Thailand. We will compare the result from the aforementioned algorithm to the base case in which no algorithm was applied. To simulate customer decision to buy or not to buy, we will simulate the process as a random process with some common probability distributions in combination with information from the sample data.

4.1 Description of Data

The data obtained is for a distribution center in the north-eastern part of Thailand. The distribution center is responsible for distributing products to stores ranging from small stores such as minimarts to larger convenient stores, supermarkets, and wholesalers. The data is for a period of seven months from July 2006 to January 2007, during which there were more than 20,000 transactions from approximately 8,600 customers purchasing from 672 SKUs.

In order to make the data useful in our algorithm, we transformed the origi-

Table 4.1. The description of the fields of the tables created to store the data.

Table	Field	Description
Product	sku	The SKU of the product.
	unit_cost	The procurement cost per unit of the product.
	unit_price	The price of a unit of the product.
Basket	customer_id	The customer associated with this transaction.
	invc_date	The date the order was made.
	basket	The content of the basket. A list of items separated by commas.
Transaction	customer_id	The customer associated with this transaction.
	invc_date	The date the order was made.
	sku	The SKU of the product.
	quantity	The quantity of the products ordered in that transaction.

nal data into a different format. The original data contains the following fields: *Invoice Number, Invoice Date, Customer ID, SKU, Shipment Quantity, Product Price,* and *Product Cost.* The data was transformed into three tables: *product, transaction,* and *basket.* The *basket* table is the basket data of each transaction, which essentially is the de-normalized version of the *transaction* table. The de-normalized table is there to facilitate the transformation phase of the sequential pattern mining algorithm in [22]. The fields in the tables and their descriptions are shown in Table 4.1.

4.2 Offline Results

In the next few subsections below, we will be discussing the results we got from running the offline algorithm as outlined in the previous chapter. The results include sequential patterns from the sequential pattern mining step, fuzzy membership functions obtained from FCM clustering and cluster projection, and finally

Table 4.2. A small portion of the sequences output from the sequential pattern mining algorithm.

1-Sequences	2-Sequences	3-Sequences
1	7 15	12 11 35
2	14 36	
6	11 36	
8	9 34	
19		
20		

fuzzy inference rules derived from fuzzy membership functions and trained discount acceptance data.

4.2.1 Sequential Pattern Mining

The algorithm outlined in [22] was employed over the data described in the section above. With the minimum support set at 0.15, the algorithm returns 39 large itemsets and 38 large sequences. For the purpose of benchmarking the performance of our algorithm, we have picked one sequence to be used in the online algorithm. A small portion of the output is shown in Table 4.2. It is important to recognize that the integers shown below are each mapped to an itemset, therefore they may not be singular. For example, sequence 14 36 implies that items in itemset 36 are likely to be bought after items in itemset 14.

4.2.2 Membership Functions and Fuzzy Inference Rules

We generate membership functions using the method described in the previous chapter. We first select the features we are interested in. These features would appear in the antecedent of each of the fuzzy inference rules. The features we used were:

- Days since last purchase (We assume that the current date is one day after the last day of the data that we have. In practice, this would have been the date the algorithm is being run.)
- Number of purchases over the past six months
- The average amount spent per visit

These features were used in determining distances between clusters during FCM clustering. In addition to these features, we also used the training data, which is the discount given to each customer. We used 40 clusters during the first FCM pass. This number was chosen so that the second FCM pass would be done on a sufficiently large number of data points allowing for meaningful generation of membership functions. After projecting the cluster centroids into the input space, the centroids were reclustered again into 4 clusters. This resulted in 4 membership functions for each of the features. The membership functions for the features are shown in Figure 4.1.

The antecendents of fuzzy inference rules are then formed from the Cartesian product of the membership functions. The consequents are formed from the training data weighted by each of the rule's firing strength as described in the previous chapter (Equation 3.4). We obtained a total of 256 (4^4) rules from the membership functions. In order to make evaluation of the rules quicker, the rules with zero in the consequent are removed. Finally, we are left with 155 rules. Examples of the rules are shown in Table 4.3. The complete list of rules is listed in Appendix B. If we look at the rules, we could see that they make perfect sense. For example, rule 3 and rule 6 are different in that the amount spent and the discount for rule 6 is greater than those in rule 3, and we see that the probability that a customer's

