Temporal Cross-Selling Optimization Using Action Proxy-Driven Reinforcement Learning

Nan Li
Computer Science Department
University of California, Santa Barbara
Santa Barbara, CA 93106, U.S.A.
Email: nanli@cs.ucsb.edu

Naoki Abe
Business Analytics & Mathematical Sciences Department
IBM T. J. Watson Research Center
Yorktwon Heights, NY 10598
Email: nabe@us.ibm.com

***Abstract*—Customer lifetime value modeling and cross-selling pattern mining are two important areas of data mining applications in marketing sciences. In this paper, we propose a novel approach that can address both of these problems in a unified manner. We propose a variant of reinforcement learning, enhanced with the notion of "action proxy", which is applicable to the cross-selling pattern discovery even in the absence of actions. For action proxies, we consider the target reward (changes) across product categories. The motivation is to optimize the target values of immediate rewards to maximize the expected overall long-term reward. Since the changes are directly tied to the reward, unconstrained formulation would result in unbounded behavior, leading us to constrain the learned policy. The goal is to optimize the target values while keeping their effects on the overall immediate rewards constrained. Experiments on real world data not only verify the effectiveness of our framework, but also provide qualitative study of allocation behavior, with particular emphasis on temporal cross-selling optimization.**

***Keywords*-Lifetime Value; Cross-Sell; Markov Decision Process; Reinforcement Learning; Business Optimization;**

I. Introduction

As contemporary economy becomes increasingly service oriented, corporate revenues become dependent on the establishment and maintenance of *long-term* customer relationships. It is a topic of interest in marketing sciences, often referred to as customer *lifetime value* (LTV) modeling. Customer LTV is defined as the present value of the long-term profits attributed to the customer. A recent approach in data mining to maximize LTV is based on *Reinforcement Learning* (RL) and *Markov Decision Process* (MDP) [1], [2]. However, RL-based approach requires historical marketing *actions* data, as a premise to learn the effects of the actions on the rewards. A marketing action is a process that allows an organization to concentrate its limited resources on the greatest opportunities to achieve a competitive advantage over time. Marketing actions take the form of various marketing activities involving customers, spanning across promotion advertisements, discount offers, campaigns, and so forth. In many business scenarios, historical marketing actions data may be infeasible to collect or extract. It is desirable to devise ways to still leverage the RL and MDP-based approach in the absence of actions.

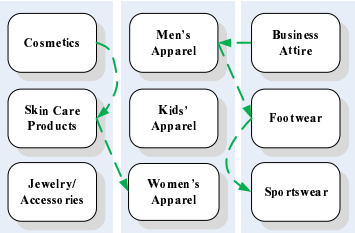


Figure 1. Cross-Selling Patterns Example

Previous works show the importance of another interesting problem: cross-selling mining [3], [4], [5]. Products that do not generate much profit by themselves can possibly be the catalyst for the sales of others. As in Fig. 1, cosmetics sales might lead to higher degree of customer loyalty, which subsequently leads to increased sales of skin care products and women's apparel. Another case in point is the movie recommendation system, where cultivating users' initial interest in film series (e.g. "Harry Potter") may spawn repeated rentals of the sequels. Association rule mining has been widely used to perform affinity analysis among products to discover cross-selling [3], [4].

While the above two topics are gaining attention, combining them together to study customer behavior is rarely considered. RL-based LTV maximization tends to overlook temporal cross-selling, because RL emphasizes on the optimal choice of actions. Typical formulation of the cross-selling problem usually involves only historical sales, not actions. Meanwhile, cross-selling mining tends to overlook long-term reward maximization, because the typical objective in cross-selling mining is the frequency of pattern occurrence.

In this paper, we propose a variant of reinforcement learning integrating LTV modeling and cross-selling mining within a unified framework. Our motivation is twofold: 1) utilize cross-selling effects among product categories to help enterprise maximize their overall customer lifetime value; 2) extend standard RL framework to address the challenge

imposed by the lack of actions data. We introduce the concept of *action proxy* to enhance RL to achieve both goals. We propose the framework of ***C***onstrained ***A***ction ***P***roxy-Driven ***R***einforcement ***L***earning (C-APRL), where observable features related to rewards are used as action proxies, to account for situations where historical actions data are not available. In order to discover temporal cross-selling, we further formulate action proxies using rewards across product categories.

Contributions: 1) We address both customer LTV modeling and temporal cross-selling mining in a unified manner. 2) We address the challenge of missing actions data via the concept of action proxies. 3) We present empirical evaluation on real business data from multiple domains.

II. Problem Formulation and Observation

This paper addresses an important business problem: how do we effectively model customer LTV, which can lead to a solution to optimizing cross-selling effects across product categories? To achieve this goal, we design a system to help marketers maximize the expected LTV of their customer base, or long-term customer satisfaction, while optimizing cross-selling effects among various product categories.

Challenges arise when historical marketing actions data are absent, and a standard RL-based approach is inapplicable. We propose to modify the RL framework with the notion of action proxy. More specifically, we design a family of action proxies, called *categorical action proxy* (CAP). CAPs are formulated with respect to target reward values for a given set of categories. In the business problem under consideration, CAPs are used to quantify cross-selling effects across product categories. CAPs are substitutes for missing marketing actions, and can be learned and optimized by our framework. For each customer, a learned policy is generated to determine a sequence of product categories designated for this customer. Such sequence represents an optimized temporal cross-selling pattern across categories, providing insights to decision makers.

III. Markov Decision Process and Lifetime Value Modeling

We briefly review MDP and RL-related concepts and the motivation of using them for customer LTV modeling.

