Dynamic Marketing Policies: Constructing Markov States for Reinforcement Learning

By

Yutting Zhu

B.A. in Economics, Renmin University of China (2015)
B.S. in Mathematics, Renmin University of China (2015)
M.A. in Economics, University of Rochester (2017)

SUBMITTED TO THE SLOAN SCHOOL OF MANAGEMENT IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MANAGEMENT RESEARCH

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MAY 2020

Signature of Author:____

Department of Management
MAY 04, 2020

Certified by: ____

Juanjuan Zhang
John D.C. Little Professor of Marketing
Thesis Supervisor

Accepted by: ____

Catherine Tucker
Sloan Distinguished Professor of Management
Professor, Marketing
Faculty Chair, MIT Sloan PhD Program

Dynamic Marketing Policies: Constructing Markov States for Reinforcement Learning

by

Yutting Zhu

Submitted to the THE SLOAN SCHOOL OF MANAGEMENT
on MAY 04, 2020, in partial fulfillment of the
requirements for the degree of
MASTER OF SCIENCE IN MANAGEMENT RESEARCH

Abstract

Many firms want to target their customers with a sequence of marketing actions, rather than just a single action. we interpret sequential targeting problems as a Markov Decision Process (MDP), which can be solved using a range of Reinforcement Learning (RL) algorithms. MDPs require the construction of Markov state spaces. These state spaces summarize the current information about each customer in each time period, so that movements over time between Markov states describe customers' dynamic paths. The Markov property requires that the states are "memoryless," so that future outcomes depend only upon the current state, not upon earlier states. Even small breaches of this property can dramatically undermine the performance of RL algorithms. Yet most methods for designing states, such as grouping customers by the recency, frequency and monetary value of past transactions (RFM), are not guaranteed to yield Markov states.

We propose a method for constructing Markov states from historical transaction data by adapting a method that has been proposed in the computer science literature. Rather than designing states in transaction space, we construct predictions over how customers will respond to a firm's marketing actions. We then design states using these predictions, grouping customers together if their predicted behavior is similar. To make this approach computationally tractable, we adapt the method to exploit a common feature of transaction data (sparsity). As a result, a problem that faces computational challenges in many settings, becomes more feasible in a marketing setting. The method is straightforward to implement, and the resulting states can be used in standard RL algorithms. We evaluate the method using a novel validation approach. The findings confirm that the constructed states satisfy the Markov property, and are robust to the introduction of non-Markov distortions in the data.[1]

Thesis Supervisor: Juanjuan Zhang
Title: John D.C. Little Professor of Marketing

[1]This essay is based on joint work with Duncan Simester, Jonathan Parker and Antoinette Schoar

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Standard targeting models are myopic. They treat marketing actions as single period decisions with single period outcomes. However, in practice, firms can often implement a sequence of marketing actions in order to affect current and future outcomes. For example, instead of just deciding which customers to call (and which not to call), it may be optimal to call, then email, then mail to one customer, while for another customer it is optimal to call twice and then email.

A sequential targeting problem can be interpreted as a Markov Decision Process (MDP), which can be solved using a range of Reinforcement Learning methods. We will formally define MDPs in Section 3, but we can illustrate the concept using a baseball example. The objective of the batting team is to score as many runs as possible in each inning. At each point in time, the situation can be summarized using a finite set of possible states. For example, suppose there is a runner on first base, the other bases are empty, and there is one out in the inning. We will label this as State 1. A key feature of an MDP is that there is a decision to be made. In our example, we will focus on the runner's decision to attempt to steal second base (for simplicity we will ignore the actions and outcomes for the batter). The runner has two action options (his "action space"): he can attempt to steal or not. If he steals and succeeds, the new state of the game will be: a runner on second base, other bases are empty, and there is one out in the inning (we will denote this as State 2). In contrast, if the runner steals and fails, the state of the game will change to: all of the bases are empty and there are two outs (denote this as State 3). Thus, the choice of action by the runner affects the transitions to future states.

The transitions to future states in turn affect the expected number of runs the team will score in that inning (the "expected rewards"). A simple empirical average from past Major League Baseball games reveals that on average a team scores 0.5088 runs in State 1, 0.6675 runs in State 2 and 0.0986 runs in State 3. Using these expected rewards, we can calculate the values associated with each action in State 1 (the "state-action pairs"): if the runner does not attempt to steal we remain in State 1 and expect to score 0.5088 runs. If the runner steals, the expected rewards from the State1-Steal (state-action) pair is: $0.6675 * \beta + 0.0986 * (1 - \beta)$, where β is the probability of success. By comparing the values of the two state-actions pairs (State 1-Steal and State 1-Not Steal), the runner can choose the optimal action in State 1. In this case, the runner should steal as long as the probability of success exceeds 0.7201. This probability might also be estimated by observing past outcomes (ideally past outcomes for that runner and that pitcher).

Notice that the value associated with attempting a steal incorporates the long-term impact of the current action. This also allows the choice of the current action to incorporate the impact of future actions. For example, if the runner is particularly good at stealing bases, and so is also likely to successfully steal from second base to third base, this will increase the expected number of runs the team will score in State 2. The expected rewards for State1-Steal should (and would) incorporate this adjustment. The states we have described are just three of many possible states in baseball. Players can learn the optimal action in the other states using a similar process.

It is helpful to list the four steps in the process:

1. Design states that summarize the situation in each time period. These time periods typically match the timing in which a decision-maker (a firm) needs to choose actions (whether or not to call a customer).

2. For each action in each state (each state-action pair), estimate the current period

rewards and transition probabilities from the current state to any possible states. In our baseball example, the transition probability for the State 1- Steal pair, is β to State 2 and $(1-\beta)$ to State 3.

3. Estimate the value of each state-action pair for a given policy ("Policy Evaluation"). This is typically the expected value of current rewards (if any) together with the expected value of the rewards in future states. For example, in our baseball example, the value of the State1-Steal pair is calculated as $0.6675 * \beta + 0.0986 * (1 - \beta)$.

4. Improve the policy by choosing the action in each state that has the highest value ("Policy Improvement"). In State 1 the runner only steals second base if $0.5088 < 0.6675 * \beta + 0.0986 * (1 - \beta)$.

Standard Reinforcement Learning (RL) algorithms mimic this logic; a typical RL algorithm uses the same four steps. Steps 1 and 2 can be thought of as preliminary steps. Once the states are designed and the transition probabilities estimated, we do not re-visit Steps 1 and 2. However, because the value of each state-action pair depends upon the actions in future states, Steps 3 and 4 (Policy Evaluation and Policy Improvement) are solved iteratively. We value the current state-action pairs using an initial policy (choice of actions in each state), we then use the values of each state-action pair to improve the policy. We can then re-value the state-action pairs under this improved policy, and then further improve the policy. This is the standard iterative approach for solving RL problems, and there are theoretical guarantees on its performance (we discuss this iterative approach in greater detail in Section 6).

This approach is directly applicable to targeting a sequence of marketing actions. However, because most RL methods are based on the MDP framework, designing states is important. The states must satisfy the Markov property, which requires that the rewards in the current state and the transitions to future states depend only upon which state a customer is in, and they do not depend upon past states. In our baseball example, the expected number of runs in State 2 must not depend upon how the status of the inning arrived at State 2. The probability of future outcomes should be the same irrespective of whether the inning was in State 1 and the runner stole second base versus a batter hit a double with no runners and 1 out in the inning. In this respect, the states must be memoryless. As we will demonstrate, even small breaches of the Markov assumption can lead to dramatic errors. To help understand why, consider the implications of systematic errors in the transition probabilities. The errors will introduce inaccuracies in predictions of which states the system will transition to in future periods, and these inaccuracies will generally survive and perpetuate throughout the dynamic system.

In this paper, we address the following research question: how do we construct Markov states from historical transaction data so that we can use RL to solve sequential targeting problems in marketing? Standard approaches to designing states from transaction data will not yield states that satisfy the Markov property. For example, states designed using so-called "RFM" measures (e.g., Bult and Wansbeek 1995, Bitran and Mondschein 1996), generally do not satisfy the Markov property. The approach we propose relies on a key insight: states that satisfy the Markov property contain the same information about the probability of future outcomes as the raw training data. The predictive state representation (PSR) literature in computer science exploits this insight by proposing that we use the training data to calculate probabilities over how agents (customers) will respond to future (firm) actions.[1] We group customers together if they are expected to respond in the same way to future marketing actions, and the resulting states are guaranteed to be Markov.

Intuitively, we first construct predictions over how customers will respond to a firm's marketing actions. We then divide these continuous probabilities into discrete buckets to define discrete states. The main challenge is that there is an infinite combination of future customer behaviors (future events) to predict. To address this, we identify a finite set of "core" future behaviors, from which we can estimate the probability of any future customer behavior. We then focus on the probabilities of these core future behaviors to create discrete states. The resulting discrete states

[1] See for example: Littman et al. (2001) and Singh et al. (2004). Notice that these probabilities are not propensity scores. They measure the responsiveness of customers to the firm's marketing actions.

satisfy Step 1 of the four steps in a standard RL algorithm (described above). Once we design the states, the remaining three steps can be performed using standard methods.

One of our contributions is to recognize that this problem is often easier to solve in a marketing setting, because transactions by individual customers are often relatively sparse. This can greatly reduce the range of future events that we need to consider, and considerably lower the computational burden. A problem that is computationally challenging in many settings becomes more feasible in many marketing settings.

We show that the states designed by our proposed approach are robust to the introduction of non-Markov dynamics in the data generation process. To accomplish this, we need to know the ground truth of the data. For this reason, we use simulated data, which is constructed based on actual email experiments conducted by a large company. We introduce non-Markov distortions to both the rewards and the transition probabilities (Step 2 in the four steps listed above). We then show that our value function estimates are robust despite these distortions.

Our contributions are three-fold. First, we identify an important marketing problem that can be solved by RL techniques. More importantly, we point out that a key to the successful application of these RL methods to sequential targeting in marketing is to construct states that satisfy the Markov property. Second, we adapt the methods proposed in the PSR literature to our setting by exploiting the sparsity in transaction data. Sparsity arises if only a relatively small proportion of customers purchase at each time period, which is a feature that is common in customer transaction data. Third, we propose a novel approach to validation, which highlights the importance of constructing states that satisfy the Markov property.