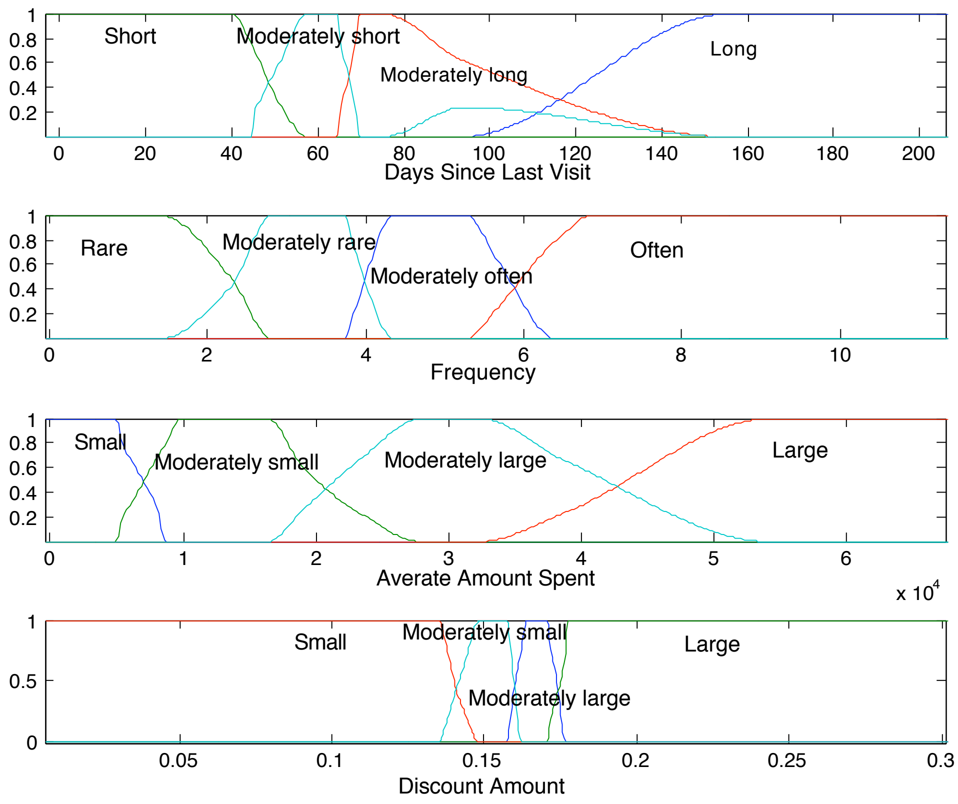

Figure 4.1. The membership functions generated by the algorithm.

features and discount pair which falls under rule 6 would be more likely to accept the discount.

These rules are then used to determine the probability that a customer will accept an offer of a certain price. With this information, we could determine the expected revenue using (3.1) and (3.5).

4.3 Online Results

Since we were not able to experiment on real customers to validate our model, we used a simulation to benchmark the performance. We first generated customer data based on the existing data that we have. In order to make comparisons

Table 4.3. Some of the inference rules obtained from the algorithm.

1.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.3981
2.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 0.566
3.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.052
4.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.375
5.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.830
6.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately small** *and* `discount` is **large** *then* `p(accept)` is 0.914
7.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately small** *and* `discount` is **small** *then* `p(accept)` is 0.220
8.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.370

between the two systems (the two algorithms), we used common random numbers for each simulation trial across the two algorithms. This means that the order of the customers entering the systems are the same for both our algorithm and the benchmark algorithm within the same trial.

1) The total revenue generated from the sales

2) The amount of time it takes for the product to be sold out

Since our product is at the end of its lifecycle, our main concern is moving it out of the inventory as fast as possible without incurring extra costs (the missing sale opportunity as well as the cost of destroying/storing the product). The total revenue generated from the sales is the amount we get at the end of the sales, which is when the product is sold out. The larger the revenue, the better the algorithm is. In addition, retailers and suppliers often look to replenish their shelf with a new product. If they can move the old product out of the inventory quicker, then they can start generating revenue from the new product sooner. Therefore, the quicker an algorithm can move a product out of the inventory, the better the algorithm is.

For the purpose of comparison, we compared our algorithm with another algorithm that we created. This algorithm looks at customers' purchase behavior and categorizes them into different groups. Each of the groups has different characteristics in how they would respond to the discounts. We assume that their goal is to minimize cost. The algorithm, however, does not take into account, the minimum revenue requirement. The results of the simulations are shown collectively in Table 4.4. Since we are using common random number, it is expected that our results are positively correlated, and therefore we can use (4.1) to find the confidence interval between the difference of the means [30].

$$\bar{Y}_1 - \bar{Y}_2 \pm t_{1-\alpha,n-1}\sqrt{\frac{S_D^2}{N}} \tag{4.1}$$

Our 95% confidence interval on the difference on the revenue means is then:

$$\begin{aligned} 39711 - 35510 &\pm 1.699\sqrt{675933.3224} \\ 4201 &\pm 1396.83564 \end{aligned}$$

and

$$\begin{aligned} 175.53 - 364.13 &\pm 1.699\sqrt{717.65643} \\ -188.6 &\pm 45.51469838 \end{aligned}$$

for the difference on the number of customer means.

The result shows that our algorithm is better than the benchmark algorithm in term of performance which is determined by the revenue generated from the sale as well as the amount of time it takes for the inventory to be completely depleted.

Table 4.4. The results comparing the performance our algorithm with the benchmark algorithm.