A. Markov Decision Process

MDP is a Markov system with discounted rewards. Key components of a standard MDP include: **(1)** the state space, $S = \{s_1, s_2, \ldots, s_n\}$, and an initial state distribution $\phi : S \to \mathbb{R}$; **(2)** the action space $A = \{a_1, a_2, \ldots, a_m\}$, with a transition probability function $\tau : S \times A \times S \to [0, 1]$, such that $\forall s \in S$, $\forall a \in A$, $\Sigma_{s' \in S} \tau(s'|s, a) = 1$, where $\tau(s'|s, a)$ is the conditional probability of transiting to state s' from state s as a result of action a; and **(3)** the expected immediate reward function $R : S \times A \to \mathbb{R}$, where $R(s, a)$ is the expected immediate reward of taking action a at state s. For the ease of notation, $\tau(s, a)$ also denotes a random variable that satisfies $\Pr[\tau(s, a) = s'] = \tau(s'|s, a)$.

Given an MDP, a policy $\pi : S \to A$ determines the action to take in any state s. A policy π gives rise to an infinite sequence of quadruples, $\langle s_t, a_t, \tau(s_t, a_t), R(s_t, a_t)\rangle$, where s_t is the state at time t, a_t is the action in s_t, $\tau(s_t, a_t)$ is the next state and $r_t = R(s_t, a_t)$ is the expected immediate reward as a result of such transition. The initial state s_0 is drawn according to ϕ. Once an MDP is combined with a policy, the action for each state is stochastically determined.

The core task in an MDP is to find an optimal policy π^* that maximizes some cumulative function of the probabilistic rewards, typically the expected discounted sum over a potentially infinite horizon,

$$\pi^* = \text{argmax}_{\pi | a_t = \pi(s_t)} \Big(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \Big). \quad (1)$$

It is proven that for any MDP, there exists an optimal policy π^* that satisfies Bellman's fix-point equation:

$$V_{\pi^*}(s, a) = \max_{a' \in A} E[R(s, a) + \gamma V_{\pi^*}(\tau(s, a), a')]. \quad (2)$$

where $V_{\pi^*}(s, a)$ is the expected long-term cumulative reward at state s with a as the action, if policy π^* is followed at every step in the future. γ is a discount factor within $[0, 1]$.

A *constrained MDP* is an MDP where the policies under consideration must belong to a set of permissible policies, Π. In a standard constrained MDP [6], a policy is determined subject to a set of bounds on cumulative discounted "costs".

1) Reinforcement Learning: When the state transition probability function, $\tau(s, a)$ and the reward function, $R(s, a)$ are known, value iteration is a standard method to compute the optimal policy. It follows Bellman's equation, which expresses the value at a certain time point in terms of the payoff from some initial choices and the value of the remaining decision problem that results from those initial choices. Nonetheless, difficulties arise when the exact forms of $\tau(s, a)$ and $R(s, a)$ are unknown. This turns the MDP solving into a reinforcement learning problem. A variant of RL is Q-learning [7]. Q-learning uses the idea of *experience*-based updating. Value functions are updated according to the empirical quadruples, $\langle s_t, a_t, \tau(s_t, a_t), R(s_t, a_t)\rangle$ come across in the data. Let $Q_k(s, a)$ denote the Q-value function for state s if action a is taken during the k^{th} iteration.

$$\begin{aligned} Q_0(s_t, a_t) &= R(s_t, a_t), \\ Q_{k+1}(s_t, a_t) &= (1 - \alpha_k) Q_k(s_t, a_t) \\ &\quad + \alpha_k (R(s_t, a_t) + \gamma \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1})), \\ \pi^*(s_t) &= \text{argmax}_{a_t} Q_\infty(s_t, a_t), \end{aligned} \quad (3)$$

where α_k is the learning ratio at iteration k. The power of RL lies in its ability to solve the MDP without explicitly requiring the MDP model of the environment.

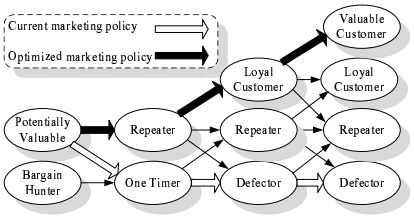


Figure 2. Customer State Transition Example

B. MDP in Customer LTV Modeling

Maximizing the expected customer LTV can be formulated as maximizing the discounted cumulative reward in an MDP. A state in MDP is extracted from a set of purchase features of a customer, such as demographics, purchase history, and so on. When using MDP, we focus on the purchase features. Customer states, such as "Repeater", "Loyal Customer" and "Valuable Customer" (Fig. 2), can further be extracted from their purchase features. Such states measure how valuable or profitable a customer is to the enterprise in the long run, and they can be translated to a set of conditions on the purchase features. An action in MDP corresponds to a marketing action, such as promotion campaigns, mailing catalogs, online advertisements, and so forth. A marketing action takes a customer from one state to another. In our problem formulation, we extract action proxies from historical rewards to approximate marketing actions. Computing LTV directly from data only reflects the value attained by the historical policy. MDP is able to attain the optimal LTV, through the optimized policy. Fig. 2 shows that MDP estimates the LTV along the optimal path (black arrow from "potentially valuable" to "valuable customer"), as opposed to the historical path (white arrow from "potentially valuable" to "defector").

IV. Methodology

We propose a MDP-based framework, ***C**onstrained **A**ction **P**roxy-driven **R**einforcement **L**earning* (C-APRL), to address LTV maximization in the absence of action data, with an emphasis on cross-sell discovery. The input of C-APRL is the historical transaction data without explicit marketing actions. The output of C-APRL is an optimized policy that gives rise to sequences of action proxies that maximize overall expected long-term rewards over all customers.