The paper continues in Section 2 with a review of the literature. We illustrate the criteria of "good" states in Section 3 and in Section 4 introduce the conceptual foundations of our proposed method. Section 5 describes the proposed approach in detail. A standard RL dynamic optimization method is briefly reviewed in Section 6, and Section 7 describes the validation of the method. The paper concludes in Section 8 with a discussion of limitations.

Chapter 2

Literature Review

There is an extensive literature investigating how to target firms' marketing actions. Many of these papers focus on the myopic problem of choosing which marketing action to give to each customer in the next period. For example, Simester et al., (2019a) investigate seven widely used machine learning methods, and study their robustness to four data challenges. Dubé and Misra (2017) describe a method for customizing prices to different customers. Ostrovsky and Schwarz (2011) propose a model for setting reserve prices in internet advertising auctions. Rafieian and Yoganarasimhan (2018) investigate the problem of targeting mobile advertising in a large network.

Separate from the targeting literature, there have also been many studies demonstrating that marketing actions can have long-term implications (see for example Mela et al. 1997, Jedidi et al. 1999, and Anderson and Simester 2004). However, relatively few papers have attempted to optimize a sequence of marketing actions to solve long-run sequential targeting problems. One of the earliest attempts to solve this problem was Gönül and Shi (1998), who focused on the optimization of a sequence of catalog mailing decisions. They used the structural dynamic programming model proposed by Rust (1994), and jointly optimized both the firm's catalog mailing policy and the optimal customer response. Khan et al., (2009) used a similar approach to optimize digital promotions for an online grocery and drug retailer. Simester et al., (2006) solve the sequential catalog mailing policies using a simple and straightforward model, which introduces Reinforcement Learning methods to marketing. They compare the performance of the model with the firm's current policy in a champion versus challenger field experiment. The model shows promise but under-performs for high-value customers. One possible explanation for this is that the state space they construct is not guaranteed to satisfy the Markov property. Zhang et al., (2014) consider a dynamic pricing problem in a B2B market using a hierarchical Bayesian hidden Markov model, and Hauser et al., (2009) use a partially observable Markov decision process model to illustrate on website morphing problem. We will highlight the difference between this hidden Markov approach and our proposed approach in later discussion.

Many studies have focused on estimation and optimization. In contrast, our focus is on the design of state spaces. The design of state spaces can be thought of as a distinct problem, in the sense that the approach we propose can be used as an input to structural dynamic programming and other RL methods.

Other approaches have been proposed in the literature for overcoming the potential failure of the Markov assumption when constructing states. One popular method in marketing and economics is to explicitly consider history when constructing states. By extending the focus to k-historical periods, it may be possible to ensure that the state space is at least kth-order Markov. Another approach is to use Partially Observable Markov Decision Processes (POMDPs) (Lovejoy, 1991). This class of methods combine MDPs (to model system dynamics) with a hidden Markov model that connects unobserved states to observations. POMDPs have proven particularly well-suited to theory, and there are now several results characterizing the performance of POMDPs and providing guarantees on their performance. Unfortunately, POMDPs have not been as useful in practice. When comparing POMDPs with PSRs (which is the focus of this paper), Wingate (2012) identifies three advantages of PSRs. First, future predictions are easier to learn from data compared with POMDP. The reason is that predicted probabilities of future events are observable quantities, while states in POMDPs are unobservable. Second, all finite POMDP and k-history methods can be transformed to a future prediction state expression, while the converse is not true. Third, due to the second advantage, dynamic optimization methods developed for POMDP and k-history approaches have the potential to be applied to states designed using the PSR approach discussed in this paper.

Finally, this paper can also be compared with recent work in marketing studying how to evaluate targeting models. The previous work has focused on validating myopic targeting problems. Simester et al., (2019b) propose the use of a randomized-by-action (RBA) design to improve the efficiency of model comparisons. Hitsch and Misra (2018) also propose an RBA design, and stress the comparison of direct and indirect methods. Three recent papers have proposed the use of counterfactual policy logging in different settings to improve the efficiency of model comparison methods (see Johnson et al. 2016, Johnson et al. 2017, and Simester et al. 2019b).

Chapter 3

Criteria for Designing States

We first introduce the concept of a Markov Decision Process (MDP) and provide formal definitions of states and the Markov property. We then apply these definitions to the task of designing an optimal sequence of marketing actions using historical transaction data. We use the simple "RFM" approach as an illustration to highlight the danger of neglecting to construct states that satisfy the Markov property.

3.1 States and MDPs

A "state" is a concept that originated from Markov Decision Processes (MDPs). An MDP is a model of framing an agent's learning problem, and the basis of many Reinforcement Learning algorithms. An MDP is composed of four elements (S, A, R, P): S represents the state space; A is the action space; $R : S \times A \to R$ denotes the stochastic numerical reward associated with each state-action pair; $P : S \times A \to S$ denotes the stochastic transition probability associated with each state-action pair. The transition probabilities represent the dynamics of the MDP. In this paper, we focus on a finite MDP where (S, A, R) all have a finite number of elements.

The baseball example can be used to illustrate each of these concepts. Here, the runner is the agent, and everything that he interacts with is the environment. The runner interacts with the environment at discrete time periods $\{0, 1, 2, 3, \dots\}$. These time periods typically match the occasions on which the agent makes decisions. In our baseball setting, the time periods might refer to each pitch, because the runner decides to steal or not on each pitch.

At each period t, a runner in state $s_t \in S$ selects an action $a_t \in A$. In this example, the runner has two action options: $A = \{Steal, NotSteal\}$. As a result of the runner's action a_t and the state of the environment s_t, the agent receives a numerical reward $r_t \in R$. Here, R is the number of runs scored in the time period, which is a number and can be 0. The reward R is generally a random variable that is a draw from a stochastic distribution. This distribution may be degenerate and so the reward in that time period may be known with certainty (in our State 1 the runner will not score a run irrespective of whether or not the runner steals second).

In period $t+1$, the runner transits to state, $s_{t+1} \in S$. This could be a transition back to the same state (this happens with certainty if the runner does not steal) or a transition to a different state. The transition probabilities depend upon the state in period t and the action taken in that period $a_t \in A$. The transition probabilities are stochastic, although the stochastic probability distribution may also be degenerate (such as when the runner does not steal). Unless the transition probabilities are degenerate, the runner enters period $t+1$ with some uncertainty about which state he will transition to in period $t+1$. We model these dynamics using the transition probability P. Specifically, consider $s_t = s \in S$, $a_t = a \in A$ and $s_{t+1} = s' \in S$, then the transition probability is: $Pr\{s_{t+1} = s^{'} | s_t = s, a_t = a\}$.

In our simple example, we defined the states in baseball using the location of runners on bases and the number of outs in the inning. However, there may be many other factors that affect the reward R and the transition probability P. For example, the success of the attempt to steal may depend upon the catcher, the pitcher, the second baseman, the weather, the stage of the season, past progress of the game, etc. In an MDP framework, we summarize all of these environmental conditions using the concept of states. A critical requirement is that in an MDP framework, the reward and transition probabilities in each time period only depend upon the current period's state and the agent's action. Formally, $r_t(s_t, a_t)$ and $p_t(s_{t+1}|s_t, a_t)$ only depend upon s_t and a_t. They do not depend upon any previous states or actions, such as s_{t-1} and

a_{t-1}. This assumption is the Markov assumption in an MDP. This requirement is one place that Reinforcement Learning models can fail when applied to marketing problems (and is the focus of this paper).

We stress that the Markov assumption is not a restriction on the decision-making process, but on the state. The agent's decision process may not be memoryless, but we can carefully design the state space to make it memoryless. For example, when defining states we can include all the past information in the environment that is relevant for the current period rewards and state transitions. If we can accomplish this, we are guaranteed to satisfy the Markov assumption. However, we cannot use all past history data because it is computationally infeasible. The transition probabilities need to be updated recursively (see the discussion of the state updating process at the end of Section 4). Thus, we need to find sufficient statistics for all of the relevant information, which is the (non-trivial) goal of state construction.

Not all Reinforcement Learning methods depend upon the Markov assumption. Approximate Dynamic Programming methods (which include Deep Reinforcement Learning), do not rely upon the Markov assumption. The method we use for learning (after defining the states) belongs to the tabular family of Reinforcement Learning methods, which have well-established theoretical convergence properties. In contrast, approximation methods have known limitations in the robustness of their convergence. In addition, approximation methods cannot augment the state representation with memories of past observations. They must still embed the k-history idea in state representations.[1]However, approximation methods have the ability to deal with large

[1]We can illustrate using linear approximation methods as an example. Linear methods assume that the approximate function, $\hat{v}(,\omega)$, is a linear function of the weight vector ω. Corresponding to every state s, there is a real-valued vector $x(s) \equiv (x_1(s), x_2(s), \ldots, x_d(s))^T$, with the same number of components as ω. The vector $x(s)$ is called a feature vector representing state s. For example, in our baseball example, features can include bases, outs, components, etc. Linear methods approximate the state-value function by the inner product between the weight ω and the features $x(s) : \hat{v}(,\omega) \equiv \omega^T x(s)$. The mapping from states to features has much in common with the familiar tasks of interpolation and regression. We can use various functional forms, for example, linear, polynomials, Gaussian, etc. No matter which functional form we use, we still look at the original data space. To control is a finite vector over time, we need to reduce feature representations to a limited number of previous periods (which is the k-history idea). Deep Reinforcement Learning, assumes a nonlinear function form between the weight and the features $x(s)$, but still cannot incorporate all past information in the data.

scale problems, which is a limitation of the learning method we present here. We see this as an important future research opportunity and will discuss it further in Section 8.

We next apply the MDP framework to the design of dynamic marketing policies in which a firm wants to optimize a sequence of marketing actions. We will illustrate the error introduced by the failure of the Markov assumption.

3.2 MDPs and Targeting

Many firms want to “target” their marketing actions by matching different marketing actions to different customers. The marketing actions may include promotions, pricing, advertising messages, product recommendations, outbound telephone calls etc. For ease of exposition, in this paper, we will focus on the targeting of a sequence of email advertisements. The procedure for applying the method to other marketing problems should be clear from this email application.