Trial	Revenue			Number of Customers		
	Ours	Benchmark	Differences	Ours	Benchmark	Differences
1	40535.77	46214.63	-5678.86	222	400	-178
2	39413.35	32073.21	7340.14	205	331	-126
3	39308.01	35578.35	3729.66	199	249	-50
4	40649.74	35494.52	5155.21	134	371	-237
5	39580.41	35317.66	4262.76	213	393	-180
6	38745.51	34939.72	3805.79	143	275	-132
7	39104.25	31962.76	7141.49	139	305	-166
8	39793.67	32178.53	7615.14	165	194	-29
9	41210.51	47958.63	-6748.11	147	489	-342
10	37206.06	32317.64	4888.43	109	253	-144
11	39740.58	32235.08	7505.5	212	372	-160
12	39715.97	33423.1	6292.87	171	250	-79
13	41887.85	33369.8	8518.05	184	270	-86
14	40491.73	32929.95	7561.78	246	599	-353
15	38581.46	36966	1615.46	117	261	-144
16	39800.58	38880.95	919.63	199	550	-351
17	41186.34	45698.87	-4512.54	120	760	-640
18	37902.83	35277.99	2624.84	113	262	-149
19	39506.59	32369.5	7137.09	249	497	-248
20	39813.1	32576.89	7236.21	145	58	87
21	39393.92	31319.24	8074.68	208	347	-139
22	39474.65	33723.63	5751.02	173	686	-513
23	38575.42	34597.29	3978.13	171	339	-168
24	39281.68	31061.33	8220.35	135	169	-34
25	39969.38	39987.52	-18.14	306	499	-193
26	41480.33	46370.98	-4890.66	205	260	-55
27	40310.42	31579.63	8730.79	78	176	-98
28	39046.4	31475.95	7570.45	143	331	-188
29	40218.9	35921.18	4297.72	210	542	-332
30	39421.98	31510.52	7911.46	205	436	-231
Mean	39711.58	35510.37	4201.21	175.53	364.13	-188.6
Std Dev	1013.04	4930.6	4503.11	49.61	158.72	146.73

Chapter 5

Conclusions and Future Work

We have presented a price model and a method to quickly move items out of the inventory. We used data mining techniques which include sequential pattern mining and clustering integrated with a knowledge base system built using fuzzy inference rules which is also formed from mining the data. In comparison with a benchmark algorithm, we found that our algorithm performs better in terms of the revenue generated as well as the amount of time it takes for the items to be completely depleted. We can guarantee that our algorithm will result in a solution that will help recovering the cost spent in procuring an item. Traditionally, this kind of problems can be solved with optimization techniques such as dynamic programming. However, the amount of computation required hinder them from being used in real-time applications. Our algorithm provides a good solution as each customer enters our store. By examining their past purchase behavior individually as well as collectively in conjunction with their proneness to discount, we can make a good estimate of a discount that would maximize our expected revenue. Once the past behaviors are mined from the database and the rules are constructed, our algorithm will run in constant time. The algorithm only needs to fuzzify the input and defuzzify the output which is straightforward. The only factor that may increase the computational time is the number of input variables used. Because the number of rules increase exponentially as the number of input variables (features) increases (recall that the number of rules is the number of fuzzy membership functions for each input variables raised to the power of the number of input variables), it can be time consuming to evaluate the rules as each customer comes in. In our experiment, it took approximately 6 seconds to evaluate each customer if the system decide to give a discount to that customer on a laptop machine running a 2.33 GHz Intel Core 2 Duo CPU with 2 GB of system memory.

The use of fuzzy inference rules also makes the system easy for humans to interpret. It might have been possible to use a different method such as neural networks in the place of fuzzy inference rules. However, it is more difficult to understand what goes on inside a neural network, and thus much more difficult to configure them.

Someone who may want to implement a similar system may be concerned with costs associated with running and maintaining it. We feel that this cost is negligible for most modern day suppliers, distributors, or retailers since they should already have transactional database in place. It should be straightforward to set up a batch process to run at night to mine the data and derive the fuzzy inference rules. If the system works as intended, the amount of savings should outweigh the cost of maintaining such system.

5.1 Future Work and Extensions

We believe that this is a good start at an attempt to create an automated knowledge based that could be used to generate personalized discount for customers.

Currently, the price model and the algorithm put more weight on the amount of revenue generated since the only constraint is that the revenue should be at least equal to the cost of the product. As it currently is, we formulated it as an infinite time horizon problem. It might be possible to generalize the methodology to also take into account the time it would take to completely move the product out of the inventory. In order to do this, we would have to keep track of the number of customers who has already been to the store. The expected revenue in the price model wil have to reflect the number of customers that we have left before the end of the decision horizon, which would now be finite.

If we make our problem a finite-time problem, there would be a number of interesting issues that we could look at. We could consider storage and write-off costs. For each time period that passes with items left in the inventory, some storage cost will be incurred. Another interesting addition is to introduce volume discount. During the training period, we could give out volume discount and see how each customer responses. We believe that this addition would allow faster sales of the items.

It would also be interesting to see how different features could be used during the clustering process. We were limited by our data and were able to only use the RFM variables. The algorithm does not have any limitations on the features. One may want to try different customers' attributes such as demographics information. As long as the features selected are a good indicator of the customers' behaviors and responses, then they are usable.