A. Action Proxies and Cross-Selling

In some business scenarios, marketing actions, such as mailings and campaigns, are difficult to acquire. Such absence obstructs the statistical learning of the relationship between customer LTV and marketing actions. In this paper, we propose action proxies, a set of observable features as substitutes for the missing actions. In order to discover temporal cross-selling relations across product categories, we employ the notion of categorical action proxies (CAPs). One instantiation of CAPs is the changes of purchase amounts in various product categories. Such changes reflect the change of attitude of a customer towards a specific product type. A positive change shows an increase of interest from the customer, and the enterprise might want to invest more in promoting related products; whereas a negative change probably implies a decrease of interest. The learned policy contains rich information on how various categories interact over time, providing valuable cross-selling insight for marketing decision makers.

A policy, in the context of CAPs, is interpreted as: $\pi : \boldsymbol{S} \rightarrow \boldsymbol{A} = \{\vec{\delta}\}$, where $\vec{\delta} = (\delta(1), \ldots, \delta(C))^T$ is the purchase amount change vector of length C with C as the number of categories. Let $\boldsymbol{s}_t$ denote the customer state vector at time t (composed of a set of features). $\pi^*(\boldsymbol{s}_t) = \boldsymbol{a}_t = (\delta_t(1), \ldots, \delta_t(C))^T$, where $\delta_t(c), c \in \{1, \ldots, C\}$ is the purchase amount change for category c from time interval $(t-1, t)$ to $(t, t+1)$.

B. Constrained Reinforcement Learning

The CAP formulation is continuous and related to the immediate rewards, thus straightforward learning would generate a policy that allocates infinite values to the action proxies (unbounded behavior). It is critical to impose constraints on the learned policy. The constrained MDP enforces the policy under consideration to belong to a set of permissible policies, which is determined with respect to a set of constraints. Our goal of constraints is twofold: 1) the assigned action proxies are comparable to what was observed empirically; 2) the assignment is subject to various business and economical limits, such as financial and man-hour budgets.

A family of constrained batch-mode RL methods have been proposed in [2]. Amongst them is the *constrained advantage updating* algorithm, a constrained version of the algorithm by Baird [8], which extends standard Q-learning to variable time intervals and continuous time-stamp scenarios. Advantage updating learns the relative advantage of competing actions in any given state, thus avoiding explicit estimate of the Q-value function. The underlying principle of advantage updating focuses on the notion of "advantage" of action $\boldsymbol{a}_t$, in comparison to the optimal action $\boldsymbol{a}_t^*$. Let $A(\boldsymbol{s}_t, \boldsymbol{a}_t)$ denote such advantage, we can derive:

$$A(\boldsymbol{s}_t, \boldsymbol{a}_t) = \frac{1}{\Delta t}(R(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma^{\Delta t} V^*(\boldsymbol{s}_{t+1}) - V^*(\boldsymbol{s}_t)), \quad (4)$$

where $V^*(\boldsymbol{s}_t) = \max_{\boldsymbol{a}_t'} Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t')$ and $Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t')$ is the Q-value of an optimal policy. Δt denotes the time interval from $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$. From Eq. (4) we can further derive:

$$A(\boldsymbol{s}_t, \boldsymbol{a}_t) = \frac{1}{\Delta t}(Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t) - \max_{\boldsymbol{a}_t'} Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t')). \quad (5)$$

Advantage updating is different from Q-learning in two aspects: 1) it normalizes the Q-value with respect to time interval; 2) it normalizes the Q-value with respect to state dependency and focuses on relative advantage of actions. For

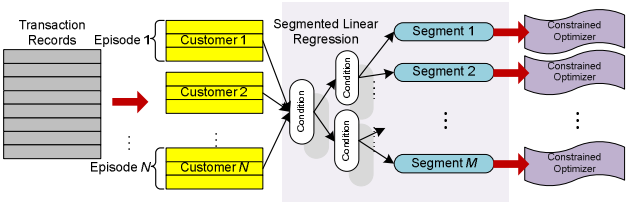


Figure 3. Overall Implementation of C-APRL Framework

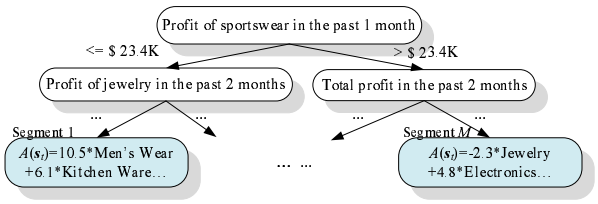


Figure 4. Segmented Linear Regression Example

marketing data where time intervals are naturally variable, this is a desirable property. Previous work [2] has observed that for the problem under study, advantage updating outperforms Q-learning empirically (also confirmed by our own experience). Therefore we adopt the batch advantage updating as the reinforcement learning method in this work.

C. Framework Implementation

To account for a state space with a lot of features, *function approximation* [1], [2] is used in the batch learning. An implementation coupling segmented regression and constrained optimization is adopted. Fig. 3 shows the overall framework. The input data are a set of time-stamped transaction records. Each record is a snap shot of the state of a customer at a time point. The data are grouped into a series of episodes. Each episode corresponds to one customer, and contains all the transaction records of this customer. In each iteration, a segmented linear regression model is applied on all the episodes, the output of which consists of a set of segments, each with a linear regression model. Constrained optimization is then applied on all the segments.