We interpret the firm’s sequence of mailing decisions as an infinite horizon task and seek to maximize the discounted stream of expected future profits. Recall that time is measured in discrete periods, which in this application is the mailing date for each email campaign. We again use S to denote the set of discrete states. In sequential targeting problems, each state groups together “similar” customers at each time period. When we say “similar”, what we hope for is that customers in the same state will respond in a similar way to future targeting policies. Clearly, this is an important challenge. How to design such a state space that satisfies the Markov assumption is the problem that we tackle.

There are two possible actions in this problem: mail or not mail. To stress the relationship between action and state, we use notation $a_{ts} \in A = \{0, 1\}$, where $a_{ts} = 1$ denotes a decision to mail at period t to all customers in state s. The numerical reward at each period $r_t(s_t, a_{ts})$ is the net profit earned from a customer’s order less (any) mailing costs, and the transition probability $p_t(s_{t+1}|s_t, a_{ts})$ gives the probability of becoming a customer in state s_{t+1} when the customer is in state s_t at period t and receives mailing action a_{ts}.

A policy ($\pi : S \rightarrow A$) describes the mailing decision for each state. The goal of the firm is to find a policy that maximizes the discounted expected future profits:

$$V^{\pi}(s_0) = E_{\pi}[\sum_{t=1}^{\infty} \delta^t r_t(s_t, a_{ts}|s_0)]$$

Here, δ is a discount factor between time periods, and s_0 is the initial state at period zero. In an MDP framework, we normally describe $V^{\pi}(s_0)$ as the value function of state s_0 under a policy π.

3.3 Illustration of the "RFM" Approach

The traditional industry approach to segmenting customers for direct mail and email campaigns is to group customers according to the Recency, Frequency and Monetary value of customers' previous transactions.[2] In this subsection, we briefly show that this (very) simple approach does not satisfy the Markov property, which introduces non-negligible estimation errors to value function estimates. To illustrate how the "RFM" approach can be used to construct states, we simulate an MDP. The primitives are constructed based on email experiments provided by a large company. We first briefly summarize these experiments.

We use historical transaction data for customers involved in a sequence of six experiments. The experiments were all designed to cross-sell new products to existing customers by advertising the benefits of the new products. In each experiment, customers were randomized into two experimental conditions. Customers in the treatment condition received an email advertisement, while customers in the control condition did not. The randomization was conducted at the individual customer level. We summarize the sample size in each of the six experiments in Table A.1.

There were 136,262 customers that participated in all six experiments. We will

[2]"Recency" measures the number of days since a customer's last purchase. "Frequency" measures the number of items customers previously purchased. "Monetary value" is the average price of the items by each customer.

restrict attention to these 136,262 customers when we create the primitives for our simulated data. The outcome from each experiment was measured by calculating the profit earned in the 90 days after the mailing date less mailing costs. The distribution of outcomes includes a large mass of customers from which the firm received no revenue. For these customers, the profit was either zero or negative, depending upon whether they were in the Treatment condition (in which case the firm incurred a mailing cost). There was also a long tail of outcomes, with a handful of customers contributing a very large amount of revenue.

We summarize the distribution of outcomes by grouping the outcomes into five discrete buckets. The first bucket included the customers for whom profit was zero or negative. The customers in the next four outcome buckets all contributed positive profits. Because the profit levels are confidential we do not disclose the cutoff limits (recall that we only use these outcomes to provide the primitives to generate simulated data). In Figure A.1 we report a histogram of the customer outcomes across the six experiments. The unit of observation is a customer in an experiment and the columns add to 100%.

We use the outcomes from this experiment to provide primitives to generate a dataset to use for illustration. The dataset is generated from an MDP with five states and two actions: $s = \{1, 2, 3, 4, 5\}$ and $a = \{0, 1\}$. The state in period t corresponds to the profit earned from the experiment in period t. For example, if the outcome in period t corresponds to the third outcome bucket, the state in period t is State 3. The rewards $R^M : S \times A \rightarrow R$, transition probabilities $P^M : S \times A \rightarrow S$, and policy $\pi^M : S \rightarrow A$ to generate this MDP are all correspondingly generated from the actual campaign experiment data.[3]

We constructed a RFM state space by discretizing the recency and frequency variables to three levels, and the monetary value variable to five levels. This yields

[3]The rewards are the midpoint of the profit earned in each of the discrete outcome buckets. If the outcome in period t corresponds to the third outcome bucket, the rewards for that customer in period t is equal to the midpoint of the rewards in that bucket. The transition probabilities for each state-action pair are calculated by counting the proportion of times a customer in that state that received that action (mail or not mail) transitioned to each of the other state in period $t+1$. Because the actions (mail and not mail) were randomized, the policy is a stochastic policy, and reflects the percentage of times a customer in that state received each action.

a state space with 45 states. At each period in the simulation data, we evaluate the RFM variables of each customer and assign the customer into the corresponding RFM state.

Because the data is simulated we know the true state space in this dataset, and so we can use standard methods to calculate the true weighted value function (we delay discussion of the methods until Section 6). When we use the true state spaces, the weighted value function estimate for policy π^M is 15,039. However, when we use the RFM states, the value function estimate for the same policy becomes 20,161. This error is solely attributable to the design of the state spaces. Further investigation reveals that the RFM approach does not yield states that satisfy the Markov property. About 62.50% of the states constructed using the RFM method are Markov.

This illustration highlights the role that the design of the states can play in contributing to the error in RL methods. In Section 7 we will show that these errors can be greatly reduced if we design states that satisfy the Markov property. We next introduce the method that we propose for accomplishing this purpose.

Chapter 4

Main Idea of State Construction

The goal while constructing states is to create a mapping from the historical transaction data to some manageable dimension space that satisfies the Markov property. What we propose is to map from the space of historical transaction data to probability space, representing predictions of how customers will respond to future marketing actions. The challenge is to find a set of events that are sufficient for the prediction of all other future purchasing events. We elaborate on this concept below.

4.1 State Representation

To understand our state representation idea, we need to introduce the concept of an observation, denoted by O. The firm's interaction with its environment can now by described as: at each time period t, the firm executes a mailing action $a_t \in A = \{NotMail, Mail\} = \{0, 1\}$ and receives an observation $o_t \in O$. Different customers can receive different a_t, and the observation o_t can be high-dimensional. For example, observation o_t can be a two-dimensional variable, with o_t^1 corresponding to the number of units a customer purchased at period t, and o_t^2 corresponding to the revenue from the customer's purchase at period t. Observation o_t can also include customer characteristics, such as age, gender, and environmental variables measuring seasonality. Because the observation can measure purchasing, the current period reward (in an MDP) can be represented in the observation. For ease of exposition, we will treat observation o_t as a one-dimensional variable. Specifically, we assume $o_t \in O = \{NotBuy, Buy\} = \{0, 1\}$.

Suppose the firm is at time period t. An action-observation sequence is a sequence of alternating actions and observations. A history, denoted as $h = a_1 o_1 a_2 o_2 \ldots a_t o_t$, describes the action-observation sequence that the firm has experienced from the beginning of time through period t. A test, denoted as $q = a^1 o^1 a^2 o^2 \ldots a^n o^n$, describes the action-observation sequence that might happen in the future. We define the predicted probability of a test ($q = a^1 o^1 a^2 o^2 \ldots a^n o^n$) conditional on a history ($h = a_1 o_1 a_2 o_2 \ldots a_t o_t$) as:

$$p(q|h) \equiv \Pr\{o_{t+1} = o^1, o_{t+2} = o^2, \ldots, o_{t+n} = o^n | h, a_{t+1} = a^1, a_{t+2} = a^2, \ldots, a_{t+n} = a^n\}$$

We will use an example to help illustrate the meaning of this probability. In an email (or direct mail) targeting problem, the firm observes whether each customer received each past email, and whether the customer bought in each past period. This is the history we refer to. For example, consider the test: ($q = a^1 o^1 a^2 o^2$), where $a^1 = a^2 = 1$ and $o^1 = o^2 = 1$. The conditional probability $p(q|h)$ represents the predicted probability that a customer bought in both period $t + 1$ and period $t + 2$, conditional on the customer's past history and receiving an email at period $t + 1$ and period $t + 2$.

The central idea of Predictive State Representations (PSRs) is to choose a set of tests, and use the predictions of those tests to construct states. Suppose the set of tests has m elements, we can write this set of tests as $Q = \{q_1, q_2, \ldots, q_m\}$. Informally, tests in Q can be understood as the prediction of future action-observation sequences. To be specific, we use the column vector $p(Q|h) = [p(q_1|h), p(q_2|h), \ldots, p(q_m|h)]^T$ as state representations for each customer. Customers with the same $p(Q|h)$ will be grouped into the same state.

The next question is how to choose the set of tests Q so that the states satisfy the Markov assumption. We require that Q is chosen so that $p(Q|h)$ forms a sufficient statistic for all future test predictions. More formally, for any test q, there exists an

$(m \times 1)$ vector m_q, such that[1]:

$$p(q|h) = p(Q|h)^T m_q$$

Throughout the paper, we assume the existence of such a set of tests Q, and call the tests in set Q as core tests. This is not a strong assumption as for any problem that can be represented by a finite POMDP or a k-history model, there exists a set Q with finite elements that can also represent the problem (Wingate 2012). Moreover, compared with a POMDP, the number of core tests is no larger than the number of hidden states in the POMDP model (Littman et al., 2001).

4.2 Properties of the Constructed State

We next explain why the PSR states satisfy the Markov property. Second, we show that the proposed state representation can be recursively updated.