Appendix A

Source Code

Shown in this Appendix is a sample MATLAB source code used to obtain the consequent of the fuzzy inference rules.

```
function [rules,rho] = trainData(numCustomers,memx,memy,maty,rules)

[numVars,numFuncs] = size(memy);
numRules = numFuncs ^ numVars;

% Load the customer
conn = database('thesis2','*****','*****', ...
    'com.mysql.jdbc.Driver','jdbc:mysql://localhost/thesis2');

sqlAttr = ...
  'SELECT last_purchase,frequency,amount_avg,discount,acceptance ' ...
  'FROM attribute';

curs = exec(conn,sqlAttr);
curs = fetch(curs);
[numCustomers,numCols] = size(curs.Data);

attrs = cell2mat(curs.Data);

rho = zeros(1,numRules);
for i=1:numRules
  uik = zeros(1,numCustomers);
  top = zeros(1,numCustomers);
  for k=1:numCustomers
    u = zeros(1,numVars);
    for j=1:numVars
```

```
      u(j) = interp1(memx(j,:),maty(rules(i,j),:,j),attrs(k,j));
    end
    uik(k) = min(u); % this is uik
    top(k) = uik(k) * attrs(k,5); % attrs(k,5) is rk
  end
  rho(i) = sum(top)/sum(uik);
end
```

Appendix **B**

Complete Fuzzy Inference Rules

This appendix list all of the fuzzy inference rules obtained from the algorithm.

154.	If `last_visit` is **moderately short** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately large** *and* `discount` is **large** *then* `p(accept)` is 1.000
155.	If `last_visit` is **moderately short** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately large** *and* `discount` is **small** *then* `p(accept)` is 0.196
1.	If `last_visit` is **long** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 0.518
2.	If `last_visit` is **long** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.058
3.	If `last_visit` is **long** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.706
4.	If `last_visit` is **long** *and* `frequency` is **moderately often**

	and `avg_spent` is **moderately small**
	and `discount` is **large**
	then `p(accept)` is 1.000
	If `last_visit` is **long**
	and `frequency` is **rare**
5.	*and* `avg_spent` is **small**
	and `discount` is **moderately large**
	then `p(accept)` is 0.161
	If `last_visit` is **long**
	and `frequency` is **rare**
6.	*and* `avg_spent` is **small**
	and `discount` is **large**
	then `p(accept)` is 0.276
	If `last_visit` is **long**
	and `frequency` is **rare**
7.	*and* `avg_spent` is **small**
	and `discount` is **small**
	then `p(accept)` is 0.024
	If `last_visit` is **long**
	and `frequency` is **rare**
8.	*and* `avg_spent` is **small**
	and `discount` is **moderately small**
	then `p(accept)` is 0.088
	If `last_visit` is **long**
	and `frequency` is **rare**
9.	*and* `avg_spent` is **moderately small**
	and `discount` is **moderately large**
	then `p(accept)` is 0.585
	If `last_visit` is **long**
	and `frequency` is **rare**
10.	*and* `avg_spent` is **moderately small**
	and `discount` is **large**
	then `p(accept)` is 0.805
	If `last_visit` is **long**
	and `frequency` is **rare**
11.	*and* `avg_spent` is **moderately small**
	and `discount` is **small**
	then `p(accept)` is 0.098
	If `last_visit` is **long**
	and `frequency` is **rare**
12.	*and* `avg_spent` is **moderately small**
	and `discount` is **moderately small**
	then `p(accept)` is 0.748
	If `last_visit` is **long**
	and `frequency` is **rare**
13.	

	and `avg_spent` is **large** *and* `discount` is **large** *then* `p(accept)` is 1.000
14.	If `last_visit` is **long** *and* `frequency` is **rare** *and* `avg_spent` is **moderately large** *and* `discount` is **moderately large** *then* `p(accept)` is 1.000
15.	If `last_visit` is **long** *and* `frequency` is **rare** *and* `avg_spent` is **moderately large** *and* `discount` is **large** *then* `p(accept)` is 1.000
16.	If `last_visit` is **long** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 1.000
17.	If `last_visit` is **long** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.478
18.	If `last_visit` is **long** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.989
19.	If `last_visit` is **long** *and* `frequency` is **often** *and* `avg_spent` is **moderately small** *and* `discount` is **large** *then* `p(accept)` is 1.000
20.	If `last_visit` is **long** *and* `frequency` is **often** *and* `avg_spent` is **moderately small** *and* `discount` is **small** *then* `p(accept)` is 0.727
21.	If `last_visit` is **long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.204
22.	If `last_visit` is **long** *and* `frequency` is **moderately rare**

	and `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 0.462
23.	If `last_visit` is **long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.083
24.	If `last_visit` is **long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.268
25.	If `last_visit` is **long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.369
26.	If `last_visit` is **long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately small** *and* `discount` is **large** *then* `p(accept)` is 0.975
27.	If `last_visit` is **long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately small** *and* `discount` is **small** *then* `p(accept)` is 0.183
28.	If `last_visit` is **long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.549
29.	If `last_visit` is **long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately large** *and* `discount` is **large** *then* `p(accept)` is 1.000
30.	If `last_visit` is **short** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.291
31.	If `last_visit` is **short** *and* `frequency` is **moderately often**

