Agentic Marketing: Designing Control Models using Causal Inference

Brandon Scott

***Abstract*—We present the efficacy of control models in optimizing marketing messaging in the context of out-of-app (OOA) messaging. To do this, we develop a deep Q network (DQN) to model the customer journey. We train the agent via a reward system that utilizes outputs from a Bayesian media mix model (MMM) and a causal uplift model. We demonstrate that utilizing cause-effect relationships in agent reward functions allows the agent to learn targeted marketing interventions. We further demonstrate this capability in the context of a bakery that releases a mobile application to increase customer loyalty and subsequently increase purchase behavior.**

***Index Terms*—Control, Marketing, Optimization, RL, Causal Inference**

I. Introduction

Within the business world, there are distinct departments that perform specific operations for a given company. The marketing department of a company is generally in charge of positioning the company through marketing activities to appeal to target customers to increase company profits. These marketing activities can be siloed into four categories: product, place, price, and promotion. These four categories are commonly known as the "4 P's of marketing" as they encompass the key responsibilities marketing has to make the company successful. Essentially, a marketing department of a company is tasked with optimizing customer experience such that a given targeted customer is satisfied with the company's product, knows where the product is located, is willing to pay the price, and is reasonably receptive to the promotion of said company/product.

In order to achieve this optimal customer experience, many marketing departments have adopted data tracking and analytical methods to verify if their company's strategy is generating success [1]. Thus, the world of marketing has been a great area of research and application for real-world applications of methods within supervised and unsupervised learning [2]. Marketers are constantly trying to identify what to send and to whom will they send marketing campaigns. This is known as targeted campaign marketing. With the abundance of data now available to marketing departments and the continual evolution of optimization techniques, marketing departments can optimally send messaging to customers at the right time using the right message to achieve their business goals.

One of the areas of exciting research for marketing departments is how to make decisions under uncertainty. In other domains such as engineering, these are commonly known as control models, where a system is managed by a controller that is commonly used to adjust the system to operate optimally. In business settings such as marketing, a control model can be viewed in a similar way because there is a process or system that should be controlled or optimized in a way that produces acceptable output. For businesses that utilize mobile applications as a channel of commerce, one effective method to increase specific business metrics (e.g., engagement) is to utilize out-of-app (OOA) messaging. Out-of-app messaging refers to any communication with users of the app that occurs outside the native app platform (e.g. push notifications, emails, texts, etc.).

As mentioned previously, marketers need to make important decisions such as which customers or non-customers to target, the type of messaging to target with, and which marketing channel to utilize. Understandably, the number of combinations grows exponentially as the number of channels, customer information, and campaign messages increases. Additionally, customer behavior is anything but static. Previously collected data could be irrelevant in a matter of weeks as customer behavior evolves. Therefore, a system to handle this kind of operation not only needs to efficiently identify optimal messaging avenues and times, but also needs to be regularly updated via the feedback that consumers explicitly and implicitly share.

In this paper, we propose an innovative system that utilizes control methodologies. Specifically, we model this problem as a control problem, where the system to be optimized is OOA messaging. To control the system, we design a controller to understand system behavior using media mix models and utilize reinforcement learning to find optimal decisions to feed to the system. We demonstrate this system using a toy example of a fictitious baker to illustrate the robustness of the system.

II. Related Work

A. Background on Reinforcement Learning

Reinforcement learning (RL) aims to find an optimal policy (strategy) for an autonomous agent to maximize some reward over time. In general, reinforcement learning can be categorized into two groups: model-based and model-free reinforcement learning [4]. Model-based reinforcement learning deals with an agent making decisions based on an inferred model of the environment. Model-free reinforcement learning deals with an agent that interacts with an environment and optimizes from those actions.