Suppose we already have some historical data up to period t for each customer, denoted as h_t^i for customer i. In our email setting, h_t^i includes whether each customer received an email and bought at each time period before time t (including period t). The state representation finds a compact summary of the historical data for each customer. A key insight is that: states that satisfy the Markov property contain the same amount of information about the probability of future purchases as the raw history data. Formally, if two customers (denoted as A and B) who have different histories, $h_t^A \neq h_t^B$, are assigned to the same state group, they should share the same probabilities for the next observation:

$$Pr\{o_{t+1} = o | h_t = h_t^A, a_t = a\} = Pr\{o_{t+1} = o | h_t = h_t^B, a_t = a\}$$

If this holds for any customers who are assigned to the same state, we can say that the constructed states satisfy the Markov property. More generally, a Markov state should not only be good at predicting the next observation, but also any higher length

[1]We assume a linear relationship for the sufficient statistic.

observations.[2] The PSR approach is motivated by this insight. If we can find the correct core tests (future events) and accurately estimate the probabilities associated with these core tests, then the resulting states will satisfy the Markov assumption.

The equation above also suggests a test of whether a state satisfies the Markov property. The following conditional independence test provides a necessary (though not sufficient) test of whether a state is Markov: $p(o_{t+1}|a_t)$ is independent of h_t for all observations in the same state.[3] This is the method we used to evaluate how many of the RFM states were Markov at the end of Section 3. We will also use this test to help evaluate our state construction method in Section 7.

We next show that the states constructed here can be easily updated. Consider a specific customer who has a history h_t up to period t. At period $t+1$, this customer receives mailing action a_{t+1} and makes purchasing decision o_{t+1}. We now have a new history up to period $t+1$, $h_{t+1} = h_t a_{t+1} o_{t+1}$. We can calculate the predicted probability $p(q_j|h_{t+1})$ where $q_j \in Q$ by:

$$p(q_j|h_{t+1}) = p(q_j|h_t a_{t+1} o_{t+1}) = \frac{p(a_{t+1} o_{t+1} q_j|h_t)}{p(a_{t+1} o_{t+1}|h_t)} = \frac{p(Q|h_t)^T m_{a_{t+1} o_{t+1} q_j}}{p(Q|h_t)^T m_{a_{t+1} o_{t+1}}}$$

This state-update function is easy to estimate, and applies to all histories and all core tests. We next discuss the empirical construction of states from data.

[2] Notice also that different periods of a specific customer should be treated in the same way as different customers.

[3] This test only considers predictions of the next observation, while Markov states should be good at predicting any length future observations. However, this proposed test offers an important indication of whether a state satisfies the Markov property.

Chapter 5

Constructing States from Data

The discussion in Section 4 reveals that we need to do two things to learn states from data. First, we need to find the core-test set Q. Second, we need to estimate the conditional probability $p(Q|h)$ for any possible h. We conclude this section by providing a theoretical guarantee for our proposed approach.

5.1 Finding Core Tests

We first discuss how to find the core-test set Q. This is a challenging task, and early attempts to solve the problem were not able to provide any theoretical performance guarantees (see for example James and Singh 2004 and Wingate and Singh 2008). We propose an adaptive approach based on James and Singh (2004) that exploits ideas from linear algebra (Golub and Van Loan, 2013). Our method, which is easy to understand and implement, exploits sparsity. As we discussed in the Introduction, a common feature of transaction data is that many customers do not purchase every period, and so the incidence of purchases is "sparse" (see for example the distribution of outcomes in Figure A.1). This feature will greatly simplify the process of finding core tests.

To understand our proposed approach, we introduce a new concept, the system dynamic matrix. The system dynamic matrix is a way to represent a sequential decision-making problem (without introducing any new assumptions). Consider an ordering over all possible tests: $q_1, q_2 \dots$. Without loss of generality, we assume that the tests are arranged in order of increasing length. Within the same test length, they are ordered in increasing values of our binary outcome indicator.

We define an ordering over histories $h_1, h_2 \dots$ similar to the ordering over tests. The difference is that we include the zero-length/initial history $\varnothing$ as the first history in the ordering. The system dynamic matrix is such that the column corresponds to all tests by ordering, the row corresponds to all histories by ordering, and the entries are the corresponding conditional probability $p(q|h)$. An example is given in Figure A.2.

The core tests correspond to the linearly independent columns in the system dynamic matrix. This means that to find the set of core tests Q, we just need to find the linearly independent columns of the system dynamic matrix (which has infinite-dimension). If we know all of the matrix entries of the first row, we can get all of the other matrix entries using the state update relationship (which we presented at the end of Section 4). Specifically, we can obtain a specific entry of the system dynamic matrix through:

$$p(q|h) = \frac{p(hq)}{p(h)}$$

Thus, if we know all of the matrix entries of the first row, we can get all of the other matrix entries using this state update relationship.

The difficulties are two folds. First, we do not know the true conditional probability $p(q|h)$. Instead, we need to estimate these probabilities from data. This introduces the potential for estimation errors, and these errors could make linearly independent columns appear dependent (it is also possible, though less likely, that dependent columns may become independent). Second, in practice the length of the available data is finite. For example, we used just six experiments in the example at the end of Section 3. Yet, we are trying to learn the independent columns of an infinite dimension matrix. Because of these data challenges, we can only hope to find approximately correct core tests.

We consider the problem in two steps. First, we will assume that the probabilities in the system dynamic matrix are known exactly, and consider how to find the linearly independent columns.

1. Delete history rows whose corresponding tests have zero probability at zero-length history. If a test has zero probability with zero length history, it also has zero probability with longer history. The histories that are not deleted provide the rows of the submatrices used in the next steps.

2. Start from the submatrix containing all tests up to length one. Calculate the rank of this matrix, and the corresponding linearly independent columns.

3. Expand the submatrix to the one whose columns are the union of all length-one tests, the linearly independent tests found so far, and all one-step extensions of these independent tests.[1]

4. Calculate the rank and linearly independent columns of this new submatrix.

5. Repeat Steps 3 and 4 until the rank does not change.

Step 1 of this algorithm is where we exploit the sparsity that is common in transaction histories. For example, in our implementation data, there are five observation levels and two action levels. This results in 10 length-one tests, 100 length-two tests, 1000 length-three tests, etc. We can see that the system dynamic matrix grows exponentially. However, because transactions are sparse, our method will delete many history rows in Step 1. This greatly reduces the computational burden when we expand the submatrix in Steps 3, 4 and 5. In our application, the sparsity in the transaction data allows us to delete approximately 80% of the history paths, which leaves a computationally feasible submatrix.

We can also show that as the size of the system dynamic matrix increases, the level of sparsity grows at least as quickly.

[1]Given a test q, a one-step extension refers to a new test aoq, where $a \in A = \{0,1\}$ and $o \in O = \{0,1\}$.

Result 1. The zero history rows (sparsity) grow exponentially as the history length increases.

Proof: If a test $q^{'}$ has zero probability at zero-length history, all one-step expansions of test $q^{'}(q^{'}ao)$, and extensions of test $q^{'}(aoq^{'})$ have zero probability at zero-length history. If we imagine an infinite dataset, zero probability of test $q^{'}$ at zero-length history means that we will not observe any pieces of $q^{'}$ in this infinite dataset. Thus, any pieces of $q^{'}ao$ and $aoq^{'}$ will not exist in this infinite dataset. Thus, the "sparsity" will also grow exponentially. More importantly, it grows at a faster rate than the growth rate of the system dynamic matrix.

This result is important. It increases confidence that our proposed algorithm for finding core tests will continue to be computationally feasible as the size of the system dynamic matrix grows.

Recall that we have focused on cases in which we know the probabilities in the system dynamic matrix exactly. If we continue to make this assumption we can provide a guarantee on the performance of the algorithm.[2]

Result 2. The core tests identified using this 5-step algorithm are correct if all entries in the system dynamic matrix are known exactly.

Proof: See Appendix.

This result guarantees that the core tests obtained using this method are correct. We will label these linearly independent rows as "core histories".

In practice, there are likely to be estimation errors in the probabilities in the system dynamic matrix. This can raise several issues. First, it can influence how we obtain the rank and linearly independent columns in Steps 2 and 4. We relegate a discussion of these details to the Appendix. Second, it becomes more difficult to provide a guarantee on the performance of the method. To do so we need an asymptotic result that as the time-period (denoted as T) of the data increases to

[2]If we know the probabilities in the system dynamic matrix exactly, another implication is that if a test $q^{'}$ has zero probability at zero-length history, test $q^{'}$ will not be one of the core tests. The reason is that for any history h, $p(q^{'}|h)$ will be zero. It seems that then we can exclude $q^{'}$ in our five-step algorithms. However, we choose not to do this because under our probability estimation method (introduced in next subsection), $p(q^{'}|h)$ may not be zero even if $p(q^{'}|\varnothing)$ is zero. This adjustment helps reduce the effect of empirical errors in probability estimation on the finding of core tests.

infinity ($T \to \infty$), the set of core tests we find are close to the true core tests. The result will not hold if we consider a general matrix structure. However, one feature in the customer transaction behavior can help with this asymptotic result.

The observation is that customers' purchase behavior will not depend on events that happened infinite time ago. In practice, there will generally exist a time T_0 (which can be very large) such that customers' purchase behavior at most depend on T_0 time ago events. The implication is that all histories with length higher than T_0 have the same predicted probabilities of all future events as the corresponding length-T_0 histories. Thus, if we have enough time period data, that is, $T \geq T_0$, we will find the correct core tests using our five-step algorithm based on Result 1. We will also demonstrate in Section 7 that the states that we design using the core tests behave well in practice.

Our discussion of the theoretical properties of this method has also been based upon the assumption that we can learn the entries in the system dynamic matrix exactly. In reality, we need to estimate these probabilities from a finite sample of data. We leave the issue of how to estimate the predicted probabilities $p(q|h)$ until the next subsection.

We finish the discussion in this subsection with two additional comments. First, while the five-step algorithm and Results 1 and 2 are both new, we want to acknowledge that they build upon a similar method proposed by James and Singh (2004). A difference is that James and Singh (2004) propose submatrix expansion on both histories and tests in each iteration (we just expand the tests). This method is not guaranteed to find the correct core tests even if all entries in the system dynamic matrix are known. Their method also faces computation challenges, which we alleviate by exploiting the sparsity in customer transaction data. Wingate and Singh (2008) propose using randomly selected tests as core tests, but this approach is also not guaranteed to find the correct core tests.

Second, the approach we propose may have difficulties dealing with tests that have long lengths. We discuss this issue in Section 8.