	and `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 0.572
32.	If `last_visit` is **short** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.082
33.	If `last_visit` is **short** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.307
34.	If `last_visit` is **short** *and* `frequency` is **moderately often** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.489
35.	If `last_visit` is **short** *and* `frequency` is **moderately often** *and* `avg_spent` is **moderately small** *and* `discount` is **large** *then* `p(accept)` is 0.875
36.	If `last_visit` is **short** *and* `frequency` is **moderately often** *and* `avg_spent` is **moderately small** *and* `discount` is **small** *then* `p(accept)` is 0.141
37.	If `last_visit` is **short** *and* `frequency` is **moderately often** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.665
38.	If `last_visit` is **short** *and* `frequency` is **moderately often** *and* `avg_spent` is **large** *and* `discount` is **large** *then* `p(accept)` is 1.000
39.	If `last_visit` is **short** *and* `frequency` is **moderately often** *and* `avg_spent` is **moderately large** *and* `discount` is **large** *then* `p(accept)` is 1.000
40.	If `last_visit` is **short** *and* `frequency` is **moderately often**

	and `avg_spent` is **moderately large** *and* `discount` is **small** *then* `p(accept)` is 0.029
41.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.080
42.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 0.267
43.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.016
44.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.099
45.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.193
46.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately small** *and* `discount` is **large** *then* `p(accept)` is 0.871
47.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately small** *and* `discount` is **small** *then* `p(accept)` is 0.136
48.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.162
49.	If `last_visit` is **short** *and* `frequency` is **rare**

	and `avg_spent` is **large** *and* `discount` is **moderately large** *then* `p(accept)` is 1.000
50.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **large** *and* `discount` is **large** *then* `p(accept)` is 1.000
51.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **large** *and* `discount` is **moderately small** *then* `p(accept)` is 1.000
52.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately large** *and* `discount` is **moderately large** *then* `p(accept)` is 0.701
53.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately large** *and* `discount` is **large** *then* `p(accept)` is 1.000
54.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately large** *and* `discount` is **small** *then* `p(accept)` is 0.352
55.	If `last_visit` is **short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately large** *and* `discount` is **moderately small** *then* `p(accept)` is 1.000
56.	If `last_visit` is **short** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.413
57.	If `last_visit` is **short** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 0.738
58.	If `last_visit` is **short** *and* `frequency` is **often**

	and `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.133
59.	If `last_visit` is **short** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.395
60.	If `last_visit` is **short** *and* `frequency` is **often** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.352
61.	If `last_visit` is **short** *and* `frequency` is **often** *and* `avg_spent` is **moderately small** *and* `discount` is **large** *then* `p(accept)` is 0.938
62.	If `last_visit` is **short** *and* `frequency` is **often** *and* `avg_spent` is **moderately small** *and* `discount` is **small** *then* `p(accept)` is 0.239
63.	If `last_visit` is **short** *and* `frequency` is **often** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.586
64.	If `last_visit` is **short** *and* `frequency` is **often** *and* `avg_spent` is **large** *and* `discount` is **moderately large** *then* `p(accept)` is 1.000
65.	If `last_visit` is **short** *and* `frequency` is **often** *and* `avg_spent` is **large** *and* `discount` is **large** *then* `p(accept)` is 1.000
66.	If `last_visit` is **short** *and* `frequency` is **often** *and* `avg_spent` is **large** *and* `discount` is **small** *then* `p(accept)` is 0.122
67.	If `last_visit` is **short** *and* `frequency` is **often**

	and `avg_spent` is **moderately large**
	and `discount` is **large**
	then `p(accept)` is 0.958
	If `last_visit` is **short**
	and `frequency` is **often**
68.	*and* `avg_spent` is **moderately large**
	and `discount` is **small**
	then `p(accept)` is 0.167
	If `last_visit` is **short**
	and `frequency` is **often**
69.	*and* `avg_spent` is **moderately large**
	and `discount` is **moderately small**
	then `p(accept)` is 0.454
	If `last_visit` is **short**
	and `frequency` is **moderately rare**
70.	*and* `avg_spent` is **small**
	and `discount` is **moderately large**
	then `p(accept)` is 0.243
	If `last_visit` is **short**
	and `frequency` is **moderately rare**
71.	*and* `avg_spent` is **small**
	and `discount` is **large**
	then `p(accept)` is 0.438
	If `last_visit` is **short**
	and `frequency` is **moderately rare**
72.	*and* `avg_spent` is **small**
	and `discount` is **small**
	then `p(accept)` is 0.041
	If `last_visit` is **short**
	and `frequency` is **moderately rare**
73.	*and* `avg_spent` is **small**
	and `discount` is **moderately small**
	then `p(accept)` is 0.226
	If `last_visit` is **short**
	and `frequency` is **moderately rare**
74.	*and* `avg_spent` is **moderately small**
	and `discount` is **moderately large**
	then `p(accept)` is 0.511
	If `last_visit` is **short**
	and `frequency` is **moderately rare**
75.	*and* `avg_spent` is **moderately small**
	and `discount` is **large**
	then `p(accept)` is 0.870
	If `last_visit` is **short**
	and `frequency` is **moderately rare**
76.	