Model-based RL can be traced back to the birth of dynamic programming with Richard Bellman [3]. Bellman proposed the theory that we can find the optimal solution to a big problem by finding the optimal solutions to local/smaller problems and aggregating these solutions back to the original problem. Thus,

in the context of RL, the optimal decision-making strategy for an agent can be broken down into smaller decisions, then aggregated back to the original policy decision. Specifically, Bellman derived the following equation.

$$V^*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V^*(s')] \tag{1}$$

(1) states that we can find the optimal value of our current state by finding the maximum expected return over all possible actions to next states and their respective rewards. Thus, an agent in a model-based RL framework uses states, actions, and rewards to model their optimal path. The most common framework for this is a Markov Decision Process (MDP) [5].

A MDP is framework for decision making that incorporates stochasticity into the process. A MDP consists of states that are connected by actions, each with defined transition probabilities, as shown in Figure 1. Thus, as shown so far with model-based RL, one must know the states, actions, and transition probabilities in order to have a valid model. This allows the agent to plan ahead once the model of the system is discovered, whereby the agent can maximize rewards by following the optimal policy inferred through by the model.

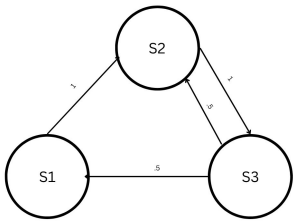

Fig. 1. Example Markov Decision Process

There are many cases where this is not feasible. One possibility could be that the state space is just too large to compute all value functions for all possible states. Another possibility is that we don't know the underlying dynamics of the environment, making it hard or even impossible to model the transition probabilities or reward structure of the environment. This is where we can extend the framework of MDPs from a model-based sequential decision process to a model-free decision process.

Model-free RL algorithms still attempt to learn an optimal policy to maximize rewards in their respective environment, but what makes them different from model-based algorithms is that they don't attempt to model the transition probabilities nor the reward functions. Thus, the difference in lies in the implemented strategy. Model-based RL attempts to learn the underlying model of the environment to maximize rewards, whereas model-free RL enables the agent to learn directly from experiencing the environment.

One of the biggest advancements in model-free RL came with *Temporal Difference Learning* (commonly known as TD learning) [6]. TD learning optimizes the value functions for a given state (as is the goal in model-based learning) but does not attempt to model the transition probabilities nor the reward function. Instead, TD learning attempts to optimize the value function by experimenting with the environment and seeing if there is an increase or decrease in the value function. Additionally, the hyperparameter α adjusts the step size of each update, which determines how granular the system will learn, as shown in (2).

$$V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{2}$$

TD learning heavily influenced one of the most popular RL algorithms, *Q-learning* [6]. Q-learning uses TD learning in order to optimize the quality function, $Q(s, a)$. The quality function is a way of viewing the "quality" of the state (the expected rewards) via a specific action. By using TD learning, we can optimize the quality function in the same way that we optimized the value function. (3) shows the full formula.

$$Q(s, a) = Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right] \tag{3}$$

Like TD learning, Q-learning interacts with the environment and updates each Q function states by iteratively converging to the action that maximizes the rewards.

In summary, model-based methods attempt to learn the underlying model of the environment in order to derive the optimal policy to maximize rewards. Model-free methods attempt to learn the policy that maximizes rewards by directly experimenting with the environment. Model-based methods tend to be more sample efficient, but do not perform as well in complex and large state space environments. In contrast, a model-free method requires more data to properly understand the environment, but can handle complex and large state space environments better [7].

B. Bayesian Marketing Mix Models

Marketing departments, like financial portfolios, generally strive to have a diverse pool of channels that they perform promotions. The question in the sense of optimization is which channels give the best return on investment (ROI). To determine this, businesses utilize marketing mix models (MMM) to assess optimal ROI channels [8]. The formal definition of this model is found in (4).

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i \cdot \left(\sum_{t'=0}^{t} \theta_i \lambda_i^{t'} x_{i,t-t'} \right) + \epsilon \tag{4}$$

One of the more popular methods in marketing is to utilize Bayesian MMM [9]. In the Bayesian paradigm, we utilize prior information to inform our models before experimentation takes place. Then, as we proceed through the process of marketing operations, we update our priors based on the data we observed to obtain our posterior. This then becomes our prior and we

continue the process of updating our beliefs based on data we observe.

