RC23132 (W0403-021) March 2, 2004
Computer Science

IBM Research Report

Cross Channel Optimized Marketing by Reinforcement Learning

Naoki Abe, Naval Verma, Chid Apte
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Robert Schroko
Database Marketing
Saks Fifth Avenue
12 E. 49th Street
New York, NY 10017

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Cross Channel Optimized Marketing by Reinforcement Learning

Naoki Abe, Naval Verma and Chid Apte
Mathematical Sciences Dept.
IBM T. J. Watson Res. Ctr.
Yorktown Heights, NY 10598
nabe, nverma, apte@us.ibm.com

Robert Schroko
Database Marketing
Saks Fifth Avenue
12 E. 49th Street, New York, NY 10017
Robert_Schroko@s5a.com

ABSTRACT

The issues of cross channel integration and customer life time value modeling are two of the most important topics surrounding customer relationship management (CRM) today. In the present paper, we describe and evaluate a novel solution that treats these two important issues in a unified framework of Markov Decision Processes (MDP). In particular, we report on the results of a joint project between IBM Research and Saks Fifth Avenue to investigate the applicability of this technology to real world problems. The business problem we use as a testbed for our evaluation is that of optimizing direct mail campaign mailings for maximization of profits in the store channel. We identify a problem common to cross-channel CRM, which we call *the Cross-Channel Challenge*, due to the lack of explicit linking between the marketing actions taken in one channel and the customer responses obtained in another. We provide a solution for this problem based on old and new techniques in reinforcement learning. Experimental evaluation using actual customer interaction data show that as much as 7 to 8 per cent increase in the store profits can be expected, by employing a mailing policy automatically generated by our methodology. These results confirm that our approach is valid in dealing with the cross channel CRM scenarios in the real world.

Keywords

customer life time value, CRM, cost sensitive learning, reinforcement learning, targeted marketing

1. INTRODUCTION

The issues of cross channel integration and customer life time value modeling are undoubtedly two of the most important topics surrounding customer relationship management (CRM) today. Despite the wide-spread acknowledgement of their importance, there has not been a satisfactory solution in the market. In many cases, vendors provide infrastructure that enables cross channel integration of customer data, but provide no special-purpose analytics. Applying existing data analytics tools, however, will not fully leverage the integrated data. This is because existing tools do not help optimize decision making in the respective channels for maximization of future profits across different channels. In the present paper, we propose a novel solution that treats these two important issues in a unified framework.

The proposed solution is based on our earlier work on the application of reinforcement learning to sequential cost-sensitive decision making [9]. In that paper, it was demonstrated that by combining reinforcement learning and scalable data mining technologies, decision rules that are optimized with respect to long term benefits can be automatically generated solely from data analytics. We use variants of this basic technology to perform customer life time value modeling in a cross-channel setting, and optimize marketing actions with respect to long term, multi-channel profits.

We have conducted a joint study between IBM Research and Saks Fifth Avenue to investigate the applicability of this technology on a practical problem, which we will report on in the current paper. The business problem we elected as a testbed for our investigation is that of optimizing interactions between the direct mail channel and the store channel. Of the various channels of customer interactions that Saks Fifth Avenue owns and operates, such as the web, telemarketing, direct mailing and the store, we chose to focus on the interactions between the latter two channels, based on the relative ease of evaluation and readiness of the relevant data.

The largest obstacle that we faced in our effort is one that is characteristic of a cross-channel scenario, and is of general interest to most applications involving cross-channel interactions. The problem, which we call *the Cross Channel Challenge*, is the lack of explicit linking between the marketing actions taken in one channel and the responses (profits or rewards) obtained in another. This translates, in practice, to very low correlations observed between marketing actions and their effects across channels. Therefore, applying off-the-shelf regression methods to model the rewards as a function of various variables, is likely to produce a model that is independent of the marketing actions, thus leading to useless marketing rules.

Reinforcement learning is a framework which is well-suited to handling situations with the so-called *credit assignment* problem. That is, unlike the usual *supervised learning* methods, reinforcement learning can infer the relationship between actions and their effects from data, without explicit linking. As it turns out, cross-channel interactions in our problem are subtle enough that straightforward application of popular reinforcement learning techniques does not lead to a satisfactory solution. We managed to resolve this challenge by devising a number of modifications to the basic reinforcement learning technology. One of these modifications, based on an existing method of reinforcement learning called *advantage updating* [2], is particularly notable. It manages to solve the cross channel challenge by focusing on learning the *difference* in the effects on the rewards of competing actions, thereby bypassing the accurate estimation of the noisy reward function.

We conducted experimental evaluation of the proposed methods using actual customer interaction data from Saks Fifth Avenue. The results of our experiments suggest that we can expect as much as 7 to 8 per cent increase in the store profits by employing targeting rules automatically produced by our methodology, as compared to the current mailing policy used at Saks. These results seem to confirm that our approach is valid in dealing with the cross channel CRM scenarios in the real world.

The rest of the paper is organized as follows. We begin by describing the business problem that we address, in Section 2. Section 3 presents the methodology, including a brief review on the general reinforcement learning techniques, as well as detailed descriptions of newly devised methods. Section 4 will describe the experiments we conducted and the results we obtained. In Section 5, we briefly touch upon some deployment details including GUI software that we developed. Section 6 concludes with discussions on issues and future directions.

2. PROBLEM DESCRIPTION

2.1 The business problem

Saks Fifth Avenue utilizes various channels of customer interactions. These include the store, the direct mail, the web, telemarketing, and agents assigned to specific customers. We decided to focus on the store and direct mail channels for the following reasons:

1. The store channel accounts for the overwhelming majority of revenue at Saks Fifth Avenue.

2. The transaction data in the store channel are stored in data warehouses and are readily available for use in data analytics.

3. The direct mail campaign mailing data are available for data analytics.

4. The marketing actions in direct mailing are easier to quantify (for analysis) and to control (for optimization), as compared to those involving human interactions (telemarketing and agents).

We would like to point out that the problem of modeling cross-channel interactions has not been previously investigated at any depth. As a first study of its kind, the choice of particular channels is rather immaterial, since the findings concerning cross-channel interactions will mostly likelily generalize to those involving other channels.