Segmented Linear Regression. At iteration k of advantage updating, we perform segmented linear regression over all the data episodes. Data records are first segmented based on their features, with a linear regression model in each segment. Each segment is defined by a conjunctive condition on the features and a regression equation. In each segment, the linear regression is conducted on the advantage value, A_k, which is estimated in terms of action proxies. Fig. 4 exemplifies this with a simple case, where a tree structure is used to segment the records. The root groups the records according to the profit of sportswear, the results of which will be further classified using criteria on other features. The leaf nodes are the final segments, each associated with a regression model, with the advantage value as the regressand and action proxies as the regressors.

$$A_k(\boldsymbol{s}_t) = A_k^0(\boldsymbol{s}_t) + \vec{\theta}_{\mathbb{S}} \cdot \boldsymbol{a}_t, \boldsymbol{s}_t \in \mathbb{S}. \tag{6}$$

Eq. (6) is the linear regression for segment $\mathbb{S}$, where $\boldsymbol{s}_t$ belongs to $\mathbb{S}$ and $A_k^0(\boldsymbol{s}_t)$ is a constant. Eq. (6) yields a set of coefficients for the action proxies, denoted by vector $\vec{\theta}_{\mathbb{S}}$.

Constrained Linear Programming Optimization. At iteration k of advantage updating, once a linear relation between the advantage and the action proxies is established, we use constrained linear programming optimization to choose the optimal action proxies for each segment. Our goal is to maximize the aggregated advantage over the entire population. According to Eq. (6), we need to maximize

$$\max\{\sum_{\mathbb{S}} |\mathbb{S}|\vec{\theta}_{\mathbb{S}} \cdot \boldsymbol{a}_t\}, \tag{7}$$

where $|\mathbb{S}|$ is the cardinality of segment $\mathbb{S}$. Two types of constraints are applied on the optimization: **1)** It is imperative to ensure the action proxies produced by the learned policy adhere to what was observed empirically in the past. We propose the usage of empirical bounds, expressed as

$$\Pi = \left\{\pi \middle| \boldsymbol{B}^L \leq E_\phi[\mathcal{C}^{\boldsymbol{s}_t} \times \boldsymbol{a}_t] \leq \boldsymbol{B}^U\right\}, \tag{8}$$

where $\mathcal{C}^{\boldsymbol{s}_t}$ is a $C \times C$ diagonal matrix with $\mathcal{C}^{\boldsymbol{s}_t}(c,c)$ as the unit cost incurred by action proxy $c, c \in \{1,\ldots,C\}$ in state $\boldsymbol{s}_t$. $\boldsymbol{B}^L(c)$ and $\boldsymbol{B}^U(c)$ are the lower and upper for action proxy c. For all segments, we constrain the policy as:

$$\boldsymbol{B}_S^L(c) \leq \frac{\sum_{\mathbb{S}} \tilde{\boldsymbol{a}}_{\mathbb{S}}(c)|\mathbb{S}|}{\sum_{\mathbb{S}} |\mathbb{S}|} \leq \boldsymbol{B}_S^U(c), c = \{1,\ldots,C\}, \tag{9}$$

where $\tilde{\boldsymbol{a}}_{\mathbb{S}}$ is the assigned action vector of the optimization for all C action proxies in segment $\mathbb{S}$. The assignment is passed on to all records in $\mathbb{S}$. $\boldsymbol{B}_S^L(c)$ and $\boldsymbol{B}_S^U(c)$ are the empirical lower and upper bounds for action proxy c over the state space S, constraining the range of allocation over the population. One way to compute such bounds is to use empirical mean minus and plus the empirical standard deviation, as in Eq. (10):

$$\mu_S(c) - \sigma_S(c) \leq \frac{\sum_{\mathbb{S}} \tilde{\boldsymbol{a}}_{\mathbb{S}}(c)|\mathbb{S}|}{\sum_{\mathbb{S}} |\mathbb{S}|} \leq \mu_S(c) + \sigma_S(c). \tag{10}$$

2) In reality, promotion is conducted subject to limited resources and budgets, which imposes a second type of constraints. We especially consider the loyalty group constraints. Each customer belongs to a loyalty group, which reflects how "loyal" this customer is to the enterprise. Each segment is associated with a loyalty group, i.e. all customers in this segment belong to the same group. Each loyalty group g has a group budget $B(g)$, which limits the the amount of actions assigned to this group.

$$0 \leq \sum_{\mathbb{S}|G(\mathbb{S})=g} \tilde{\boldsymbol{a}}_{\mathbb{S}} \cdot \mathcal{C}_{\mathbb{S}} = \sum_{\mathbb{S}|G(\mathbb{S})=g} \sum_c \tilde{\boldsymbol{a}}_{\mathbb{S}}(c)\mathcal{C}_{\mathbb{S}}(c) \leq B(g), \tag{11}$$

Data: Historical data $D = \{\langle \boldsymbol{s}_{i,j}, \boldsymbol{a}_{i,j}, R_{i,j}, t_{i,j}\rangle\}, i = 1, \ldots, N, j = 1, \ldots, l_i$ grouped into N episodes (l_i in length each), iteration number K
Result: Optimal policy π^*_K

Initialize $D_0 = \{\langle(\boldsymbol{s}_{i,j}, \boldsymbol{a}_{i,j}), \frac{R_{i,j}}{\Delta t_{i,j}}\rangle\}$ for all states;
Initialize $A_0 = \{\frac{R_{i,j}}{\Delta t_{i,j}}\}$ for all states;
for $k = 1$ *to* K **do**
  Segmented Regression: apply segmented regression upon D_k to obtain a set of segments $\{\mathbb{S}_1, \ldots, \mathbb{S}_M\}$, where $\mathbb{S}_i$ corresponds to a regression model expressing the relations between $A_k(i, j)$ and feature vector $(\boldsymbol{s}_{i,j}, \boldsymbol{a}_{i,j})$;
  Constrained Optimization: for each segment, assign the best action proxies via constrained optimization to maximize Eq. (7) s.t. Eq. (9) and Eq. (11); use those assigned action proxies to update policy π^*_k;
  Advantage update: update A_k using π^*_k;
return π^*_K;