5.2 Calculating Predicted Probabilities

Once we obtain the core tests, we can derive the state for each customer at each period by calculating the (conditional) probabilities of the core tests. Customers with the same probabilities will be assigned to the same state group. Estimating these probabilities from data is conceptually straightforward, but there are several details that deserve comment.

The goal is to estimate the predicted probabilities $p(q|h)$ for the core tests. A straightforward approach is to simply count how many h and hq appear in the dataset, and then take the fraction of hq over h.[3] For example, suppose h = "01" = $\{a_1 = Notmail, o_1 = Buy\}$ and q = "11" = $\{a^1 = Mail, o^1 = Buy\}$. Then we can just count how many times "01" and "0111" appear in the dataset, and calculate the "0111" occurrences as a proportion of "01" occurrences.

In order to do this, we need to define time period 0. Notice that in our method, each period's data has a different length of history. This makes the definition of time period 0 important, because the calculation of the probabilities could be sensitive to the choice of starting time. Moreover, if we define time period 0 to always start with the first time period in our data, we will place more reliance on the data at the start of our data period, which places more reliance on the least recent data. This could make our estimation very sensitive to non-stationarity.

To address this concern, we propose the following sampling method. Suppose we have data on T periods and N customers. We divide the N customers randomly into M groups, and for each group we randomly assign time period 0 to different calendar time periods. Each of the customers in the same group are assigned the same calendar time period as period 0, but different groups are assigned different calendar time periods as period 0. For example, recall the six email experiments described at the end of Section 3. There were 136,262 customers that participated in all six experiments. We would randomly divide these customers into six groups. In

[3] It is possible that the length of hq is larger than the total time period in the data. For this reason, when we search for the core tests, we restrict attention to submatrices for which we can directly estimate the predicted probabilities $p(q|h)$.

one group, period 0 would correspond to the first experiment, another group would treat experiment two as period 0, etc. For the customers for which experiment two is period 0, period 1 would be experiment 3, period 2 would be experiment 4, and so on. The benefit of this approach is that it ensures the estimated probabilities are less sensitive to temporal variation in outcomes. A limitation is that we do not use all of the data; in groups for which there are experiments before Period 0, we do not use these earlier experiments to estimate the probabilities.

Notice that we use the predicted probabilities in the system dynamic matrix in two stages: (a) to find the core tests in the system dynamic matrix, and (b) after finding the core tests we group customers into states according to the conditional probabilities associated with these core tests. We can either re-use the same assignment of customers to period 0 in both places, or we can re-sample after finding the core tests (to re-allocate period 0 across customers). In our implementation, we choose to resample. This has the advantage of reducing the relationship between estimation errors in the two stages. However, it may also increase the risk that random variation results in breaches of the Markov assumption in some states.

Using our data, we can observe which tests happen after which histories. For example, suppose we have a dataset with 3 time periods. Consider a specific customer A who is assigned so that period 0 is the first period in the data. Suppose the data we observe for customer A is "011100"$= \{a_1 = Notmail, o_1 = Buy, a_2 = Mail, o_2 = Buy, a_3 = Notmail, o_3 = Notbuy\}$. For customer A, after history $h =$"01", tests $q =$"11" and $q =$"1100" are observed. For some tests our data length is too short to observe whether the test succeeds or not. In these situations, we use the predicted probabilities, which we can estimate using the state-update function (provided at the end of Section 4).[4]

Probabilities are continuous numbers. Unless we round the probabilities, we may not find two observations in the same state. The key to the use of rounding is that we need to do all probability approximation before we start identifying the core tests.

[4]To use the state-update function, we also need to know the parameter vector m_q for test q.This vector can be obtained through $p^{-1}(Q|H)p(q|H)$ where H includes all possible histories. We calculate the matrix inverse using the Moore-Penrose pseudoinverse.

In this way, our rounding does not interfere with the Markov property of the states. If we round the probability after we derive the core tests, we risk introducing errors that may breach the Markov assumption in some states.

Finally, we may encounter covariate shift. Covariate shift arises if the distribution of the data used for training a targeting model is different than the data used for implementation (see for example Simester et al. 2019b). In our setting, the probability vector at the last period, which will guide the targeting policy in a sequential targeting problem, might not occur in any prior period. For those probability vectors, we have no observed outcomes. To address this issue, we assign customers with these probability vectors (in the last period) to the nearest state.[5]

In this section we have discussed how to use a sample of historical transaction data to create Markov states. In the next section we (briefly) discuss how standard Reinforcement Learning methods can use these states to estimate value functions and identify optimal policies.

[5]We measure distance using the L2 norm on predicted probability vectors.

Chapter 6

Dynamic Optimization

Recall that in the Introduction, we mention four steps in a standard Reinforcement Learning algorithm. Given that we have designed a discrete state space (the first step), we now discuss the remaining three steps. There are many standard methods to use for these steps. Because this is not the focus of the paper, we use the simplest approach, which is easy to understand and implement. This method also has well-established theoretical guarantees.

6.1 Estimating Current Period Rewards and Transition Probabilities

Just as the runner in baseball needs to know the transition probabilities from one state to another state for both "not steal" and "steal" actions, we need to estimate the transition probabilities for both the "mail" and "not mail" actions. In terms of an MDP, we need to estimate the stochastic transition probabilities $P : S \times A \to S$.

We use a simple nonparametric approach to accomplish this. For each state and mailing decision, we observe from the historical data the proportion of times customers transitioned to each of the other states. The benefits of this approach are discussed in detail in Simester et al., (2006).

6.2 Estimating the Value of Each State for a Given Policy

This step is normally called "Policy Evaluation" in the Reinforcement Learning literature. In our baseball example in the Introduction, the number of expected runs in States 1, 2 and 3 were calculated as a simple average of the number of runs scored from the states in past Major League Baseball games. However, these outcomes depend upon how the game will be played in future states. In particular, the value function under an arbitrary policy $\pi(s)$ can be written as:

$$V^{\pi}(s) = E_{r,s'}[r(s,\pi(s)) + \delta V^{\pi}(s')|s,\pi(s)]$$

We adopt the notation introduced in Section 3. Suppose we use $r(s,a)$ to represent the expected rewards in state s when the firm chooses mailing action a, then we can rewrite the value function as:

$$V^{\pi}(s) = \tilde{r}(s,\pi(s)) + \delta \sum_{s'} p(s'|s,\pi(s))V^{\pi}(s')]$$

With a slight modification of notation, we can express the above equation in vector form. Let V^{π} denote the vector with elements $V^{\pi}(s), r^{\pi}$ with elements $r(s,\pi(s))$ and p^{π} (a matrix) with elements $p(s^{'},\pi(s))$. Thus, $V^{\pi} = r^{\pi} + \delta p^{\pi} V^{\pi}$, and we can obtain the value of each state under an arbitrary policy from $V^{\pi} = (I - \delta p^{\pi})^{-1} r^{\pi}$.

To estimate $V(s)$, we also need to estimate the one-period expected rewards $r(s,a)$ for each state-action pair. Similar to estimating the transition probabilities, we estimate the one-period rewards using a simple nonparametric approach; we calculate the average one-period reward for each state and mailing decision.

To estimate $V(s)$, we also need to estimate the one-period expected rewards $r(s,a)$ for each state-action pair. Similar to estimating the transition probabilities, we estimate the one-period rewards using a simple nonparametric approach; we calculate the average one-period reward for each state and mailing decision.

6.3 Choosing the Optimal Policy

In our baseball example, the runner chooses the optimal action in state 1 using the value function estimates associated with the two state-action pairs in state 1: the runner only steals if $0.5088 < 0.6675 * \beta + 0.0986 * (1 - \beta)$. When there are many states, for any given policy we can calculate the value function for each state-action pair. We can then improve the policy by choosing the action with the highest value in each state.

Notice that when we change the policy (the "Policy Improvement" stage) this recursively changes the value function estimates for each state-action pair, and so we must then re-evaluate the improved policy (the "Policy Evaluation" stage). We can then try to further improve the policy. This is the idea behind the classical policy-iteration algorithm, which we use in this paper to find the optimal policy. If the current policy already chooses the action with the highest value in each state then the current policy is optimal.

In practice, the policy-iteration algorithm starts with any arbitrary policy for which we estimate the value function. We use this value function to improve the policy, which yields a new policy with which to begin the next iteration. The sequence of policies improves monotonically until the current policy is optimal. The policy-iteration algorithm is guaranteed to return a stationary policy that is optimal for a finite state MDP (Bertsekas, 2017). However, this guarantee only holds if the states satisfy the Markov property.

In the next section we validate our proposed method for constructing states by investigating whether it is robust to the introduction of non-Markov distortions in the data.

Chapter 7

Validation

We start by describing our validation approach. The first set of results investigate the robustness of our method to non-Markov distortions in the state space used to generate the data. We then investigate the impact of errors in the estimated (conditional) probabilities. Finally, we evaluate the proposed approach using model-free field data.

7.1 Validation Approach

We design a validation environment in which any error in the design of the state space is solely attributable to non-Markov distortions that we control. We start by describing a state space that does not have any non-Markov distortions. We then introduce two types of distortions; distortions in either the current period rewards, or distortions in the transition probabilities.

We return to the simulated dataset that we discussed in Section 3. Consistent with the discretization of the outcomes in each time period, we consider an MDP with five states and two actions: $s = \{1, 2, 3, 4, 5\}$ and $a = \{0, 1\}$. The reward $R^M : S \times A \to R$, transition probabilities $P^M : S \times A \to S$ and policy $\pi^M : S \to A$ to generate the MDP are all constructed from the actual experimental data. Details are discussed in Footnote 4.

The non-Markov deviations in the transition probabilities are constructed as follows:

- With probability α, the state at time period t is determined by the state at time period $t-2 : P^M(s_t|s_{t-2}, a_{t-1})$; and

- With probability $1-\alpha$, the state at time period t is determined by the state at time period $t-1 : P^M(s_t|s_{t-1}, a_{t-1})$.

With probability $1-\alpha$ the transition probabilities only depend upon the state in period $t-1$ and so the states $s = \{1, 2, 3, 4, 5\}$ satisfy the Markov property. However, with probability α the transition probabilities depend upon the state in period $t-2$ and so the states $s = \{1, 2, 3, 4, 5\}$ do not satisfy the Markov property.