The business problem we address, more specifically, is that of optimizing direct mail catalogue mailings over multiple campaigns, to maximize the effect on the profits/revenue obtained in the store channel. At Saks Fifth Avenue over 60 major direct mail campaigns are conducted each year. These campaigns vary from mailings of general store catalogues to those specific to particular product groups, such as women's apparel and cosmetics. Some are seasonal campaigns, such as Christmas season campaigns. Some may involve store coupons, while others may provide information on upcoming sales and their contents and durations. Many of these campaign features are available for use in data analysis.

Currently, the generation of mailing list for each of these campaigns is based on a number of criteria and constraints, and is not fully automated. There are intricate issues surrounding the process of generating these mailing lists. Our goal is not neccesarily a full and immediate automation of this process, but rather in demonstrating the potential use of sophisticated data analytics in assisting and improving it. As a step in this endeavor, our *technical task* is to analyze the past data and automatically generate targeting rules that can be used to construct mailing lists for future campaigns.

The data we used for this task are categorized into the following four types: (1) customer data, (2) transaction data from the store channel, (3) campaign data from the direct mail channel, and (4) product data.

These data were used to generate feature vectors for each of the customers, which contains the summarized history of cross-channel interactions. These features include information on both the actions taken by Saks Fifth Avenue (i.e. mailings), and customers' responses (i.e. store transactions). It is reasonable to suppose, therefore, that by running data analysis on these data, the interactions between the actions and responses can be modeled, and rules for optimizing the mailing actions can be derived. More precisely, our task is to do this modeling with respect to the cross-channel, long term profits, and then use the resulting model to generate targeting rules for the direct mail campaign mailings.

2.2 The cross channel challenge

The problem just described is a challenging one, and turns out not to be solvable by straightforward applications of existing modeling techniques. For example, the simplest solution would be to model the short term profits (or revenues) in the store generated by a particular customer, say in a window of one month, as a function of various features of that customer, including the control variable of whether a given catalogue is mailed to that customer. As we elaborate in the section on experiments, this will result in a model that is independent of the control variable, thus giving no interesting information on the effect of the mailing action.

At the heart of the problem is the *credit assignment* problem. That is, there is no explicit information in the data, linking the actions taken in the direct mail channel to the responses observed in the store channel. This would be possible, if a good part of the transactions were associated with coupons issued in the outbound channel of interest, and this fact were recorded. This is rarely the case in practice, and in particular is not the situation we face in our current problem.

As we remarked earlier, reinforcement learning is in general well-suited for dealing with this problem, since it is designed to handle implicit and delayed form of credit assignment. It does so by keeping track of episodes, i.e. sequences of actions and rewards over time, and essentially crediting all actions preceding given rewards in the episodes, with varying degrees. It turns out, however, that the cross-channel interaction we wish to model is subtle enough that straightforward application of representative reinforcement learning methodology does not adequately solve the problem.

To place further burden on the modeling task, our problem setting involves events with variable length time intervals, i.e. the intervals between the decision points (campaign mailings) are variable in length. This adds considerable amount of noise in the data, and makes the task of modeling the responses of marketing actions even more difficult.

The following list summarizes the challenges we face:

1. *Credit assignment* between marketing actions (in the direct mail channel) and the customers' responses (in the store channel) is implicit.
2. The effect of marketing actions on the responses is subtle and difficult to model directly.
3. The intervals between decision points (campaign mailings) are variable, adding another source of noise.

3. METHODOLOGY

3.1 Cross channel life time value maximization and MDP

Common practice in database marketing and CRM today is to organize customer data into a table consisting of fields representing various attributes of customers and response fields, model a response field of interest as a function of those attributes, and then optimize marketing actions against the obtained model. Here we go a step further: We use time stamped sequences of such data to represent time-varying sequences of customer's attributes (state) and marketing actions. We then model the process of customer interactions as a dynamic process, and optimize marketing actions with respect to such a model. The technical framework we employ is the so-called Markov Decision Process (MDP) model, popular in dynamic programming and reinforcement learning. Here we give a brief description of MDP, referring the reader to the literature for more details, e.g. [6, 10].

For simplicity, we assume a discrete time model. At any point in time, the process is assumed to be in some state s. At each time step (in the discrete case), the agent/learner takes an action a, and receives a reward r, and the process makes a transition to another state s'. The reward r and the transition state s' are both determined by probability distributions that depend on the state s and action a.

The process starts in some initial state s_1 and the learner is to repeatedly take actions, resulting in a sequence of action, state and reward triples, $\{(s_t, a_t r_t)\}_{t=1}^{\infty}$. The goal of the learner is to maximize the total rewards accrued over time, usually with future rewards discounted. That is, it is to maximize the cumulative reward R,

$$R = \sum_{t=1}^{\infty} \gamma^{t-1} r_t \tag{1}$$

where γ is a positive constant less than 1, called the discount factor.

We assume that the learner takes its actions based on a certain *policy*, namely a function π that maps states to actions. It is well-known that for any MDP, an optimum policy π^* exists, which maximizes the cumulative reward as defined above.

In our application, we can use the attribute vector corresponding to a customer at a given point in time to represent the state for that customer at that point in time. Cross-channel integration of data allows us to represent the entire history of interactions between a given customer and the enterprise, across all channels, thereby providing a unified view of the customer. The maximization of life time value of a given customer, across all channels, can then be naturally formulated as the maximization of the cumulative profit, as defined by Eq. 1. It follows that life time value maximization in the cross-channel setting is reduced to solving for the optimum policy of the MDP with the cross-channel state representation. Since the obtained policy is a mapping from customer attribute vectors to actions, it can be readily translated to generic if-then style rules for use in any rules engine.

3.2 Batch reinforcement learning

While it is easy to describe MDP and the optimum solution for it, solving for the optimum policy or an approximation thereof is a challenging problem. A big portion of the field of *reinforcement learning* is concerned with this problem, and many competing approaches and methodologies have been developed. Below we briefly review some of the basic theory known in the literature.