Algorithm 1: C-APRL Framework

where the loyalty group of segment $\mathbb{S}$ is g and $\mathcal{C}_{\mathbb{S}}(c)$ is the unit cost of taking action c for segment $\mathbb{S}$. Eq. (11) states that the cost of promotion over all segments associated with a certain group should not exceed a budget specific to this group. Alg. 1 describes the overall C-APRL framework.

V. Lift Advantage Evaluation

A challenge for any data-driven business optimization methodology is to effectively evaluate the performance. Such evaluation is typically required to be done without real deployment and using only historical data. It is difficult since historical data only behave according to the observed policy. Importance sampling-based *bias correction* [9] is a viable method to overcome such difficulty, and has been used in past works [1], [2]. We introduce bias correction-based *lift advantage* evaluation, an effective way to measure the expected cumulative reward of a learned policy when sampling the data with respect to the observed policy. Let $\tilde{\pi}$ be the learned policy by the C-APRL framework, while let π_0 be the observed policy. Let $\widehat{R}_{\pi_0}(\tilde{\pi})$ be the expected cumulative reward of $\tilde{\pi}$ over all possible states when sampled with respect to π_0 and state distribution ϕ. It is formulated as:

$$\begin{aligned}\widehat{R}_{\pi_0}(\tilde{\pi}) &= E\Big[\frac{\Pr_{\tilde{\pi}}[\boldsymbol{a}_t|\boldsymbol{s}_t]}{\Pr_{\pi_0}[\boldsymbol{a}_t|\boldsymbol{s}_t]}\widehat{R}_{\pi_0}(\boldsymbol{s}_t, \boldsymbol{a}_t)\Big] \\ &= E\Big[\frac{\Pr_{\tilde{\pi}}[\boldsymbol{a}_t|\boldsymbol{s}_t]}{\Pr_{\pi_0}[\boldsymbol{a}_t|\boldsymbol{s}_t]}\big(\Sigma_{i=0}^{i=f}\gamma^i R(\boldsymbol{s}_{t+i}, \boldsymbol{a}_{t+i})\big)\Big],\end{aligned} \tag{12}$$

which measures the expected cumulative reward achieved by $\tilde{\pi}$, estimated using π_0 as the sampling policy. $\widehat{R}_{\pi_0}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is the observed discounted cumulative reward, with f as the number of time stamps looked ahead. As shown, lift evaluation employs bias correction, in which $\widehat{R}_{\pi_0}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is multiplied by the ratio between the probabilities of the action proxy vectors by the respective policies. $\Pr_{\pi_0}[\boldsymbol{a}_t|\boldsymbol{s}_t]$ is the conditional probability of observing $\boldsymbol{a}_t$ at $\boldsymbol{s}_t$ according to π_0, and $\Pr_{\tilde{\pi}}[\boldsymbol{a}_t|\boldsymbol{s}_t]$ is that according to $\tilde{\pi}$. Assuming independence among action proxies we have:

$$\frac{\Pr_{\tilde{\pi}}[\boldsymbol{a}_t|\boldsymbol{s}_t]}{\Pr_{\pi_0}[\boldsymbol{a}_t|\boldsymbol{s}_t]} = \prod_c \frac{\Pr_{\tilde{\pi}}[\boldsymbol{a}_t(c)|\boldsymbol{s}_t]}{\Pr_{\pi_0}[\boldsymbol{a}_t(c)|\boldsymbol{s}_t]}, c = \{1, \ldots, C\}, \tag{13}$$

which is the *bias ratio* in the bias correction. Due to its wide usage in finance and marketing [10], where large monetary values exist pervasively, we propose to adopt *log-normal distribution* ($\ln\mathcal{N}$) to model action proxy allocations. We consider the action proxy distribution within each regression segment. Each record is associated with an observed action proxy vector $\boldsymbol{a}_t$ (from π_0) and an assigned vector $\tilde{\boldsymbol{a}}_t$ (from $\tilde{\pi}$). Log-normal distribution is assumed for $\boldsymbol{a}_t(c)$ and $\tilde{\boldsymbol{a}}_t(c)$ within any segment $\mathbb{S}$. $\Pr[\boldsymbol{a}_t|\boldsymbol{s}_t] = \Pr[\boldsymbol{a}_t|\mathbb{S}, \boldsymbol{s}_t \in \mathbb{S}]$ holds for both $\tilde{\pi}$ and π_0. Based on the $\ln\mathcal{N}$ PDF

$$P(x) = \frac{1}{x\sqrt{2\pi\sigma^2}}\exp\{-\frac{(\ln x - \mu)^2}{2\sigma^2}\},$$

where μ and σ are the *mean* and *standard deviation* of the variable's natural logarithm. If we assume $\tilde{\pi} \sim \ln\mathcal{N}(\tilde{\mu}, \tilde{\sigma}^2)$ and $\pi_0 \sim \ln\mathcal{N}(\mu_0, \sigma_0^2)$, we can derive