	and `avg_spent` is **moderately small** *and* `discount` is **small** *then* `p(accept)` is 0.251
77.	If `last_visit` is **short** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.628
78.	If `last_visit` is **short** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately large** *and* `discount` is **large** *then* `p(accept)` is 1.000
79.	If `last_visit` is **short** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately large** *and* `discount` is **small** *then* `p(accept)` is 0.150
80.	If `last_visit` is **moderately long** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.193
81.	If `last_visit` is **moderately long** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 0.507
82.	If `last_visit` is **moderately long** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.036
83.	If `last_visit` is **moderately long** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.281
84.	If `last_visit` is **moderately long** *and* `frequency` is **moderately often** *and* `avg_spent` is **moderately small** *and* `discount` is **large** *then* `p(accept)` is 1.000
85.	If `last_visit` is **moderately long** *and* `frequency` is **moderately often**

	and `avg_spent` is **moderately large**
	and `discount` is **large**
	then `p(accept)` is 1.000
	If `last_visit` is **moderately long**
	and `frequency` is **rare**
86.	*and* `avg_spent` is **small**
	and `discount` is **moderately large**
	then `p(accept)` is 0.109
	If `last_visit` is **moderately long**
	and `frequency` is **rare**
87.	*and* `avg_spent` is **small**
	and `discount` is **large**
	then `p(accept)` is 0.340
	If `last_visit` is **moderately long**
	and `frequency` is **rare**
88.	*and* `avg_spent` is **small**
	and `discount` is **small**
	then `p(accept)` is 0.019
	If `last_visit` is **moderately long**
	and `frequency` is **rare**
89.	*and* `avg_spent` is **small**
	and `discount` is **moderately small**
	then `p(accept)` is 0.128
	If `last_visit` is **moderately long**
	and `frequency` is **rare**
90.	*and* `avg_spent` is **moderately small**
	and `discount` is **moderately large**
	then `p(accept)` is 1.000
	If `last_visit` is **moderately long**
	and `frequency` is **rare**
91.	*and* `avg_spent` is **moderately small**
	and `discount` is **large**
	then `p(accept)` is 0.845
	If `last_visit` is **moderately long**
	and `frequency` is **rare**
92.	*and* `avg_spent` is **moderately small**
	and `discount` is **small**
	then `p(accept)` is 0.157
	If `last_visit` is **moderately long**
	and `frequency` is **rare**
93.	*and* `avg_spent` is **moderately small**
	and `discount` is **moderately small**
	then `p(accept)` is 0.686
	If `last_visit` is **moderately long**
	and `frequency` is **rare**
94.	

	and `avg_spent` is **large** *and* `discount` is **large** *then* `p(accept)` is 1.000
95.	If `last_visit` is **moderately long** *and* `frequency` is **rare** *and* `avg_spent` is **moderately large** *and* `discount` is **moderately large** *then* `p(accept)` is 1.000
96.	If `last_visit` is **moderately long** *and* `frequency` is **rare** *and* `avg_spent` is **moderately large** *and* `discount` is **large** *then* `p(accept)` is 0.759
97.	If `last_visit` is **moderately long** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 1.000
98.	If `last_visit` is **moderately long** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.151
99.	If `last_visit` is **moderately long** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.101
100.	If `last_visit` is **moderately long** *and* `frequency` is **often** *and* `avg_spent` is **moderately small** *and* `discount` is **large** *then* `p(accept)` is 1.000
101.	If `last_visit` is **moderately long** *and* `frequency` is **often** *and* `avg_spent` is **moderately small** *and* `discount` is **small** *then* `p(accept)` is 0.499
102.	If `last_visit` is **moderately long** *and* `frequency` is **often** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.366
103.	If `last_visit` is **moderately long** *and* `frequency` is **often**

	and `avg_spent` is **large** *and* `discount` is **moderately large** *then* `p(accept)` is 1.000
104.	If `last_visit` is **moderately long** *and* `frequency` is **often** *and* `avg_spent` is **moderately large** *and* `discount` is **moderately large** *then* `p(accept)` is 1.000
105.	If `last_visit` is **moderately long** *and* `frequency` is **often** *and* `avg_spent` is **moderately large** *and* `discount` is **large** *then* `p(accept)` is 1.000
106.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.398
107.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 0.566
108.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.052
109.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.375
110.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.830
111.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately small** *and* `discount` is **large** *then* `p(accept)` is 0.914
112.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare**

	and `avg_spent` is **moderately small** *and* `discount` is **small** *then* `p(accept)` is 0.220
113.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.370
114.	If `last_visit` is **moderately long** *and* `frequency` is **moderately rare** *and* `avg_spent` is **moderately large** *and* `discount` is **large** *then* `p(accept)` is 1.000
115.	If `last_visit` is **moderately short** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.292
116.	If `last_visit` is **moderately short** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 0.547
117.	If `last_visit` is **moderately short** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.044
118.	If `last_visit` is **moderately short** *and* `frequency` is **moderately often** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.210
119.	If `last_visit` is **moderately short** *and* `frequency` is **moderately often** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately large** *then* `p(accept)` is 1.000
120.	If `last_visit` is **moderately short** *and* `frequency` is **moderately often** *and* `avg_spent` is **moderately small** *and* `discount` is **large** *then* `p(accept)` is 0.925
121.	If `last_visit` is **moderately short** *and* `frequency` is **moderately often**