We begin by defining a fundamental notion in an MDP known as the *value function* Q^π of a policy π. A value function maps a state s and an action a to the expected value of the cumulative reward that would be obtained if the process started in state s, and the learner performed action a and then followed policy π ever after. Formally, $Q^\pi(s, a)$ is defined as

$$Q^\pi(s,a) = E_\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \;\middle|\; s_1 = s, a_1 = a \right] \tag{2}$$

where E_π denotes the expectation with respect to the policy π that is used to define the actions taken in all states except the initial state s_1.

A remarkable property of Markov Decision Processes is that the value function Q^* of an optimum policy π^* satisfies the following recurrence relation, known as the *Bellman optimality equation*:

$$Q^*(s,a) = E_r[r \mid s,a] + \gamma E_{s'}\left[\max_{a'} Q^*(s',a') \;\middle|\; s,a\right] \quad (3)$$

where $E_r[r \mid s,a]$ is the expected immediate reward obtained by performing action a in state s, and $E_{s'}[\max_{a'} Q^*(s',a') \mid s,a]$ is the expected cumulative reward when performing the optimum action in the state s' that results when action a is performed in state s, and thereafter.

The Bellman equation can be solved via fixed-point iteration, known as *value iteration*, essentially by initializing the value function with the expected immediate rewards, and updating the estimate by iterative pplications of Eq. 3. Note, however, that this iteration process requires knowledge of both the expected immediate rewards and the state transition probabilities (for calculating the expectation on the right hand side). Since these are unknown in real world applications, they need to be estimated from data. In the application to database marketing and CRM, this estimation can be naturally performed by applying a regression engine on the past data, which is the essence of the proposal in [9].

Pednault et al [9] describe several variants of this general scheme, which are collectively referred to as *batch reinforcement learning with function approximation*. Here, batch reinforcement learning refers to a form of reinforcement learning in which the learning does not take place in an on-line fashion, but rather makes use of static training data that have been collected through prior experience. Function approximation means that the value function is represented as a function of state features and actions, rather than that of atomic states and actions (see, for example, [3, 11, 12]). Here, we review a representative method of this type, based on a popular (on-line) reinforcement learning method known as *Q-learning* [13]. Figure 1 presents a pseudo-code for this method, following [9].

Procedure Batch-RL(Q)
Premise:
A base learning module, Base, for regression is given.
Input data: $D = \{e_i | i = 1, ..., N\}$ where
$e_i = \{\langle (s_{i,j}, a_{i,j}, r_{i,j}\rangle | j = 1, ..., l_i\}$
(e_i is the i-th episode, and l_i is the length of e_i.)
1. For all $e_i \in D$
$D_i^{(0)} = \{\langle (s_{i,j}, a_{i,j}), r_{i,j}\rangle | j = 1, ..., l_i\}$
2. $Q^{(0)} = \text{Base}(\bigcup_{i=1,...,N} D_i^{(0)})$
3. For $k = 1$ to K
3.1 Set α_k, e.g. $\alpha_k = \frac{1}{k}$
3.2 For all $e_i \in D$
For $j = 1$ to $l_i - 1$
$v_{i,j}^{(k)} = (1-\alpha_k) Q^{(k-1)}(s_{i,j}, a_{i,j}) + \alpha_k (r_{i,j} + \gamma \max_a Q^{(k-1)}(s_{i,j+1}, a)$
$D_i^{(k)} = \{\langle (s_{i,j}, a_{i,j}), v_{i,j}^{(k)}\rangle | j = 1, ..., l_i - 1\}$
3.3 $D^{(k)} = \bigcup_{i=1,...,N} D_i^{(k)}$
3.4 $Q^{(k)} = \text{Base}(D^{(k)})$
4. Output the final model, $Q^{(K)}$.

Figure 1: Batch reinforcement learning based on Q-learning.

3.3 Q-learning with variable time intervals

As we mentioned in the Introduction, for our problem it is necessary to extend the MDP framework to a formulation involving variable time intervals. The *variable time interval MDP* we consider here is identical to the discrete time MDP, except that every event is timed. Here we assume that the time at the initial state is 0, and then all subsequent events will have positive time associated with them. The process starts in some initial state s_1 at $t_1 = 0$, and then the learner repeatedly takes actions, resulting in a sequence of action, state, reward and time quadruples, $\{(s_i, a_i, r_i, t_i)\}_{i=1}^{\infty}$. The goal of the learner is then defined as the maximization of the total discounted rewards, with the discounted factors determined as a function of the time durations. That is, it is to maximize the cumulative reward R,

$$R = \sum_{i=1}^{\infty} \gamma^{t_i} r_i \quad (4)$$

We note that the model we introduce here is different from and simpler than the extension of MDP to the continuous time setting e.g. SMDP proposed by [4], in that we still assume discrete time steps, though having variable interval length.

While the changes in the definitions of MDP and the goal of learning for the variable time interval set-up are relatively simple, how the changes should be reflected in the corresponding learning methods is less clear. Part of the difficulty is in determining how the rewards in various time intervals should be normalized, in order to lessen the effect of noise introduced by the varying interval length. For the immediate rewards, it is clear that the reward received in a time interval can be normalized by dividing it by the time interval. For the value function, the situation is significantly more complicated. In particular, we must account for the fact that, at any learning iteration, the interval over which the discounted rewards is summed (for approximating the value function) is incremented. Furthermore, the effective interval is affected by the learning rate, α_k, since a larger α_k corresponds to assigning greater weights to the rewards received far into the future. Here we propose a normalization scheme in which both of these factors are taken into account in determining the effective time interval to be used for normalization. More specifically, we use an analogous update rule for the normalization factor, as that for the Q-value estimate. The resulting variant of the batch Q-learning method for the variable time interval set-up, Vari-RL(Q), is presented in Figure 2. (The update rules for both Q-values, $v_{i,j}^{(k)}$, and normalization factors, $Z_{i,j}^{(k)}$, are in block 4.2 of the pseudo-code.)