$$\begin{aligned}\frac{\Pr_{\tilde{\pi}}[\boldsymbol{a}_t(c)|\boldsymbol{s}_t]}{\Pr_{\pi_0}[\boldsymbol{a}_t(c)|\boldsymbol{s}_t]} &= \frac{\frac{1}{\boldsymbol{a}_t(c)\sqrt{2\pi\tilde{\sigma}^2}}\exp\{-\frac{(\ln \boldsymbol{a}_t(c) - \tilde{\mu})^2}{2\tilde{\sigma}^2}\}}{\frac{1}{\boldsymbol{a}_t(c)\sqrt{2\pi\sigma_0^2}}\exp\{-\frac{(\ln \boldsymbol{a}_t(c) - \mu_0)^2}{2\sigma_0^2}\}} \\ &= \frac{\sigma_0}{\tilde{\sigma}}\exp\{\frac{(\ln \boldsymbol{a}_t(c) - \mu_0)^2}{2\sigma_0^2} - \frac{(\ln \boldsymbol{a}_t(c) - \tilde{\mu})^2}{2\tilde{\sigma}^2}\},\end{aligned}$$

where both policies' mean and standard deviation can be estimated from the observed action proxy vectors embedded in the data, $\{\boldsymbol{a}_t\}$, as well as the allocated action proxy vectors, $\{\tilde{\boldsymbol{a}}_t\}$. Therefore by using log-normal distribution, we are able to estimate the expected lift advantage of the learned policy according to Eq. (12).

VI. Experiments

Experiments are done on real-world data of multiple domains, ranging from retail to recommendation systems. C-APRL is shown to produce policies that can enhance the long-term rewards. Coin-OR linear programming (Clp)[1] toolkit is used as the optimization solver.

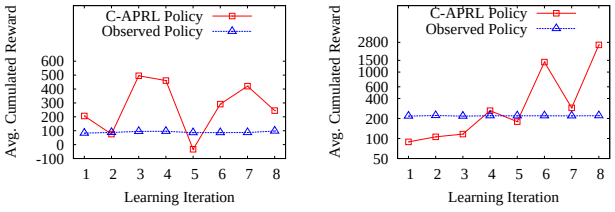
A. Data Sets

Two real-world data sets are used. Each is processed into a series of time-stamped records, half for learning and half for testing, i.e. lift evaluation. Each record contains a set of features, including the action proxy fields and the immediate

[1] `https://projects.coin-or.org/Clp`

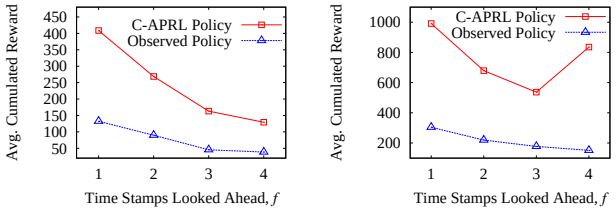
Table I
CATEGORICAL ACTION PROXIES DESCRIPTION ON *Saks* AND *MovieLens*

Saks				*MovieLens*			
ID	**Description**	**LB** ($\$K$)	**UB** ($\$K$)	**ID**	**Description**	**LB**	**UB**
1	Women's clothing, bridal apparel, etc.	-0.07	0.18	1	Action, crime, war, western, etc.	-13.40	60.98
2	Jeans, denim, sportswear, etc.	-0.04	0.14	2	Adventure, fantasy, thriller, etc.	-10.17	46.84
3	Loungewear, kids' clothing, etc.	-0.05	0.14	3	Animation, children's, etc.	-2.38	11.35
4	Men's furnishing & accessories, etc.	-0.04	0.09	4	Comedy, musical, romance, etc.	-12.07	54.91
5	Specialty shops, etc.	-0.04	0.06	5	Documentary, etc.	-0.21	0.90
6	Footwear, etc.	-0.17	0.35	6	Drama, etc.	-8.79	38.27
7	Catalog goods, etc.	-0.01	0.01				
8	Fragrance, body treatments, etc.	-0.02	0.04				
9	Accessories, jewelry, etc.	-0.14	0.19				



(a) *Saks* (b) *MovieLens*

Figure 5. Iterative Lift Evaluation on C-APRL and Observed Policies



(a) *Saks* (b) *MovieLens*

Figure 6. Lift Advantage vs. f on C-APRL and Observed Policies

reward field. Table I shows the semantics of the categorical action proxies (CAPs), their empirical lower bound ("LB") and upper bound ("UB") (Eq. (10)).

Saks Fifth Avenue Data. The *Saks* data are the sales data provided by Saks Fifth Avenue. A sample of 5,000 customers is used, and a sequence of 68 states is generated for each of them, corresponding to 68 marketing campaigns in 2002, amounting to 340,000 data records. Each record represents a time-stamped historical transaction. CAPs are the purchase changes of 9 categories (Table I), where "catalog goods" are those purchased from the catalogs.

MovieLens Data. The *MovieLens* data collected from MovieLens[2] contain 1,000,209 anonymous ratings (each is an integer in $[1, 5]$) of approximately 3,900 movies made by 6,040 MovieLens users. We segment the data into 15 time windows for each user. CAPs are the ratings of 6 movie categories (Table I) in the current time window and the immediate reward is the total rating in the next, since we consider the customer's overall satisfaction as an indicator of this customer's potential value to the rental company.

B. Lift Evaluation and Comparison

In order to establish comparison, we compare C-APRL to the observed policy and a baseline policy. The observed policy is that embedded in the data without optimization. A non-iterative baseline does not apply reinforcement learning ("Non-RL Baseline") and finishes after one iteration.