In the context of optimizing marketing spend, Bayesian MMM essentially aims to understand the incremental impact of changing spend in different activities. Thus, Bayesian MMM utilizes prior information about different channels and other environmental factors to calculate metrics like return on ad spend (ROAS). This allows marketers to find the optimal ad spend areas using this technique.

C. Causal Uplift Models

Although MMM are good at comparing ROI across channels, they do not show why these channels provide good ROI. In order to identify the levers that businesses can pull in order to change ROI, we need to utilize causal models. Causal inference models help us to identify which marketing treatments actually produce cause and effect relationships.

A prominent use of causal inference in marketing is to determine how customers will react to promotions. Usually, a marketing department aims to identify four personas: lost causes, do-not-disturbs, persuadables, and sure things. This method of causal inference is known as uplift modeling [10].

Uplift modeling utilizes causal relationships to help marketing departments better personalize their promotions by essentially segmenting their customers into the four personas mentioned above. By doing this, marketing teams become more informed on what are the actual levers they can pull to affect customer behavior.

III. Implementations

Many companies have utilized control methods in order to optimize operations. For example, Spotify used RL techniques to provide explainable recommendations to users while also providing recommendations that were "optimized" for selection [12]. Additionally, Adobe utilized RL methods to optimize personalized ad recommendations to maximize customer life time value [13].

Specifically, in the world of OOA, Uber developed their own architecture to better personalize their Eats marketplace [14]. Additionally, Duolingo utilized bandit algorithms to increase user engagement via push notifications [15].

These are only a few examples within industry that have utilized control methodologies to optimize an operation. Many other algorithms are proprietary to their respective companies. Needless to say, many business problems can be organized into control problems.

IV. Problem Formulation

We introduce in this section our toy example. We aim to help a thriving baked goods company, "Baked by Brandon." "Baked by Brandon" owns one bakery in a metropolitan area and has seen great success over the past couple of years. To help increase customer loyalty and provide ease of purchase for customers, "Baked by Brandon" has adopted the use of technology by creating a mobile phone app. The app allows customers to order ahead so that they can come pick up their order whenever convenient for them.

"Baked by Brandon" hopes to develop a loyalty program to reward customers based on their purchase behavior. Furthermore, they hope to enhance their marketing efforts by focusing their resources on customers that would best respond to campaigns while also not annoying those who would already purchase. Specifically, because they believe the app can enhance loyalty by providing a new touchpoint from anywhere, they want help optimizing their marketing efforts via out-of-app (OOA) messaging.

In essence, the task at hand is to model the sequential transactions of customers such that we can identify what the next best marketing action would be for each customer. By training a model to identify this, we can better personalize our marketing efforts to utilize campaigns (or not utilize campaigns) on specific customers that would have the best reaction to them.

V. Methodology

To tackle the problem of modeling sequential transactions of customers for "Baked by Brandon," we first frame the problem utilizing control theory. From this perspective, we identify two core components: the system to be optimized/controlled and the controller of the system. The system to be optimized is the OOA messaging system. Our controller needs to be designed in such a way that aligns system performance/behavior with desired business outcomes. In our case, the business outcomes can be boiled down to three key strategies: increase purchase behavior of customers, convert customers into higher loyalty tiers (or retain them in the highest tier), and not cause annoyance that would drive customers to lower tiers (decrease purchase behavior).

A. Data Generation

Since real world data of this scale and nature are incredibly difficult to find, we used Python to generate synthetic data for these modeling purposes. We generated three separate datasets for different purposes of the modeling process. First, we generate synthetic data showing the effects of ad spend via our three marketing channels (push notifications, text, and email). This data is used to model the media mix of our company to help identify which channel is most "effective" (i.e. has the highest ROI).

Second, we generate synthetic data showing the causal effects of marketing treatments on customers. These data are used to model the uplift of customers to help identify how different demographics of customers react to marketing treatments.