3.4 Batch Advantage Updating

A number of past papers have addressed the problem of how to extend Q-learning and other related learning methods to variable time intervals and more generally to the continuous time setting, e.g. [2, 4].Of these, the work by Baird [2] is of particular interest to us, since his solution appears to

Procedure Vari-RL(Q)
Premise:
A base learning module, Base, for regression is given.
Input data: $D = \{e_i | i = 1, ..., N\}$ where
$e_i = \{\langle s_{i,j}, a_{i,j}, r_{i,j}, t_{i,j}\rangle | j = 1, ..., l_i\}$
(e_i is the i-th episode, and l_i is the length of e_i.)
1. For all $e_i \in D$
1.1 For $j = 1$ to l_i, $\Delta t_{i,j} = t_{i,j+1} - t_{i,j}$
2. For all $e_i \in D$
2.1 For $j = 1$ to $l_i - 1$
$Z_{i,j}^{(0)} = \Delta t_{i,j}$
$v_{i,j}^{(0)} = r_{i,j}$
$D_i^{(0)} = \{\langle (s_{i,j}, a_{i,j}), \frac{v_{i,j}^{(0)}}{Z_{i,j}^{(0)}}\rangle | j = 1, ..., l_i\}$
3. $\widetilde{Q}^{(0)} = \text{Base}(\bigcup_{i=1,...,N} D_i^{(0)})$
4. For $k = 1$ to K
4.1 Set α_k, e.g. $\alpha_k = \frac{1}{k}$
4.2 For all $e_i \in D$
For $j = 1$ to $l_i - 1$
$Z_{i,j}^{(k)} = (1 - \alpha_k) Z_{i,j}^{(k-1)} + \alpha_k (\Delta t_{i,j} + \gamma^{\Delta t_{i,j}} \cdot Z_{i,j+1}^{(k-1)})$
$v_{i,j}^{(k)} = (1 - \alpha_k) \widetilde{Q}^{(k-1)}(s_{i,j}, a_{i,j}) \cdot Z_{i,j}^{(k-1)}$
$+\alpha_k (r_{i,j} + \gamma^{\Delta t_{i,j}} \max_a \widetilde{Q}^{(k-1)}(s_{i,j+1}, a) \cdot Z_{i,j+1}^{(k-1)})$
$D_i^{(k)} = \{\langle (s_{i,j}, a_{i,j}), \frac{v_{i,j}^{(k)}}{Z_{i,j}^{(k)}}\rangle | j = 1, ..., l_i - 1\}$
4.3 $\widetilde{Q}^{(k)} = \text{Base}(\bigcup_{i=1,...,N} D_i^{(k)})$
5. Output the final unnormalized model, i.e. $\widetilde{Q}^{(K)} \cdot Z^{(K)}$.

Figure 2: Variable time interval batch Q-learning.

be addressing closely related problems to the one we currently face: function approximation in Q-learning involving variable time intervals is often difficult due to the noise introduced by the varying intervals. Baird proposes a novel method, called *advantage updating*, which tries to learn the relative advantage of competing actions in any given state. This procedure avoids having to explicitly estimate the Q-value function, thereby bypassing the noisy estimation problem. We briefly describe this method and some modifications we made to make it work for our problem.

Advantage updating is based on the notion of *advantage* of an action a relative to the optimal action at a given state s, written $A(s, a)$. The following is one of the definition of this quantity.

$$A^*(s,a) = \frac{1}{\Delta t_s}(r_{i,j} + \gamma^{\Delta t_s} E_{s'}[V^*(s')] - V^*(s)) \quad (5)$$

In the above, note that V^* is defined as $V^*(s) = \max_a Q^*(s, a)$, where $Q^*(s, a)$ is the Q-value of an optimal policy. We used Δt_s to denote the time interval immediately following state s. Alternatively, the advantage can be written in terms of the Q-value by:

$$A^*(s,a) = \frac{1}{\Delta t_s}(Q^*(s,a) - \max_{a'} Q^*(s,a')) \quad (6)$$

This quantity is extremely interesting for us for two reasons: It factors out the dependence of the value function on the time interval, *and* on the state. Given this notion of advantage, *advantage updating* is an on-line learning method that learns this function iteratively, by a coupled set of update

Procedure Batch-AU
Premise:
A base learning module, Base, for regression is given.
Input data: $D = \{e_i | i = 1, ..., N\}$ where
$e_i = \{\langle s_{i,j}, a_{i,j}, r_{i,j}, t_{i,j}\rangle | j = 1, ..., l_i\}$
(e_i is the i-th episode, and l_i is the length of e_i.)
1. For all $e_i \in D$
1.1 For $j = 1$ to l_i, $\Delta t_{i,j} = t_{i,j+1} - t_{i,j}$
2. For all $e_i \in D$
$D_i^{(0)} = \{\langle (s_{i,j}, a_{i,j}), \frac{r_{i,j}}{\Delta t_{i,j}}\rangle | j = 1, ..., l_i\}$
3. $A^{(0)} = \text{Base}(\bigcup_{i=1,...,N} D_i^{(0)})$
4. For all $e_i \in D$ and for $j = 1$ to $l_i - 1$, initialize
4.1 $A_{i,j}^{(0)} = A^{(0)}(s_{i,j}, a_{i,j})$
4.2 $Amax_{i,j}^{(0)} = \max_{a'} A^{(0)}(s_{i,j}, a')$
4.3 $V_{i,j}^{(0)} = Amax_{i,j}^{(0)}$
5. For $k = 1$ to K
5.1 Set α_k, β_k and ω_k, e.g. $\alpha_k = \beta_k = \omega_k = \frac{1}{k}$
5.2 For all $e_i \in D$
For $j = 1$ to $l_i - 1$
$A_{i,j}^{(k)} = (1 - \alpha_k) A_{i,j}^{(k-1)}$
$+\alpha_k (Amax_{i,j}^{(k-1)} + \frac{r_{i,j} + \gamma^{\Delta t_{i,j}} V_{i,j+1}^{(k-1)} - V_{i,j}^{(k-1)}}{\Delta t_{i,j}})$
$D_i^{(k)} = \{\langle (s_{i,j}, a_{i,j}), A_{i,j}^{(k)}\rangle | j = 1, ..., l_i - 1\}$
5.3 $A^{(k)} = \text{Base}(\bigcup_{i=1,...,N} D_i^{(k)})$
5.4 For all $e_i \in D$ and for $j = 1$ to $l_i - 1$, update
$A_{i,j}^{(k)} = A^{(k)}(s_{i,j}, a_{i,j})$
$Amax_{i,j}^{(k)} = \max_{a'} A^{(k)}(s_{i,j}, a')$
$V_{i,j}^{(k)} = (1 - \beta_k) V_{i,j}^{(k-1)}$
$+\beta_k (\frac{Amax_{i,j}^{(k)} - Amax_{i,j}^{(k-1)}}{\alpha_k} + V_{i,j}^{(k-1)})$
5.5 For all $e_i \in D$ and for $j = 1$ to $l_i - 1$, normalize
$A_{i,j}^{(k)} = (1 - \omega_k) A_{i,j}^{(k)} + \omega_k (A_{i,j}^{(k)} - Amax_{i,j}^{(k)})$
6. Output the final advantage model, $A^{(K)}$.