[2]http://www.movielens.org/

1) Learned vs. Observed: Lift evaluation illustrates the advantage of C-APRL policy compared to the observed policy, using the estimated average cumulated reward (EAC-reward) over a specified number (f) of time stamps into the future. Fig. 5 plots how the EAC-rewards of the two policies change with the iteration for $f = 2$. For C-APRL, a typical run starts with a policy that is relatively uninformed, which does not show apparent advantage over the observed. In later iterations, C-APRL framework achieves significant advantage via iterative re-modeling and re-optimization. The learning curves of the C-APRL policy exhibit local optima, and tend to fluctuate through the iterations. It shows that C-APRL is able to learn from previous errors and improve the performance in response; once the performance starts to deteriorate due to accumulated errors, this cycle repeats, which leads to the presence of local optima. It indicates the trade-off between the benefits of iterative improvement, and the errors cumulated due to many applications of regression. The results are nonetheless encouraging. C-APRL can increase the long-term rewards for both data sets with an appropriate stopping rule.

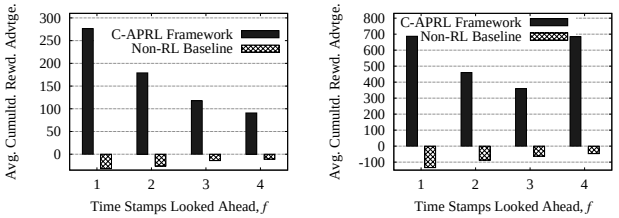
Fig. 6 shows the comparison from another perspective, i.e. how the EAC-rewards change over f. In most cases, the EAC-rewards decrease as f increases. This is because the EAC-reward for each record, is divided by a factor that measures the overall cumulated discount from time t to $t+f$. A larger value of f results in a larger denominator. The apparent lift advantage over the observed policy validates the effectiveness of C-APRL.

Table II
MOST FREQUENT ACTION SEQUENCES ON *Saks* AND *MovieLens*

Saks				***MovieLens***			
C-APRL, $f = 2$		Observed		C-APRL, $f = 2$		Observed	
k	**Action Sequence**	k	**Action Sequence**	k	**Action Sequence**	k	**Action Sequence**
1	catalog→jewelry→women's	1	men's furnish→footwear→women's	1	war→crime→drama	1	comedy→romance→comedy
2	specialty→catalog→jeans	2	specialty→jewelry→footwear	2	crime→action→romance	2	drama→drama→drama
3	jeans→jewelry→jeans	3	footwear→catalog→footwear	3	western→drama→adventure	3	action→crime→adventure
4	specialty→catalog→catalog	4	specialty→men's furnish→specialty	4	action→crime→adventure	4	action→comedy→romance
5	kids'→catalog→jewelry	5	jeans→catalog→catalog	5	adventure→western→crime	5	action→crime→romance

Table III
EXAMPLE MODELING SEGMENTS ON *Saks* AND *MovieLens*

Saks		***MovieLens***	
Segment Definition	**Action Allocation**	**Segment Definition**	**Action Allocation**
If {profit_total $<$ 86.56, profit_jewelry $<$ 317.50, profit_men_furnish $\geq$ 3.00, etc.} **Then** {reward - 146.45 = 4.66 $*$ jeans + 9.99 $*$ footwear}	footwear: 12.96 fragrance: 75.83	**If** {2 $\leq$ rating_total $<$ 176 rating_fantasy $\geq$ 1, rating_drama $<$ 8, etc.} **Then** {reward + 0.1 = 0.08 $*$ action + 0.05 $*$ comedy}	comedy: 58.39



(a) *Saks* (b) *MovieLens*

Figure 7. Lift Advantage vs. f on C-APRL and Baseline Policies

2) Iterative Learning vs. Non-iterative Baseline: To verify that the advantage by C-APRL is indeed caused by iterative modeling, a non-RL baseline method is used for comparison. The baseline completes only one iteration of C-APRL without iterative improvement, but the advantages are initialized as empirical LTVs (cumulative discounted rewards from the current time till the end), rather than just immediate rewards. The effect of iterative improvement is shown in Fig. 7. It is evident that the lift advantage by C-APRL over the observed policy, is much larger than that by the baseline. Apparently direct modeling without iterative learning does not lead to any policy advantage.

C. Sequential Pattern Recognition

A policy gives rise to a sequence of actions. We study the semantics of the policies via examining the most frequent action sequences. An action sequence is built from a set of $f + 1$ records, each associated with a representative action proxy (RAP). Each action proxy is chosen to be the representative with a probability proportional to its observed value in that record, i.e. $\Pr[\text{RAP}(\boldsymbol{s}_t) = c] \sim \boldsymbol{a}_t(c)$. All $f + 1$ RAPs form an action sequence. The k most frequent action sequences are estimated after a number of records are sampled from the test data. Each sampled record and the f number of records following it are used to build a sequence. For the observed policy, records are sampled randomly; for the C-APRL policy, records are sampled stochastically by their bias ratios. Table II shows the top-5 most frequent action sequences. Interesting temporal cross-sell effects are observed. For example, C-APRL on *Saks* favors sequences such as: catalog goods, followed by jewelry/accessories and followed by women's clothing (No. 1); while on *MovieLens* it favors sequences such as: western, followed by drama and then by adventure (No. 3).