Third, we generate synthetic data showing the sequential transactions of customers. This data utilizes logic from the previous two models to show how customer behavior changed due to marketing interventions and will be used to train a deep q network (DQN) model. The code for these data generation functions and respective models can be found here.

B. Controller Design

We aim to design a controller that utilizes important output information from our system. Specifically, we strive to utilize causal methodologies to train our agent to perform actions that have true cause-effect relationships. Identifying causal relationships is important in business to identify what levers we can pull to change customer behavior. Therefore, we propose designing a prototype causal controller that utilizes information from our Bayesian media mix model and causal uplift model. The output of these models inform the reward design of our DQN model so that it better aligns its actions with causal inference rather than correlation-based inference.

C. MMM and Uplift Model

We utilized a Bayesian Media Mix Model to identify marketing channel ROI. In our model, we incorporated both saturation and carryover effects by modeling saturation with a Hill function and carryover with geometric decay. Priors for a and b in the saturation function were $Gamma(3, 2)$ and $Beta(2, 2)$. Our β prior was $Normal(5, 3)$ with σ prior being $HalfNormal(1)$. For the carryover function, we used a prior of $Beta(2, 2)$. The model was implemented using numpyro in Python.

For our uplift model, we used a T regressor model to estimate the causal impact of our treatments. We used XGBoost as the underlying predictive model in our T regressor. We implemented the T regressor using the causalml package in Python along with the xgboost library.

D. DQN Model

We implemented a DQN to implement our next-best action model. We set up the model using PyTorch by inheriting the nn.Module class to define our DQN. We used three fully connected layers with input layer having the size of our state space, followed by an intermediate layer with 24 neurons, followed by the output layer with size of our action space.

We then defined our agent class. The agent has methods of remember, act, and replay. Our agent also contains the hyperparameters of γ = .95, ϵ = 1, ϵ decay = .995, and $min(\epsilon)$ = .01. To train our agent, we use a batch size of 50 with 10 episodes.

The reward function is designed with the ultimate goal of driving customers to higher loyalty tiers. In order for customers to transition to higher tiers, they need to make a purchase. So, the long-term strategy of the agent is to maintain high levels of loyalty (i.e., retain gold-tier customers) while continuing to drive lower-tier customers to make a purchase such that they have a non-zero probability to transition to a higher loyalty tier. To do this, the base reward for the agent is the ticket size produced by a customer. Additionally, if the agent provoked a purchase by a customer via a marketing action, the additional reward is the uplift provided by the purchase. Furthermore, if the customer in the next state is at a higher tier than previously, the agent receives a large reward. If the customer is already a gold-tier customer and the agent retains the gold tier, the agent receives a large reward. If the customer does not make a purchase and subsequently drops in tier, the agent receives a negative reward.

VI. Results

Due to the synthetic nature of our data, the main purpose of our modeling was to see the effects of causal model outputs in reward functions for agents. It suffices to say that our Bayesian MMM and Causal uplift model identified correctly the underlying functions we used to create the data. Therefore, we focus this section on the output/results of our DQN model.

We used a dataset that consisted of 500 customers each with 50 steps of time, totaling in 25000 observations, with 40% no tier, 30% bronze tier, 20% silver tier, and 10% gold tier. There were 8 state attributes (age, gender, distance to store, time step, and booleans for each loyalty tier) and 10 actions (none, ad-push, small discount-push, large discount-push, and the same combinations for text and email channels). We used a batch size of 50 with 10 episodes of training.

After training our agent, we designed a function that simulated the agent being deployed to production where each customer time step would be influenced by the agent. Our test data included 500 users each with 1 time step. We ran the model such that each user generated 50 steps with agent influence. Based on our results, 99% of users ended up as gold tier customers, demonstrating that our agent discovered an optimal policy utilizing our causal logic.