Figure 3: Batch reinforcement learning based on advantage updating.

rules for the estimates for A and V, and a normalization step for $A^*(s, a)$ which drives $\max_{a'} A^*(s, a')$ towards zero. We exhibit a batch version of this method in Figure 3.

3.5 Modifications

A couple of other modifications were necessary, before the two methods just presented, Vari-RL(Q) and batch-AU, could be made to work satisfactorily. One modification has to do with the initialization of the quantities being estimated in the two methods, and the other has to do with optional applications of function approximation. We will describe these two modifications in some detail below.

The first modification concerns the initialization of the Q-value and A-value estimates, respectively. In the generic description methods, given in Figures 2 and 3, these values were initialized by the normalized immediate rewards, that is, the rewards observed in the time interval immediately following the decision point in question. Here, normalization is intended to lessen the noise introduced by the large variations in the time intervals between decision points, and thus lighten the burden on the function approximator (re-

gression engine). Recall that both of these estimates are variants of normalized Q-values, which are quantities defined over a (discounted) sum over indefinitely many future steps. In an on-line learning setting, in which one observation is made at a time, initializing the estimates with the (normalized) immediate rewards observed in the first time interval is inevitable. In a batch learning setting, in which we have access to the entire data over multiple time intervals, this is no longer necessary. It is thus natural to consider initializing them using empirical estimates of Q and A-values over longer time periods. Specifically, we can initialize them using the *empirical life time value (LTV)*, or a discounted sum of rewards observed in the data.

Initialization of Q-value estimates by the empirical LTV in Vari-RL(Q) can be done by replacing the initialization step in block 2 by the following: For all $e_i \in D$

$$v_{i,j}^{(0)} = \sum_{j=1}^{l_i-1} \gamma^{t_{i,j}} r_{i,j}$$

$$Z_{i,j}^{(0)} = \sum_{j=1}^{l_i-1} \gamma^{t_{i,j}} \Delta t_{i,j}$$

Likewise, the initialization of A-value estimates by the empirical LTV in batch-AU can be done by replacing block 2 by the following: For all $e_i \in D$

$$v_{i,j}^{(0)} = \sum_{j=1}^{l_i-1} \gamma^{t_{i,j}} r_{i,j}$$

$$D_i^{(0)} = \{\langle (s_{i,j}, a_{i,j}), \frac{v_{i,j}^{(0)}}{\Delta t_{i,j}} \rangle | j = 1, ..., l_i\}$$

Note that the normalization factor used in the batch-AU procedure corresponds to the immediate time interval following the state in question, and not one corresponding to the discounted cumulative sum as in Vari-RL(Q).

The second modification we made is that of optional application of function approximation. Function approximation refers to the application of the base regression engine as part of the process for estimating the Q-values and A-values, respectively, in the two methods. Function approximation serves to smooth the estimation, and allows us to calculate the estimates for unseen events, which is necessary to apply maximization over the actions to the Q-value and A-value estimates. It also contributes, however, to accumulation of noise and can lead to worsen the quality of estimation, especially if learning is performed over a large number of iterations. In order to address the trade-off between these two effects of function approximation, one is naturally led to balancing them by allowing optional applications of function approximation. Specifically, we let our methods take an extra parameter n specifying that function approximation is to be performed at every n-th iteration. This can be done by modifying a few lines in the two procedures. Specifically, for Vari-RL(Q), line 4.3 in Figure 2 is to be replaced by the following, in every iteration in which function approximation is not to be applied.

$$\widetilde{Q}^{(k)} = \widetilde{Q}^{(k-1)}$$

Similarly, for Batch-AU, line 5.3 in Figure 3 should be replaced by

$$A^{(k)} = A^{(k-1)}$$

and the update statement for $A_{i,j}^{(0)}$ should be removed, in all non-function approximating iterations.

4. EXPERIMENTS

We evaluated the proposed methods using actual customer interaction data from Saks Fifth Avenue from the past years. Below we will describe the data we used for this experimentation in some detail. Then we will discuss the challenge we face in evaluating the methods using past data. We then describe the experimental results.

4.1 Data

As briefly explained in Introduction, the data we used for our data analysis can be categorized into the following four types.

1. *Customer data* for 1.6 million customers, whose recent annual spendings exceeded a certain threshold. These contain demographic and other types of information on the customers. We note that privacy sensitive information was stripped off before they were used for data analytics.

2. *Transaction data* for the said 1.6 million customers for the past three years. The transaction data are time stamped and contain the entire point of sales data, including the categories of purchased items and sales price, among other things.

3. *Campaign data* for the major campaigns for the year 2002. There were 69 campaigns. These data contain timing of mailing, duration of sales, types of catalogues sent, and product groups (divisions) being targeted by the campaigns.

4. *Product data* for the purchased items in the transactions data. These contain taxonomy information on them, ranging in granularity from product groups down to the SKU level.

These data were then used to generate time stamped sequences of feature vectors containing summarized information on the history of cross-channel interactions. The features we generated and elected to use, representing the state of a customer at any given point in time, are summarized in Table 1.

The features are also divided into four types: (1) customer features; (2) transaction features; (3) campaign features; and (4) product group specific campaign features. Initially we synthesized many more features (exceeding one hundred), but chose a small subset of them based on correlation analysis and the resulting quality of function approximation.

The customer features are simply a subset of the customer information stored in the data warehouse at Saks. The transaction features were calculated by joining the store transaction history (p.o.s.) data with the customer data.