D. Case Study on Modeling Segments

We also inspect concrete examples of regression segments. For example, the left column of Table III shows the 16^{th} segment taken from the 3^{rd} iteration of C-APRL on *Saks* ($f = 2, \lambda = 0.6$). It contains 8,573 records, around 5% of the entire test population. The segment definition consists of constraints on the features and the assigned coefficients on action proxies. The definition can be interpreted as: "if within the short past, the total profit is less than $\$86.56K$, the profit of accessories/jewelry is below $\$317.5K$, and that of men's furnishing/accessories is above $\$3K$, then jeans/sportswear and footwear would potentially have high impacts on the long-term customer rewards". A cross-sell effect can thus be indirectly identified. The "Action Allocation" column denotes the assigned values in this segment by constrained optimization. As indicated, the optimized allocation favors footwear and fragrance/body treatment, which further attributes the lift advantage to enhanced consideration of temporal cross-selling.

VII. Related Works

Two directions have emerged in modeling customer behaviors, customer lifetime value modeling [2], [1], [11], [12], [13], [14] and cross-sell mining [4], [15]. Typical examples in the first direction include RFM model [13], probability model [14], and MDP model [2], [1]. The second direction focuses on discovering cross-sell effects. An association rule-based objective function is proposed in [4] to measure the cross-sell utility of products and is further used to select products for promotion. Our work is motivated to combine those two into a unified framework.

Our framework couples MDP with RL to address a business optimization problem. MDP has been studied as the framework for autonomous learning in both humans and robotics [16]. However, applying MDP to business optimization has been under-explored. Meanwhile, various RL techniques were proposed [7], [17], [18]. A variant of RL is Q-learning [7]. Q-learning uses the idea of experience-based updating. It can be done either online or in batch [17], [18]. Batch RL reduces the amount of data needed by leveraging available computation and memory [18]. In this work, batch advantage updating [2] is adopted instead, given its advantage in business optimization.

The problem addressed in this work is integrating data modeling into the process of decision making, which has been studied extensively. Notable examples include cost sensitive learning [19] and economic learning [20]. The cost structure used in those works, is however simple and straightforward. In real world, complex constraints exist to restrict permissible decisions. It is essential to extend cost minimization to constrained cost optimization, which is one of the goals of this paper.

VIII. Conclusions

In this paper, we address a general problem in customer LTV modeling, and propose a solution using a novel variant of RL supported by "action proxy". We present a practical application of this general idea, namely, the problem of temporal cross-sell optimization. Experiments on real world data show that the enhanced performance is due to both the iterative re-optimization and the consideration of temporal cross-selling effects. Our future works include: **1)** explore other alternatives of formulating action proxies; **2)** incorporate more types of business constraints; **3)** make the system less parameter-laden for future deployment.

References

[1] N. Abe, P. Melville, C. Pendus, C. K. Reddy, D. L. Jensen, V. P. Thomas, J. J. Bennett, G. F. Anderson, B. R. Cooley, M. Kowalczyk, M. Domick, and T. Gardinier, "Optimizing debt collections using constrained reinforcement learning," in *KDD*, 2010, pp. 75–84.

[2] N. Abe, N. K. Verma, C. Apté, and R. Schroko, "Cross channel optimized marketing by reinforcement learning," in *KDD*, 2004, pp. 767–772.

[3] R. C.-W. Wong, A. W.-C. Fu, and K. Wang, "MPIS: maximal-profit item selection with cross-selling considerations," in *ICDM*, 2003, pp. 371–378.

[4] N. Li, Y. Yang, and X. Yan, "Cross-selling optimization for customized promotion," in *SDM*, 2010, pp. 918–929.

[5] M. Armony and I. Gurvich, "When promotions meet operations: cross-selling and its effect on call center performance," *Manufacturing & Service Operations Management*, vol. 12, no. 3, pp. 470–488, 2010.

[6] E. Altman, "Asymptotic properties of constrained markov decision processes," 1991.

[7] C. J. C. H. Watkins and P. Dayan, "Technical note Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[8] L. C. Baird, "Reinforcement learning in continuous time: advantage updating," in *ICNN*, 1994, pp. 2448–2453.

[9] S. Kakade and J. Langford, "Approximately optimal approximate reinforcement learning," in *ICML*, 2002, pp. 267–274.

[10] D. Plikynas and R. Aleksiejunas, "Neural network based multiagent system for simulation of investing strategies," in *CIA*, 2007, pp. 164–180.

[11] S. Gupta, D. Hanssens, B. Hardie, W. Kahn, V. Kumar, N. Lin, N. Ravishanker, and S. Sriram, "Modeling customer lifetime value," *Journal of Service Research*, vol. 9, no. 2, pp. 139–155, November 2006.

[12] S. Gupta and D. Lehmann, "Customers as assets," *Journal of Interactive Marketing*, vol. 17, no. 1, pp. 9–24, 2003.

[13] Y.-L. Chen, M.-H. Kuo, S. yi Wu, and K. Tang, "Discovering recency, frequency, and monetary (RFM) sequential patterns from customers' purchasing data," *Electronic Commerce Research and Applications*, vol. 8, no. 5, pp. 241–251, 2009.

[14] D. Schmittlein, D. Morrison, and R. Colombo, "Counting your customers: who they are and what will they do next?" *Management Science*, vol. 33, pp. 1–24, January 1987.

[15] X. Zhang, W. Gong, and Y. Kawamura, "Customer behavior pattern discovering with web mining," in *APWeb*, 2004, pp. 844–853.

[16] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[17] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, pp. 503–556, 2005.

[18] S. Kalyanakrishnan and P. Stone, "Batch reinforcement learning in a complex domain," in *AAMAS*, 2007, p. 94.

[19] C. Elkan, "The foundations of cost-sensitive learning," in *IJCAI*, 2001, pp. 973–978.

[20] F. J. Provost, P. Melville, and M. Saar-Tsechansky, "Data acquisition and cost-effective predictive modeling: targeting offers for electronic commerce," in *ICEC*, 2007, pp. 389–398.