The next steps to improve our model begin with the synthetic data logic. Currently, the transition probability between tiers is very simple and could be enhanced with different time-sensitive and over-exposure features. Additionally, our model was trained on a relatively small dataset for the task it is required to do. Increasing the dataset so that when the agent is faced with more complex logic, it can generalize better. Furthermore, we did not perform any hyperparameter tuning for the agent so ensuring we find optimal training hyperparameters for the agent would be beneficial. Finally, if we were to release this agent to production use, we would need to determine the proper feedback update loop. Due to the static nature of the data, we don't have any timeline on when to update our causal models and subsequently update our reward logic. To create a robust system, we would need to identify the best update logic for our feedback loop in order to keep the agent up to date with current customer behavior without overburdening our systems with too many updates.

Acknowledgment

We'd like to acknowledge the class CS 513 for motivating us to create this system.

References

[1] A. Krasnikov and S. Jayachandran, "Implementing Big Data Analytics in Marketing Departments: Mixing Organic and Administered Approaches to Increase Data-Driven Decision Making," *Informatics*, vol. 8, no. 4, p. 66, 2021. doi: 10.3390/informatics8040066..

[2] L. Bernardi, T. Mavridis, and P. Estevez, "150 Successful Machine Learning Models: 6 Lessons Learned at Booking.com," in *Proc. 25th ACM SIGKDD Conf. Knowl. Discov. Data Min. (KDD '19)*, Anchorage, AK, USA, 2019, pp. 1-9. doi: 10.1145/3292500.3330744.

[3] R. Bellman, ”On the Theory of Dynamic Programming,” RAND Corporation Paper P-550, Santa Monica, CA, USA, 1954.
[4] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019, ch. 11.
[5] R. Bellman, ”A Markovian decision process,” *J. Oper. Res. Soc. Am.*, vol. 3, no. 1, pp. 81-87, Feb. 1955.
[6] R. S. Sutton, ”Learning to Predict by the Methods of Temporal Differences,” *in Proc. Fourth Int. Workshop Mach. Learn.*, Irvine, CA, USA, 1987, pp. 9-13.
[7] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, May 1992.
[8] S. C. Graves, A. H. G. Rinnooy Kan, and P. Zipkin, “Marketing Mix Models” in *Handbook in Operations Research and Management Science*, vol. 4, S. C. Graves, A. H. G. Rinnooy Kan, and P. Zipkin, Eds. Amsterdam: North-Holland, 1993, ch. 15, pp. 665–740.
[9] Y. Jin, W. Yueqing, et al., ”Bayesian Methods for Media Mix Modeling with Carryover and Shape Effects,” Google Research Publications, 2017.
[10] P. Gutierrez and J.-Y. Gerardy, ”Causal Inference and Uplift Modeling: A Review of the Literature,” [Online]. Available: https://proceedings.mlr.press/v67/gutierrez17a/gutierrez17a.pdf.
[11] R. Clavera, A. Nagabandi, S. Levine, and C. Finn, “Learning to adapt in dynamic, real-world environments through meta-reinforcement learning,” arXiv preprint arXiv:2201.05433, 1 2022.
[12] Y. Chen, H. Benson, R. Chen, R. Guo, and H. Liu, ”Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits,” in *Proc. ACM Int. Conf. Inf. Knowl. Manag. (CIKM ’19)*, Beijing, China, Nov. 2019, pp. 1041-1050.
[13] Y. Liu, S. Li, Y. Zhang, and J. Tang, ”Personalized Ad Recommendation Systems for Life-Time Value Optimization with Guarantees,” in *Proc. ACM Int. Conf. Inf. Knowl. Manag. (CIKM ’20)*, Virtual Event, Ireland, Oct. 2020, pp. 245-254.
[14] V. Pham, B. Draves, H. Fu, D. Yu, N. Papadakis, A. Prakash, and I. Huang, “Personalized Marketing at Scale: Uber’s Out-of-App Recommendation System,” Uber Blog, Jun. 13, 2024. [Online]. Available: https://www.uber.com/blog/personalized-marketing-at-scale/
[15] K. P. Yancey and B. Settles, ”A Sleeping, Recovering Bandit Algorithm for Optimizing Recurring Notifications,” in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD)*, 2020.