Features	Descriptions	Cor.
full_line_store_of_residence	if a full-line store exists in the area	0.004
off_fifth_store_of_residence	if an off-fifth store exists in the area	-0.004
loyalty_program_level	loyalty program level	0.074
fvrt_store_channel	favorite store channel (web/store)	0.046
purchase_amt_1m	amount of purchase in last month	0.085
purchase_amt_2_3m	amount of purchase in last 2-3 month period	0.098
purchase_amt_6m	amount of purchase in last 4-6 month period	0.102
purchase_amt_1y	amount of purchase in last year	0.139
purchase_amt_tot	total amount of purchase (in 3 years)	0.155
promo_purchase_ratio	ratio of purchases in promotion periods	0.047
cur_div_purchase_amt_1m	purchase amount last month in current division	0.090
cur_div_purchase_amt_2_3m	purchase amount last 2-3 month in current division	0.080
cur_div_purchase_amt_6m	purchase amount last 4-6 month in current division	0.091
cur_div_purchase_amt_1y	purchase amount last 1 year in current division	0.128
cur_div_purchase_amt_tot	total purchase amount in current division	0.147
div_purchase_amt_tot_j	total purchase amount in division j	0.093*
n_cat_1m	number of catalogues sent in last month	0.021
n_cat_2_3m	number of catalogues sent in last 2-3 month	0.028
n_cat_4_6m	number of catalogues sent in last 4-6 month	0.063
n_cat_tot	total number of catalogues sent	0.086
cur_div_n_cat_1m	number of catalogues sent in last month targeting current division	0.028
cur_div_n_cat_2_3m	number of catalogues sent in last 2-3 month targeting current division	0.025
cur_div_n_cat_4_6m	number of catalogues sent in last 4-6 month targeting current division	0.062
cur_div_n_cat_tot	total number of catalogues sent targeting current division	0.062
action	to mail or not to mail	0.008

Table 1: Features used in our experiments: Features, their descriptions, and their correlations with the reward field. The features are listed in 4 groups: (1) customer features; (2) transaction features; (3) campaign features; and (4) product group specific campaign features. (* An example correlation value is exhibited for several features of this type.)

The campaign features were constructed using the per campaign mailing lists, again joining them with the customer data. the product group specific campaign features were synthesized using both campaign data and taxonomy information in the product data. Of the many levels of product information available, we elected to use the coarsest categorization in terms of *divisions*, of which there are nine.

The group of features, *current division number of catalogues, etc*, in the last category, are calculated by counting the effective number of catalogues sent in the past, targeting products in the division that is targeted by the current campaign. Since a given campaign can target products in multiple divisions, a weight vector was computed for each campaign, which assigns a relative weight to each division. The effective number of past catalogues targeting similar product divisions as the current catalogue was then calculated as the sum of the inner products between the weight vector of the current catalogue and those of the catalogues already sent.

Note that as the third column in Table 1 the correlation coefficients between each of the features and the response variable (reward) are listed. Here, the response variable was calculated simply by summing the observed profits in the data, over a fixed period of time since the time of the mailing in question. Therefore it does not exactly correspond to the *cumulative discounted profits* that we wish to maximize via reinforcement learning. It is worth noting, nonetheless, that a very low correlation is observed between the control variable (mailing action) and the response variable, as compared to some of the other features. The control variable has the third lowest correlation coefficient (at 0.008), and is magnitudes lower than those of typical (transaction and campaign) features. This explains the nature of *the cross-channel challenge*, and in particular why we tend to get a model that is independent of the control variable, when we run a standard regression engine to model the response variable as a function of these features.

4.2 Evaluation

A common problem in performance evaluation of reinforcement learning methods is that often it is difficult or impossible to conduct real life experiment in which the learning methods have access to on-line interactions with the MDP. Our current application domain of CRM and database marketing is no exception. We rarely have the luxury to test the performance of a learning method in experimentation involving live customer interactions. It is almost always necessary to validate the performance in an in-lab evaluation before real life deployment. To conduct such performance evaluation reliably using only static historical data is itself a big challenge, however. The problem is that we need to evaluate the policy generated by the learning procedure using only past data, which presumably were collected using some policy that is different from it.

Here we propose a solution to this problem, and use it to conduct performance evaluation for our methodology. Our solution is based on a notion recently proposed by Kakade and Langford [7] called *policy advantage*, and a bias correc-

tion technique based on importance sampling, c.f. [15]. We elaborate on this below.

First, we define a discrete time version of the notion of *advantage* introduced earlier (in Eq. 6), with respect to any policy π.

$$A_\pi(s,a) = Q_\pi(s,a) - \max_{a'} Q_\pi(s,a') \quad (9)$$

Then the policy advantage of a new policy π' with respect to an old (or sampling) policy π and initial state distribution μ, written $A_{\pi,\mu}(\pi')$, is defined as follows.

$$A_{\pi,\mu}(\pi') = E_{s\sim\pi,\mu}[E_{a\sim\pi'(a|s)}[A_\pi(s,a)]] \quad (10)$$

Intuitively, the policy advantage measures how much advantage can result by replacing the action of the old policy by that of the new policy at a random state selected by the sampling policy, while all other actions remain unchanged (specified by the sampling policy). In some sense, this measure quantifies how much local improvement is attained by changing the old policy by the new policy, assuming that the overall state distribution is not significantly affected by that change.[1]

The policy advantage can be estimated using only data collected by the sampling policy π, using a bias correction technique based on importance sampling as follows.

$$A_{\pi,\mu}(\pi') = E_{s,a\sim\pi,\mu}[\frac{\pi'(a|s)}{\pi(a|s)}[A_\pi(s,a)]] \quad (11)$$

Note that $\pi'(a|s)$ is known since it is the (possibly stochastic) policy generated by reinforcement learning, but $\pi(a|s)$ need be estimated from the data, since we do not know the sampling policy explicitly. It should be pointed out that this quantity becomes impossible to estimate for a deterministic sampling policy, since the bias correction factor $\frac{\pi'(a|s)}{\pi(a|s)}$ diverges for actions a never chosen by the sampling policy π. In real world settings, however, the state information is often not sufficient to determine the chosen action deterministically, as is the case in our setting.