	and `avg_spent` is **moderately small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.743
122.	If `last_visit` is **moderately short** *and* `frequency` is **moderately often** *and* `avg_spent` is **moderately large** *and* `discount` is **large** *then* `p(accept)` is 1.000
123.	If `last_visit` is **moderately short** *and* `frequency` is **rare** *and* `avg_spent` is **small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.125
124.	If `last_visit` is **moderately short** *and* `frequency` is **rare** *and* `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 0.290
125.	If `last_visit` is **moderately short** *and* `frequency` is **rare** *and* `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.018
126.	If `last_visit` is **moderately short** *and* `frequency` is **rare** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.100
127.	If `last_visit` is **moderately short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately small** *and* `discount` is **moderately large** *then* `p(accept)` is 1.000
128.	If `last_visit` is **moderately short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately small** *and* `discount` is **large** *then* `p(accept)` is 0.909
129.	If `last_visit` is **moderately short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately small** *and* `discount` is **small** *then* `p(accept)` is 0.144
130.	If `last_visit` is **moderately short** *and* `frequency` is **rare**

	and `avg_spent` is **moderately small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.837
131.	If `last_visit` is **moderately short** *and* `frequency` is **rare** *and* `avg_spent` is **large** *and* `discount` is **large** *then* `p(accept)` is 1.000
132.	If `last_visit` is **moderately short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately large** *and* `discount` is **moderately large** *then* `p(accept)` is 1.000
133.	If `last_visit` is **moderately short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately large** *and* `discount` is **large** *then* `p(accept)` is 1.000
134.	If `last_visit` is **moderately short** *and* `frequency` is **rare** *and* `avg_spent` is **moderately large** *and* `discount` is **small** *then* `p(accept)` is 0.399
135.	If `last_visit` is **moderately short** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **moderately large** *then* `p(accept)` is 0.339
136.	If `last_visit` is **moderately short** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **large** *then* `p(accept)` is 0.844
137.	If `last_visit` is **moderately short** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **small** *then* `p(accept)` is 0.099
138.	If `last_visit` is **moderately short** *and* `frequency` is **often** *and* `avg_spent` is **small** *and* `discount` is **moderately small** *then* `p(accept)` is 0.071
139.	If `last_visit` is **moderately short** *and* `frequency` is **often**

and `avg_spent` is **moderately small**
and `discount` is **moderately large**
then `p(accept)` is 0.717

If `last_visit` is **moderately short**
and `frequency` is **often**
140. *and* `avg_spent` is **moderately small**
and `discount` is **large**
then `p(accept)` is 0.829

If `last_visit` is **moderately short**
and `frequency` is **often**
141. *and* `avg_spent` is **moderately small**
and `discount` is **small**
then `p(accept)` is 0.208

If `last_visit` is **moderately short**
and `frequency` is **often**
142. *and* `avg_spent` is **moderately small**
and `discount` is **moderately small**
then `p(accept)` is 0.607

If `last_visit` is **moderately short**
and `frequency` is **often**
143. *and* `avg_spent` is **large**
and `discount` is **moderately large**
then `p(accept)` is 1.000

If `last_visit` is **moderately short**
and `frequency` is **often**
144. *and* `avg_spent` is **moderately large**
and `discount` is **moderately large**
then `p(accept)` is 1.000

If `last_visit` is **moderately short**
and `frequency` is **often**
145. *and* `avg_spent` is **moderately large**
and `discount` is **large**
then `p(accept)` is 0.514

If `last_visit` is **moderately short**
and `frequency` is **moderately rare**
146. *and* `avg_spent` is **small**
and `discount` is **moderately large**
then `p(accept)` is 0.366

If `last_visit` is **moderately short**
and `frequency` is **moderately rare**
147. *and* `avg_spent` is **small**
and `discount` is **large**
then `p(accept)` is 0.477

If `last_visit` is **moderately short**
and `frequency` is **moderately rare**
148.

	and avg_spent is **small** *and* discount is **small** *then* p(accept) is 0.040
149.	If last_visit is **moderately short** *and* frequency is **moderately rare** *and* avg_spent is **small** *and* discount is **moderately small** *then* p(accept) is 0.247
150.	If last_visit is **moderately short** *and* frequency is **moderately rare** *and* avg_spent is **moderately small** *and* discount is **moderately large** *then* p(accept) is 0.957
151.	If last_visit is **moderately short** *and* frequency is **moderately rare** *and* avg_spent is **moderately small** *and* discount is **large** *then* p(accept) is 0.896
152.	If last_visit is **moderately short** *and* frequency is **moderately rare** *and* avg_spent is **moderately small** *and* discount is **small** *then* p(accept) is 0.087
153.	If last_visit is **moderately short** *and* frequency is **moderately rare** *and* avg_spent is **moderately small** *and* discount is **moderately small** *then* p(accept) is 0.793
154.	If last_visit is **moderately short** *and* frequency is **moderately rare** *and* avg_spent is **moderately large** *and* discount is **large** *then* p(accept) is 1.000
155.	If last_visit is **moderately short** *and* frequency is **moderately rare** *and* avg_spent is **moderately large** *and* discount is **small** *then* p(accept) is 0.196

Bibliography

[1] ELMAGHRABY, W. and P. KESKINOCAK (2003) "Dynamic Pricing in the Presence of Inventory Considerations: Research Overview, Current Practices, and Future Directions," *Management Science*, **49**(10), pp. 1287–1309.