To introduce non-Markov deviations in the rewards we use a similar approach:

- With probability α, the reward at time period t is determined by the state at time period $t-1 : (R^M_{t-1}(s_{t-1}, 1) + R^M_{t-1}(s_{t-1}, 0))/2$; and

- With probability $1-\alpha$, the state at time period t is determined by the state at time period $t : R^M_t(s_t, a_t)$.

We present results using data generated under different values of α. In particularly, we allow to vary from 0 to 0.9 in increments of 0.1, and separately evaluate deviations in the rewards and the transition probabilities. For each value of , we simulate a dataset with 200,000 customers and 8 time periods.

We compare two state space designs: **Benchmark** the five states in the MDP: $s = \{1, 2, 3, 4, 5\}$; **Our Method** states generated using the method proposed in this paper. Because the Benchmark state space uses the five states in the data-generating MDP, we can isolate the effects of non-Markov deviations on the value function estimates. If we use other methods to generate states, we cannot isolate whether errors are due to the non-Markov distortions, or from other failings in the state space design. We expect that the performance of the five Benchmark states should worsen as the parameter increases due to the introduction of the non-Markov distortions. In contrast, the states generated by the method proposed in this paper should be robust to non-Markov distortions, and so the results should be stable when α increases.

7.2 Validation Results Using True Probabilities

We first present results using the true probabilities used to generate the simulated data. We also initially focus on the "current policy" reflected in the data (recall that this policy was randomized). We will later also investigate how well the state space designs perform when they are used to design "optimal" policies.

In Figure A.3 we report the error rates of the estimated value functions. For both state space designs, we weight the value function estimates for each state by the number of visits to each state in the data. Our error measure is the absolute difference in the average performance compared to the true value function estimate. In the datasets with deviations, the true value function estimate is calculated using a state space with 25 states to account for whether the transitions (or rewards) depend upon period $t-1(t)$ or period $t-2(t-1)$ (this is a 1-history state space). Standard deviations are estimated using the method proposed in Mannor et al., (2007). The shaded areas in the figure identify 95% confidence intervals.

In Figure A.3 we see that, without any deviations, both state spaces perform well and yield value function estimates for the current policy that are close to the true values. However, with the introduction of distortions in the transition probabilities, there is an immediate and large deterioration in the accuracy of the benchmark state space. The reduction in accuracy is dramatic even for $\alpha = 0.10$. This deterioration is completely attributable to the non-Markov distortions, and highlights the cost of breaching the Markov assumption in the design of state spaces. The performance continues to deteriorate as the size of the distortions increases beyond $\alpha = 0.10$. In contrast, while there is some volatility, the overall performance of the proposed method is relatively robust to the non-Markov dynamics in the data.

We observe a similar pattern when we investigate deviations in the rewards. However, an important difference is that the performance of the benchmark method decreases linearly with increases in the non-Markov distortions (measured by α). The reason for this is that the reward matrix enters the value function estimates in linear form. When we vary α, there is α probability of getting errors in the reward matrix. As a result, the accuracy of the Benchmark state space decreases linearly.

In Figure A.3 we focused on the policy represented in the data. We can also investigate the performance of the "optimal" policies designed using each state space. These results are reported in Figure A.4. An additional issue arises when comparing value function estimates for an optimized policy. The choice of the optimal action in the Policy Improvement step tends to favor positive errors. This issue is discussed in detail in Mannor et.al, (2007). We use the cross-validation approach proposed in Mannor et.al, (2007) to de-bias the estimates of the value function. In particular, we divide the training data into two subsamples: calibration data and validation data. We use the calibration sample to estimate the rewards and transition probabilities and identify the optimal policy. We then evaluate this optimal policy using the validation sample. The independence of the errors between the two samples de-biases the value function estimates. We weight the value function estimates by the frequency of the states in the data. Our performance measure is the average absolute deviation from the true performance of the true optimal policy.

When optimizing the firm's mailing policy, our proposed method is again robust to non-Markov distortions in both transition probabilities and rewards. Notice that the standard deviations are larger in Figure A.3 than in Figure A.4. This is because we only use half of the data (validation data) to evaluate the optimal policy. If we increase the size of the validation data to match Figure A.3, the confidence intervals shrink to a similar size.

We can also use the test proposed in Section 4 to calculate the percentage of states that satisfy the Markov property. Recall that a necessary (though not sufficient) condition for satisfaction of the Markov property is that a state satisfies the following conditional independence test: for all observations in the state, the probability of o_{t+1} conditional on a_t is independent of h_t. The proportion of states that satisfy this test under the different values of α are reported in Figure A.5.[1] We compare our proposed method with states generated using the benchmark state space and a traditional RFM approach.

Under all of the values of α, essentially all of the states constructed using our proposed method satisfy the Markov property. In contrast, almost none of the states constructed using the benchmark method satisfy the Markov property once we introduce non-Markov distortions.

The RFM state space also yields many states that are not Markov, for all values of α. They confirm that the accuracy of the value function estimates produced with the RFM states is also relatively poor (for all values of α). These two results are consistent, and highlight the importance of constructing Markov states.

7.3 Effect of Estimated Probabilities

In the previous subsection, we evaluated our proposed method using the true probabilities that we used to generate the simulated data. This is equivalent to estimating the results using a dataset with a very large (infinite) number of observations. In practice, our sample sizes are often small enough to introduce estimation errors in the (conditional) probabilities. In this subsection, we explore the effect of these estimation errors on the performance of the proposed method.

We start by investigating the variation in probability estimates calculated from our field experiment data. In particular, we use the sampling method described in Section 5 to estimate the conditional probabilities in the system dynamic matrix 50 times. For each entry, we derive the average, minimum and maximum value across these 50 samples. We use the absolute difference between the average and maximum (minimum) as the positive (negative) estimation errors for each entry. The average (weighted) positive estimation error is approximately 38%, while the average negative error is approximately 4%. We use these two numbers to guide the magnitude of the errors we will introduce to the true probabilities in our simulated data.

[1]To minimize finite sample errors, we restrict attention to histories of length 2.

To introduce errors, we allow a proportion of entries in the system dynamic matrix to deviate from their true probabilities. We vary the proportion of deviated entries from 10% to 50%, in 10% increments: $\tau \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. For each deviation, there is a 50% probability of a positive deviation (37%), and a 50% probability of a negative deviation (-1%). Recall that these are the largest observed deviations in our sample of 50 draws from the field data, and so these represent relatively large distortions.[2]

Recall that our proposed approach uses the conditional probabilities in two stages when designing the state space. It first uses them to design the core tests, and then after identifying the core tests, it uses them to group customers into states (according to the conditional probabilities associated with these core tests). To decompose how errors in these probabilities affect the performance of the method, we separately introduce the distortions just when designing the core tests, just when grouping customers into states, or in both stages. We report the findings in Table A.2.

We obtain several insights from Table A.2. First, our core-test finding algorithm proposed in Section 5 is robust to errors in the conditional predicted probabilities. This can be seen from the stability of the errors when varying the estimation errors in designing core tests. Second, when we introduce estimation errors when either designing core tests or grouping customers, the performance of the proposed method does not dramatically worsen. However, when we introduce estimation errors in both stages, the estimation errors in each stage reinforce each other. The reason is that our algorithm finds more core tests than the true set; recall that estimation errors in the conditional probabilities can make linearly independent columns appear dependent. When we identify additional core tests, there are more conditional probabilities to estimate. If these probabilities are estimated with error, this amplifies the errors in the system. As a result, the combination of too many core tests and errors in the probabilities used to group customers undermines the performance of the method.

These findings suggest that the performance of the proposed method depends

[2] We also provide an additional way to introduce errors in probabilities where we vary the magnitude of the errors. Similar insights can be got.

upon having sufficient data to reduce errors in the estimation of the conditional probabilities. We investigate this issue next using the raw data from the field experiments.

7.4 Model-Free Evidence

In this final sub-section, we step away from the simulated data, and use the field experiment data to provide model-free evidence of the performance of the method. We use the following four-step procedure:

- We split the 136,262 customers into two sub-samples. We use the first sub-sample as training data and the second sub-sample as validation data.

- We use the training data to implement our proposed algorithm; identify core tests, group customers using probabilities on these core tests, calculate rewards and transitions for each state-action pair.

- For each customer in the validation data, we randomly identify a length-two observation,[3] and calculate the actual observed profit for each customer in this length-two observation.[4]

- We compare the actual length-two outcome with the length-two outcome predicted under our proposed approach (under the current policy), and calculate the average absolute error as a percentage of the actual value.

We report the findings in Table A.3.

We report five sets of results according to how many observations we include in the training data. We allow the size of the training data to vary from 20,000 to 100,000 customers in increments of 20,000. However, for all five replications we hold the size of the validation data fixed at 36,262 customers. As a benchmark we also report the error of a state space designed using RFM measures. In this benchmark, we discretize

[3]We also provide a comparison based on length-three observation.

[4]To reduce errors from discretization of the outcomes, we use the average profit in each profit bucket.

the RFM variables using the same approach that we described in Section 3. We train the RFM model using the same training data that we use for our proposed approach.

There are two findings of interest. First, with sufficient training data, our method performs very well on the field data. When the number of observations in the training data is at least 80,000, the error rate is less than 10%. Second, the performance of the RFM method is unreliable. Even with large amounts of data, the performance of the RFM states does not converge to accurate outcomes. Moreover, the performance improvement is not consistent as we increase the size of the training sample. This again highlights the importance of constructing states that satisfy the Markov property.

Chapter 8

Conclusions and Limitations

In this paper, we study the problem of optimizing a sequence of marketing actions. We highlight the importance of constructing states that satisfy the Markov property when using traditional Reinforcement Learning methods, and propose a method for designing states that satisfy this requirement. Our findings can be used with traditional Reinforcement Learning methods, or with related methods such as POMDPs and Approximate Dynamic Programming techniques.

The method begins with the insight that states that satisfy the Markov property contain the same information about the probability of future outcomes as the raw training data. By constructing predictions over how customers will respond to a firm's marketing actions, and grouping customers together if their predicted behavior is similar, we can guarantee that the resulting states will be Markov. There is no obvious way to obtain the same guarantee if we instead group customers directly according to the similarity of their past transactions (rather than grouping them based on the probabilities of future events).