4.3 Experimental Results

We used the evaluation method just introduced, namely that of bias corrected estimation of policy advantage in the data, to validate the performance of the proposed methods. We used both of the proposed methods, Vari-RL(Q) and Batch-AU. In both cases, we used IBM's scalable regression engine, ProbE^{TM} [8, 1], as the basic regression module. For both methods, we used the feature to initialize the Q-value and A-value estimates by the empirical life time values. Also, for both methods, we used the option of applying function approximation at every fourth learning iteration. In our evaluation, we randomly sampled approximately 1.0 percent of the individual customers from the entire data (approximately 16 thousand customers from 1.6 million). The episodic data were then generated, for use in training, by randomly selecting a sub-episode of length 10 (consisting of 10 events) corresponding to each of the sampled individuals. A separate test data set consisting of 5,000 randomly selected individuals was also sampled for calculating the policy advantage. The policy advantage was calculated using these individual data over the entire 68 campaigns, and for each learning iteration, the results were averaged over 10 random runs.

The results of this evaluation are exhibited in Figures 4 and 5. Figure 4 plots how the policy advantage of the policy output by the Vari-RL(Q) method changes as the learning iteration progresses in a typical run. Figure 4 shows the analogous graph for the batch-AU method. In each of the graphs, the y-axis is the policy advantage shown as the percentage over the value of the old policy. Strictly speaking, what is shown on the x-axis is not the number of iterations, but rather is the number of times function approximation is performed. In both cases, we chose to run function approximation at every fourth learning iteration.

A definite trend can be read off from these graphs. For both of the methods, a typical run starts with a policy that is relatively uninformative which does not show any advantage over the sampling policy. It is worth noting that, since both methods were initialized with the empirical life time value observed in the data, this shows that direct modeling with empirical LTV does not lead to any advantage over the existing mailing policy. At the third function approximation, or after 9 learning iterations, the policy advantage peaks and then it starts declining again. This behavior is thought to be attributable, in part, to the nature and limitation of the evaluation method. Policy advantage measures the advantage of a new policy with respect to an old policy, using the old policy as the sampling policy. It is therefore more effective when the two policies are relatively similar. As the learning progresses and the two policies start diverging, the measure becomes less and less reliable. Note that in an on-line reinforcement learning set-up, for which the notion of policy advantage was proposed, one can use the newly learned policy as the sampling policy to evaluate the next policy and iterate such a process. This is emphatically not possible in the batch setting.

Even with the limitation mentioned above, the obtained results are quite encouraging. With the assumption that the new policy does not significantly change the state distribution, the results imply that as much as 7 to 8 percent increase in the store profits can be expected, by employing the policy output by our methodology, over the current mailing policy used at Saks.

5. THE CCOM SYSTEM

5.1 System overview and functionalities

While working on validating the proposed methodology on real world data, we have also developed a solution based on this technology, which we call CCOM (Cross-Channel Optimized Marketing). The CCOM system consists of the following components.

1. *Data Preparation module*, which accesses data stored in databases, via JDBC interface, and generates episodic data suitable for applications of the batch reinforcement learning engine.

[1] Kakade and Langford [7] have established a theoretical result which implies that a new policy with a positive policy advantage can be used to define a new policy, which provably has better performance than the old policy.

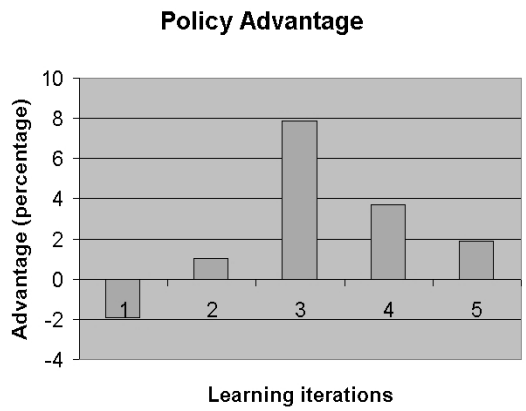

Figure 4: The policy advantage for the temporal Q-learning method (in a typical run.)

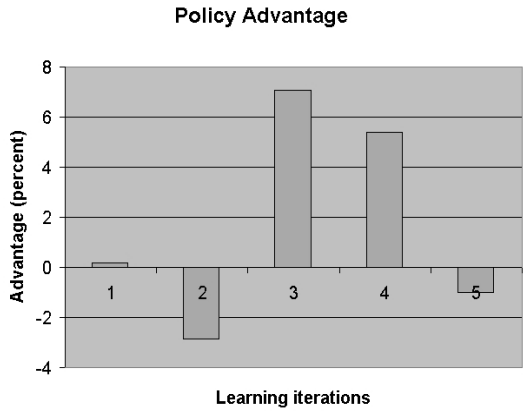

Figure 5: The policy advantage for the Batch Advantage Updating method (in a typical run.)

2. *Learning module*, which takes the training data from the data preparation module, and runs batch reinforcement learning on them. This module includes a scalable regression engine, ProbE^{TM} [8, 1], as the basic regression module used in the respective reinforcement learning procedure.

3. *Rule Mapping module*, which takes the output regression model from the learning module and transforms it into a set of if-then style rules.

4. *Profiling and Scoring module*, which periodically access the customers' interaction history data, and constructs updated profiles for them, namely the current feature vectors. These feature vectors are fed to the rules to perform scoring.

5. *Graphical User Interface module*, which allows data analysts to use the CCOM system, for example, run the learning module specifying various parameters, view the obtained rules, and perform scoring.

We note that the data preparation module and the profiling module, which are closely related, need be customized for each new application. This is because the exact choice of features for use in data analysis is highly application dependent, and cannot be determined *a priori*. The generality of the CCOM system is maintained, however, by localizing the need of customization to these modules.

The rule mapping process, in and of itself, is a technically interesting problem. In essence, the problem is to translate a regression model representing the Q-value function $Q(s, a)$ for Vari-RL(Q), or the advantage function $A(s, a)$ for Batch-AU, into a deterministic policy π that maps states to actions. We do this by a procedure akin to MetaCost due to Domingos [5]. That is, given a training data consisting of (s, a) pairs, we apply the Q-value model Q, and relabel the training data as $(s, \arg\max_a Q(s, a))$ (and analogously for A.) We then run a classifier on the relabeled data set. As long as we use classifiers with interpretable hypotheses, such as decision trees or rules, the output classifier can be readily translated into if-then rules for use in rules engines. Our experience shows that this translation process hardly compromises the predictive quality of the obtained policy.