[2] BAKER, W. L., E. LIN, M. V. MARN, and C. C. ZAWADA (2001) "Getting Prices Right on the Web," *The McKinsey Quarterly*, (2), pp. 54–63.

[3] MANAWANITJARERN, W. (2008), Personal Interview.

[4] JENCHARAT, S. (2008), Personal Interview.

[5] SU, X. "Inter-temporal Pricing with Strategic Customer Behavior," .

[6] NARASIMHAN, C. (1984) "A Price Discrimination Theory of Coupons," *Marketing Science*, **3**(2), pp. 128–147.

[7] SHAFFER, G. and Z. J. ZHANG (1995) "Competitive Coupon Targeting," *Marketing Science*, **14**(4), pp. 395–416.

[8] REIBSTEIN, D. J. and P. A. TRAVER (1982) "Factors Affecting Coupon Redemption Rates," *Journal of Marketing*, **46**(4), pp. 102–113.

[9] BAWA, K., S. S. SRINIVASAN, and R. K. SRIVASTAVA (1997) "Coupon Attractiveness and Coupon Proneness: A Framework for Modeling Coupon Redemption," *Journal of Marketing Research*, **34**(4), pp. 517–525.

[10] INMAN, J. J. and L. MCALISTER (1994) "Do Coupon Expiration Dates Affect Consumer Behavior?" *Journal of Marketing Research*, **31**(3), pp. 423–428.

[11] FEDERGRUEN, A. and A. HECHING (1999) "Combined Pricing and Inventory Control Under Uncertainty," *Operations Research*, **47**(3), pp. 454–475.

[12] Bitran, G. R. and S. V. Mondschein (1993) "Pricing Perishable Products: An Application to the Retail Industry," Working paper.

[13] Rossi, P. E., R. E. McCulloch, and G. M. Allenby (1996) "The Value of Purchase History Data in Target Marketing," *Marketing Science*, **15**(4), pp. 321–340.

[14] Kaymak, U. (2001) "Fuzzy Target Selection Using RFM Variables," in *IFSA World Congress and 20th NAFIPS International Conference*, vol. 2, Institute of Electrical and Electronics Engineers, pp. 1038–1043.

[15] Lilien, G. L., A. Rangaswamy, and A. D. Bruyn (2007) *Principles of Marketing Engineering*, DecisionPro, Inc.

[16] McFadden, D. (1986) "The Choice Theory Approach to Market Research," *Marketing Science*, **5**(4), pp. 275–297.

[17] Besanko, D., J.-P. Dube', and S. Gupta (2003) "Competitive Price Discrimination Strategies in a Vertical Channel Using Aggregate Retail Data," *Management Science*, **49**(9), pp. 1121–1138.

[18] Zhang, X., W. Gong, and Y. Kawamura (2004) *Customer Behavior Pattern Discovering with Web Mining*, Springer-Verlag.

[19] Song, H. S., J. K. Kim, and S. H. Kim (2001) "Mining the Change of Customer Behavior in an Internet Shopping Mall," *Expert Systems with Applications*, **21**, pp. 157–168.

[20] Changchien, S. W., C.-F. Lee, and Y.-J. Hsu (2004) "On-line Personalized Sales Promotion in Electronic Commerce," *Expert Systems with Applications*, **27**(1), pp. 35–52.

[21] Agrawal, R. and R. Srikant (1994) "Fast Algorithms for Mining Association Rules," in *Proceedings of the 20th VLDB Conference*, Santiago, Chile, pp. 487–499.

[22] ——— (1995) "Mining Sequential Patterns," in *Eleventh International Conference on Data Engineering* (P. S. Yu and A. S. P. Chen, eds.), IEEE Computer Society Press, Taipei, Taiwan, pp. 3–14.

[23] Dunn, J. C. (1973) "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," *Journal of Cybernetics*, (3), pp. 32–57.

[24] Bezdek, J. C. (1981) *Pattern Recognition With Fuzzy Objective Function Algorithms*, Plenum Press, New York.

[25] Kruse, R., C. Döring, and M.-J. Lesot (2007) "Fundamentals of Fuzzy Clustering," in *Advances in Fuzzy Clustering and its Applications* (J. V. de Oliveira and W. Pedrycz, eds.), chap. 1, John Wiley & Sons, pp. 3–30.

[26] Han, J. and M. Kamber (2001) *Data Mining: Concepts and Techniques*, Academic Press.

[27] Tanaka, K. (1997) *An Introduction to Fuzzy Logic for Practical Applications*, Rassel, Inc.

[28] Espinosa, J., J. Vandewalle, and V. Wertz (2005) *Fuzzy Logic, Identification, and Predictive Control*, Springer.

[29] Bult, J. R. and T. Wansbek (1995) "Optimal Selection for Direct Mail," *Marketing Science*, **14**(4), pp. 378–394.

[30] Banks, J. (ed.) (1998) *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, Wiley-IEEE.