While this approach is theoretically appealing, a practical challenge is that there is an infinite combination of future customer behaviors (future events) to predict. One of our contributions is to recognize that transactions by individual customers are often relatively sparse. This can greatly reduce the range of future events that we need to consider, and considerably lower the computational burden. We turn a problem that is often impractical, into a problem that is relatively straightforward to solve in many marketing settings.

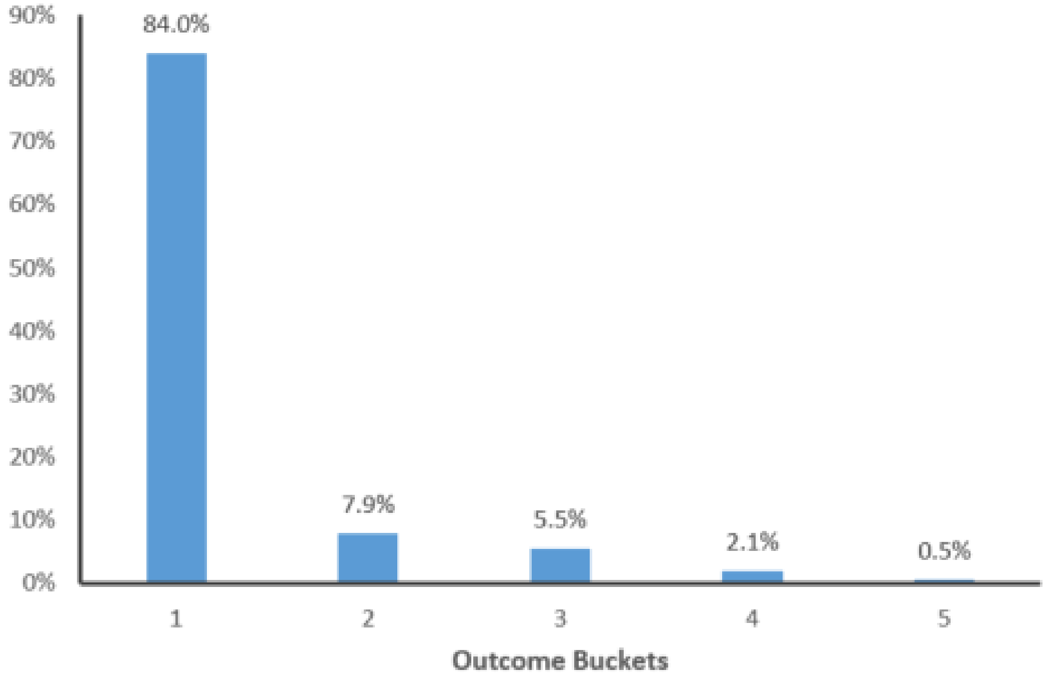
We validate the proposed method by showing that it is robust to non-Markov distortions in the data-generating process. In contrast, state spaces that do not adjust for these distortions suffer immediate performance losses, even when the distortions are small.

The main limitation of our current method is scalability. As the number of observed time periods increases, the length of the core tests grows, and the algorithm we use for finding core tests faces increasing computational challenges. We see this as an important direction for future research. There are two directions that might help to improve the scalability of the algorithm. First, we can try to exploit methods in computer science designed to handle high-dimensional data. Deep learning, transfer learning, and spectral learning, are all methods that can be considered. Second, we can use more domain knowledge to simplify the problem. Currently, we exploit the sparsity in the customer transaction data. However, it is likely that there are other ways to further improve the use of sparsity in the algorithm.

Appendix A

Figures

Figure A-1: Histogram of Customer Outcomes in the Six Experiments



Notes. The figure reports a histogram of the outcomes from the six experiments (the columns add to 100%). The outcomes are grouped into five discrete buckets and we report the proportion of observations in each bucket. The unit of analysis is a customer in an experiment and the sample sizes are reported in Table 1.

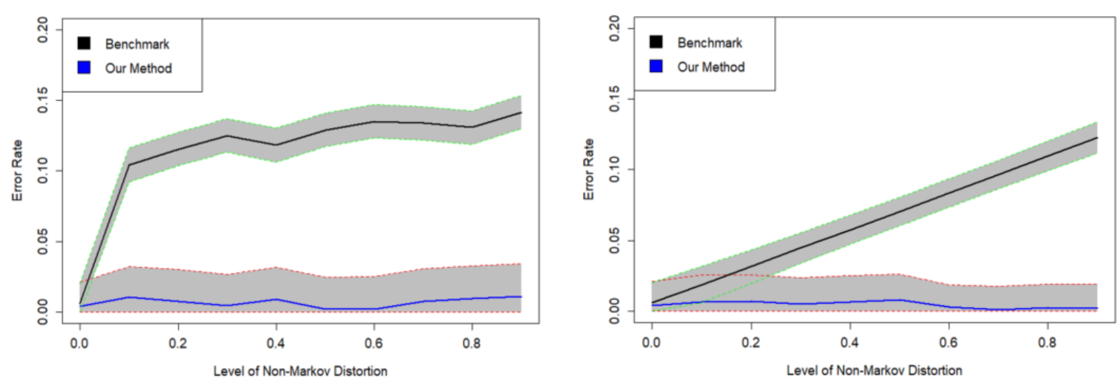
Figure A-2: System Dynamic Matrix

	q_1	q_2	q_3	q_4	q_5	$\cdots$
$\emptyset$	$p(q_1\|\emptyset)$	$p(q_2\|\emptyset)$	$p(q_3\|\emptyset)$	$p(q_4\|\emptyset)$	$p(q_5\|\emptyset)$	
h_1	$p(q_1\|h_1)$	$p(q_2\|h_1)$	$p(q_3\|h_1)$	$p(q_4\|h_1)$	$p(q_5\|h_1)$	
$\vdots$						
h_4	$p(q_1\|h_4)$	$p(q_2\|h_4)$	$p(q_3\|h_4)$	$p(q_4\|h_4)$	$p(q_5\|h_4)$	
h_5	$p(q_1\|h_5)$	$p(q_2\|h_5)$	$p(q_3\|h_5)$	$p(q_4\|h_5)$	$p(q_5\|h_5)$	
$\vdots$						$\cdots$

Notes. Tests and histories are arranged in order of increasing length and, within the same length, in increasing categorical indicator order. The length one tests and histories are listed in the main text: q_1 = "00", q_2 = "01", q_3 = "10", q_4 = "11". The length two tests and histories start with q_5 = "0000" $= \{a^1 = Not\ mail, o^1 = Not\ buy, a^2 = Not\ mail, o^2 = Not\ buy\}$. The other length two tests and histories are listed in the Appendix. $\emptyset$ represents zero-length history, $h_1 = q_1$, $h_4 = q_4$ and $h_5 = q_5$.

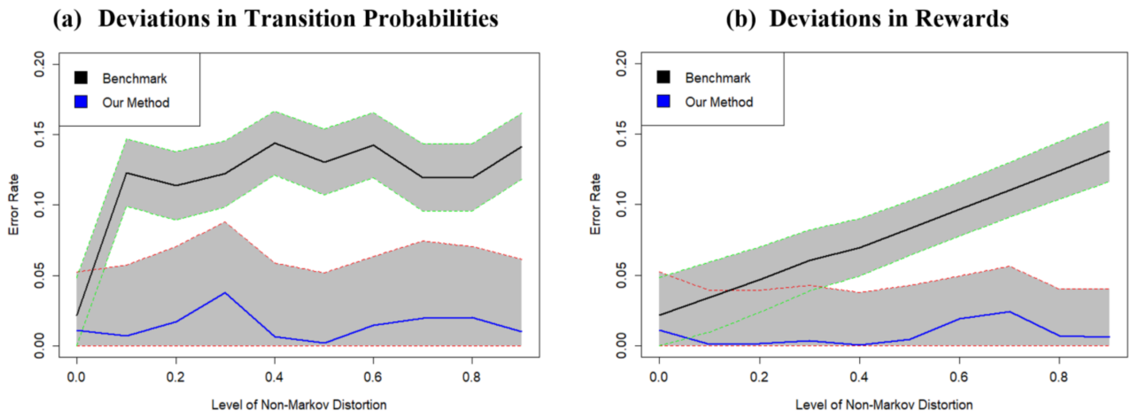
Figure A-3: Error Rates Under the Current Policy Using the True Probabilities

(a) Deviations in Transition Probabilities **(b) Deviations in Rewards**



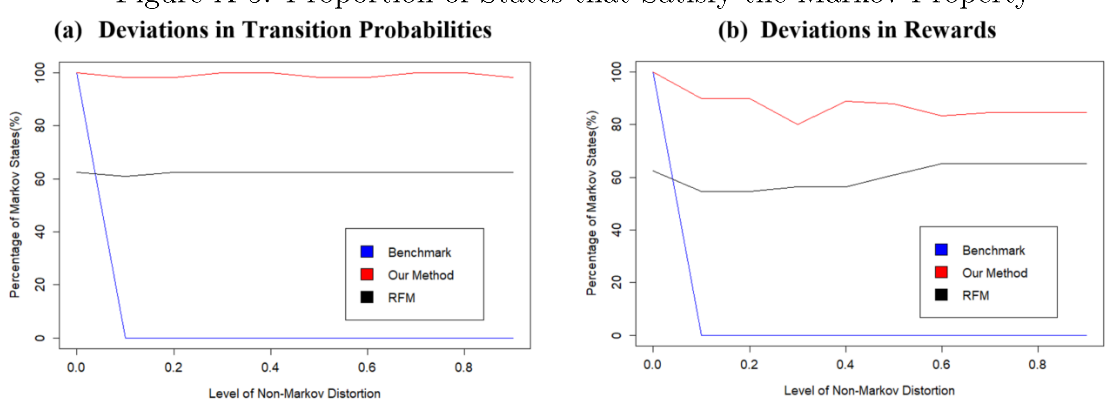
Notes. The figures report the error rates of value function estimates under the current policy for both the benchmark state spaces and the PSR states produced by our proposed method using the true probabilities $p(q|h)$. In panel (a) the x-axis is the level of non-Markov distortion in the transition probabilities (measured by α). In panel (b) the x-axis is the level of distortion in the rewards. The performance measure is the average absolute deviation from the true performance of the current policy. Additional details are provided in the Appendix.