5.2 Graphical user interface

We briefly explain the GUI component of CCOM, since it also serves to give an intuitive overview of the functions of CCOM. The CCOM GUI consists of four distinct panes, data preparation, learning, rule display, and evaluation. The respective pane lets the data analyst run various modules of the CCOM system. For example, Figure 6 exhibits an example screen shot of the learning pane. It allows the user to specify various parameters of the learning process, such as the number of iterations, the discount (decay) constant, the learning rate, etc. It is also possible to specify other parameters such as the number of customers to be sampled, and the names of certain files containing customized information. Figure 7 shows an example of how the rules display pane looks like. It allows the user to display the obtained rules and inspect them for gaining marketing insight. The rules can be sorted with respect to criteria such as the accuracy and coverage of the displayed rules.

6. CONCLUSIONS

We validated our reinforcement learning based approach to life time value modeling and cross-channel optimized marketing on a real world problem. In the course of our investigation, we identified a general problem common to modeling cross-channel interactions, and proposed a solution based on old and new techniques of reinforcement learning. We also provided a solution to the sampling bias problem in evaluation of learned policies, and used it to evaluate the proposed approach. The results of our experimental evaluation confirm the applicability of our approach to real world cross-channel scenarios. We also developed software that is readily applicable to other application problems. Some issues for further investigation include the following: (1) Easing deployment by reducing the need to customize; (2) Handling various channel constraints including budget constraints; (3) Comparing the performance of alternative methodologies, for example, an approach based on planning [14]

7. ACKNOWLEDGMENTS

We wish to thank Bill Franks and Sheri Wilson-Gray of Saks Fifth Avenue for their executive leadership in making the joint project possible. We also wish to thank Edwin

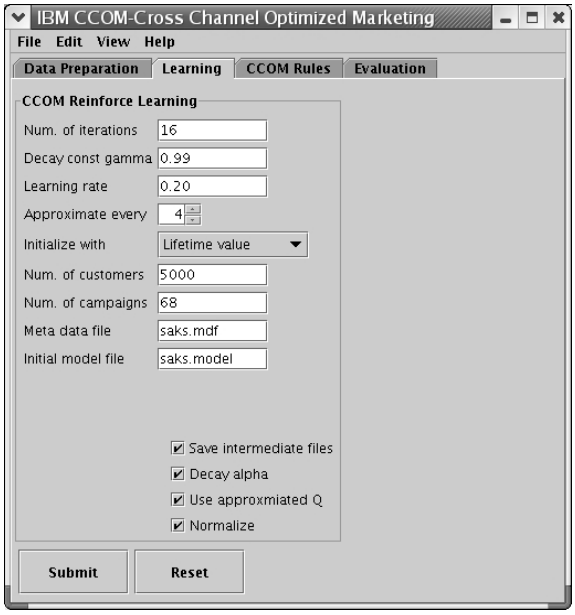

Figure 6: The CCOM GUI for controlling the learning process.

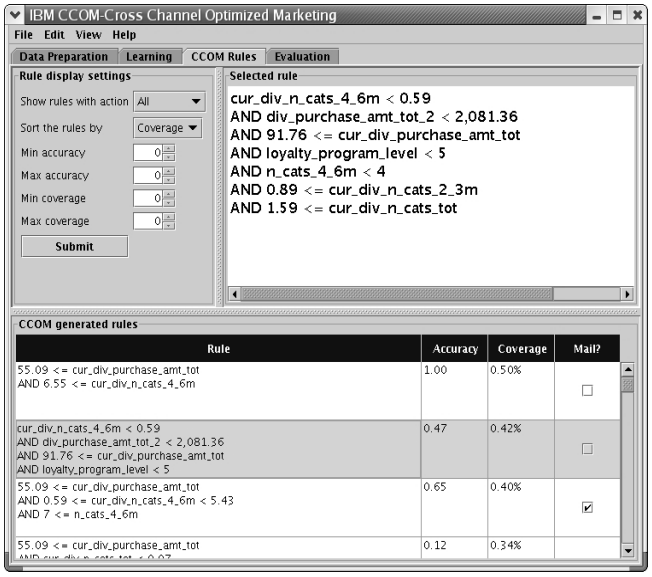

Figure 7: The CCOM GUI for displaying the obtained rules.

Pednault and Bianca Zadrozny of IBM Research and John Langford of TTI at Chicago for their discussions and assistance.

8. REFERENCES

[1] C. Apte, E. Bibelnieks, R. Natarajan, E. Pednault, F. Tipu, D. Campbell, and B. Nelson. Segmentation-based modeling for advanced targeted marketing. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 408–413. ACM, 2001.

[2] L. C. Baird. Reinforcement learning in continuous time: Advantage updating. In *Proceedings of the International Conference on Neural Networks*, June 1994.

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[4] S. Bradtke and M. Duff. Reinforcement learing methods for continuous-time Markov decision problems. In *Advances in Neural Information Processing Systems*, volume 7, pages 393–400. The MIT Press, nov 1995.

[5] P. Domingos. MetaCost: A general method for making classifiers cost sensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM Press, 1999.

[6] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 1996.

[7] S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning*, July 2002.

[8] R. Natarajan and E. Pednault. Segmented regression estimators for massive data sets. In *Second SIAM International Conference on Data Mining*, Arlington, Virginia, 2002. to appear.

[9] E. Pednault, N. Abe, B. Zadrozny, H. Wang, W. Fan, and C. Apte. Sequential cost-sensitive decision making with reinforcement learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2002. To appear.

[10] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[11] J. N. Tsitsiklis and B. V. Roy. An analysis of temporal difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

[12] X. Wang and T. Dietterich. Efficient value function approximation using regression trees. In *Proceedings of the IJCAI Workshop on Statistical Machine Learning for Large-Scale Optimization*, 1999.

[13] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, 1989.

[14] Q. Yang and H. Cheng. Mining plans for customer class transformation. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, Nov. 2003.

[15] B. Zadrozny. *Policy mining: Learning decision policies from fixed sets of data*. PhD thesis, University of California, San Diego, 2003.