Figure A-4: Error Rates Under the Optimal Policy Using the True Probabilities



Notes. The figures report the error rates of value function estimates under the optimal policy for both the benchmark state spaces and the PSR states produced by our proposed method using the true probabilities $p(q|h)$. In panel (a) the x-axis is the level of non-Markov distortion in the transition probabilities (measured by α). In panel (b) the x-axis is the level of distortion in the rewards. The performance measure is the average absolute deviation from the true performance of the optimal policy. Additional details are provided in the Appendix.

Figure A-5: Proportion of States that Satisfy the Markov Property



Notes. The figures report the proportion of states that satisfy the Markov property for the benchmark state space, the RFM state and the PSR states generated by our proposed method using true probabilities. In panel (a) the x-axis is the level of non-Markov distortion in the transition probabilities (measured by α). In panel (b) the x-axis is the level of distortion in the rewards. Additional details are provided in the Appendix.

Appendix B

Tables

Table B.1: Sample Sizes in Six Experiments

Experiment	1	2	3	4	5	6
Treatment	68273	68176	67797	68145	85609	69998
Control	67989	68086	68465	68117	50653	66264

Table B.2: Effect of Estimated Probabilities

Proportion	Current Policy (%)			Optimal Policy (%)		
(τ)	Core Tests	Grouping	Both	Core Tests	Grouping	Both
0.1	3.19	9.10	14.58	2.49	0.47	16.18
0.2	3.19	4.83	37.05	2.49	0.77	27.14
0.3	3.19	8.50	54.41	2.49	15.34	45.87
0.4	3.19	5.47	61.57	2.49	1.28	50.27
0.5	3.19	2.09	64.92	2.49	7.89	59.89

Table B.3: Model-Free Evidence: Error Rates of Length-Two Profits

Number of Customers	Proposed Method (%)	RFM (%)	Difference (%)
20,000	14.09(0.44)	162.76(1.20)	148.67(0.40)
40,000	11.64(0.62)	168.81(1.39)	157.17(0.48)
60,000	10.57(0.39)	167.89(1.39)	157.32(0.46)
80,000	9.13(0.29)	165.02(1.29)	155.89(0.42)
100,000	8.01(0.20)	164.43(1.99)	156.42(0.63)

Appendix C

Proof of Result 2

By deleting all history rows whose corresponding tests have zero probability at zero-length history, we delete history rows with all zero entries because of the impossibility of the existence of such history.

Suppose at iteration step k, a set $\hat{Q}$ of tests are returned, and at iteration step $k+1$, the algorithm stops because of unchanged rank. This means that columns corresponding to the tests in $\hat{Q}$ are a basis for all one-length tests and one-step extension of tests in $\hat{Q}$. This is just from the way we expand the submatrix. Formally speaking, it means that for all history h (including the deleted ones), all tests $\hat{q} \in \hat{Q}$ and all length-one action-observation possibilities ao, there exists parameter vectors $m_{ao\hat{q}}$ and m_{ao} such that

$$p(ao\hat{q}|h) = p(\hat{Q}|h)^T m_{ao\hat{q}}$$

and

$$p(ao|h) = p(\hat{Q}|h)^T m_{ao}$$

Thus, we are able to continuously update the state representation through computing the prediction of any test at any history based on the state-update function provided at the end of Section 4. Thus, tests in $\hat{Q}$ are a basis for all tests' predictions at all histories, and the proof is complete.

Appendix D

Dealing with Estimation Errors in Finding Core Tests

To manage the effect of predicted probabilities' estimation errors in finding core tests, we take a relatively conservative approach to calculate the rank and find linearly independent columns of the submatrices in the system dynamic matrix.

If the removal of a column causes the rank of the resulting matrix to decrease, that column belongs to the independent columns. We use this criterion to define the linearly independent columns, whose corresponding tests are used for one-step expansions.

A separate issue arises in obtaining the rank of the resulting matrix before and after the deletion of each column. Again, we use a relative conservative approach proposed by Golub and Van Loan (2013). The method uses singular value decomposition and uses a singular value cutoff σ_{cutoff}. The estimated rank is the number of singular values above the cutoff σ_{cutoff}. For a $(m \times n)$ matrix A, with $m \geq n$, $\sigma_{cutoff} = \epsilon ||A||_\infty$, where ϵ is the average error in the matrix entries. The term is calculated using Chebyshev's inequality. Chebyshev's inequality delivers a bound on the maximal error. We use this error bound as the average error. In other words, the average error is upper bounded with a given certainty. We choose 0.95 as the certainty measure, which implies that with probability 0.05 the error estimate is worse than ϵ.

References

Anderson, Eric T. and Duncan I. Simester (2004), “Long-Run Effects of Promotion Depth on New Versus Established Customers: Three Field Studies,” Marketing Science, 23(1), 4-20.

Bertsekas, Dimitri P. (2017), Dynamic Programming and Optimal Control, 4th Edition, Athena Scientific, Belmont, MA, USA.

Bitran, Gabriel R. and Susana V. Mondschein (1996), “Mailing Decisions in the Catalog Sales Industry,” Marketing Science, 42(9), 1364-1381.

Bult, Jan Roelf and Tom Wansbeek (1995), “Optimal Selection for Direct Mail,” Marketing Science, 14(4), 378-394.

Dubé, Jean-Pierre and Sanjog Misra (2019), “Personalized Pricing and Customer Welfare,” working paper, University of Chicago.

Golub, Gene H. and Charles F. Van Loan (2013), Matrix Computations, 3rd edition, The Johns Hopkins University Press, Baltimore, Maryland, USA.

Gönül, Fusun and Mengze Shi (1998), “Optimal Mailing of Catalogs: A New Methodology Using Estimable Structural Dynamic Programming Models,” Management Science, 44(9), 1249-1262.

Hauser, John R., Glen L. Urban, Guilherme Liberali, Michael Braun (2009), “Website Morphing,” Marketing Science, 28(2), 202-223.

Hitsch, Gunter J. and Sanjog Misra (2018), “Heterogeneous Treatment Effects and Optimal Targeting Policy,” working paper, University of Chicago.

James, Michael R. and Satinder Singh (2004), “Learning and Discovery of Predictive State Representation in Dynamical Systems with Reset,” Proceedings of the 21st International Conference on Machine Learning, 417-424.

Jedidi, Kamel, Carl F. Mela and Sunil Gupta (1999), "Managing Advertising and Promotion for Long-run Profitability," Marketing Science, 18(1), 1-22.

Johnson Garrett A., Randall A. Lewis, and Elmar I. Nubbemeyer (2017), "Ghost Ads: Improving the Economics of Measuring Online Ad Effectiveness," Journal of Marketing Research, 54(6), 867–884.

Johnson Garrett A., Randall A. Lewis, and David H. Reiley (2016), "When Less Is More: Data and Power in Advertising Experiments," Marketing Science, 36(1), 43–53.

Khan, Romana, Michael Lewis and Vishal Singh (2009), "Dynamic Customer Management and the Value of One-to-One Marketing," Marketing Science, 28(6), 1063-1079.

Littman, Michael L., Richard S. Sutton and Satinder Singh (2002), "Predictive Representations of State," Proceedings of the 14th advances in Neural Information Processing Systems, 1555-1561.

Lovejoy, William S. (1991), "A Survey of Algorithmic Methods for Partially Observable Markov Decision Processes," Annals of Operations Research, 28, 47-65.

Mannor, Shie, Duncan Simester, Peng Sun and John N. Tsitsiklis (2007), "Bias and Variance Approximation in Value Function Estimates," Management Science, 53(2), 308-322.

Mela, Carl F., Sunil Gupta, and Donald R. Lehmann (1997), "The Long-Term Impact of Promotion and Advertising on Consumer Brand Choice," Journal of Marketing Research, 34(2), 248-261.

Ostrovsky, Michael and Michael Schwarz (2011), "Reserve Prices in Internet Advertising Auctions: A Field Experiment," Proc. 12th ACM Conf. Electronic Commerce (ACM, New York), 59–60.

Rafieian, Omid and Hema Yoganarasimhan (2019), "Targeting and Privacy in Mobile Advertising," working paper, University of Washington.

Rust, John (1994), "Structural Estimation of Markov Decision Processes," Robert F. Engle, Daniel McFadden, eds. Handbook of Econometrics, vol. 4, Elsevier Sciences, The Netherlands, 3081-3143.

Simester, Duncan I., Peng Sun and John N. Tsitsiklis (2006), "Dynamic Catalog Mailing Policies," Management Science, 52(5), 683-696.

Simester, Duncan I., Artem Timoshenko and Spyros I. Zoumpoulis (2019A), "Targeting Prospective Customers: Robustness of Machine Learning Methods to Typical Data Challenges," Management Science, forthcoming.

Simester, Duncan I., Artem Timoshenko and Spyros I. Zoumpoulis (2019B), "Efficiently Evaluating Targeting Policies: Improving Upon Champion vs. Challenger Experiments," Management Science, forthcoming.

Singh, Satinder P., Michael R. James and Matthew R. Rudary (2004), "Predictive State Representations: A New Theory for Modeling Dynamical Systems," Proceedings of the 20th advances in Uncertainty in Artificial Intelligence, 512-519.

Sutton, Richard S. and Andrew G. Barto (2018), Reinforcement Learning: An Introduction, 2nd edition, MIT Press, Cambridge, MA, USA.

Wingate, David (2012), "Predictively Defined Representations of State," Marco Wiering, Martijn van Otterlo, eds. Reinforcement Learning: State-of-the-Art, Springer, Berlin, Germany, 415-437.

Wingate, David and Satinder Singh (2008), "Efficiently Learning Linear-Linear Exponential Family Predictive Representations of State," Proceedings of the 25th International Conference on Machine Learning, 1176-1183.

Zhang, Jonathan Z., Oded Netzer and Asim Ansari (2014), "Dynamic Targeted Pricing in B2B Relationships," Marketing Science, 33(3), 317-337.