Interpretable Policies for Dynamic Product Recommendations

Marek Petrik
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
mpetrik@us.ibm.com

Ronny Luss
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
rluss@us.ibm.com

Abstract

In many applications, it may be better to compute a good interpretable policy instead of a complex optimal one. For example, a recommendation engine might perform better when accounting for user profiles, but in the absence of such loyalty data, assumptions would have to be made that increase the complexity of the recommendation policy. A simple greedy recommendation could be implemented based on aggregated user data, but another simple policy can improve on this by accounting for the fact that users come from different segments of a population. In this paper, we study the problem of computing an optimal policy that is interpretable. In particular, we consider a policy to be interpretable if the decisions (e.g., recommendations) depend only on a small number of simple state attributes (e.g., the currently viewed product). This novel model is a general Markov decision problem with action constraints over states. We show that this problem is NP hard and develop a Mixed Integer Linear Programming formulation that gives an exact solution when policies are restricted to being deterministic. We demonstrate the effectiveness of the approach on a real-world business case for a European tour operator's recommendation engine.

1 Introduction

Interpretability in data mining and machine learning means that the computed models and solutions can be relatively easily understood by humans. Examples of algorithms that produce interpretable solutions include classical algorithms, such as decision trees [18], and newer sparse learning methods [18]. Recently, there has been a lot of interest in interpretable machine learning, but few works have explicitly focused on interpretability in decision making.

Lack of interpretable solutions can be an important roadblock in many critical domains, such as medicine [12, 11, 26]. If decision makers can understand a computed solution they are more likely to trust and implement it. Additionally, having an interpretable policy makes it easier to discover flaws resulting from incorrect models or unreasonable model assumptions, and without implementing a policy. Furthermore, it is possible to discover why the policy does not perform well. Finally, a simple policy can be easily implemented [29] and, as we show, sometimes assuming that the policy is simple circumvents the need to build complex models.

In this paper, we view the task of recommending products as one of decision making in a dynamic environment, rather than ranking products in a static setting (which is more common). With the rise in accessible customer information and product availability, the importance of matching customers to products has quickly risen. This problem arises in many domains such as recommender systems [21, 6] and personalized online advertising [8, 16, 7]; similar methods for matching customers to products are even used for email classification and spam detection [20]. Most methods for these systems use both the descriptions of the items and the historical behavior of the users.

Recommender systems fall into two basic categories [24]: collaborative filtering and content-based filtering. Collaborative filtering relates one user's preferences to other users' preferences without taking into account specific user or item properties [24]. Content-based filtering rather makes use of user profiles and the item properties. Most recommender systems combine these ideas and are a hybrid of both collaborative and content-based filtering methods. Successful recommender systems have been developed for recommending movies (e.g., Netflix), music (e.g., Pandora), personalized advertising [32], and even for recommending social-network followers [15, 5].

We focus in this paper on a single, and admittedly simple, model of policy interpretability (i.e., recommendation rules that are easily understood). Since we consider a dynamic setting, we look for recommendations based on the state of a user (e.g., whether or not interested in a currently viewed product), and these recommendations based on the state are what we refer to as a policy. We consider the standard Markov decision process (MDP) with *discrete* states and actions. Typically, an MDP will have thousands or millions of states. Therefore, even though a policy can be examined in principle, it cannot be understood in practice. To make the policy *interpretable*, we simply require that it does not prescribe more than, for example, 50 different actions. This means that the policy must prescribe the same action for a number of states and also that such a subset of states must be well defined.

Computing optimal recommendations under our interpretability constraints constitutes solving a so-called partially observable Markov decision process (POMDP), which can be cast as a Markov decision process. Policies for such MDPs tend to be extraordinarily complex and hard not only to interpret but also to implement. Instead, as in our case, one may want to compute a best possible policy that depends only on the currently viewed product. This would entail computing item to item recommendations that consider the customer dynamics, a policy which falls into a class of POMDPs that are not complex.

To motivate the need for interpretable policies, consider optimizing dynamic online product recommendations. As a user interacts with a website, their preferences become clear over time. However, the interactions themselves influence user behavior [33, 30, 27, 23]. Consider a scenario with two customer types **A** and **B** and two products X and Y where the percentages of customer types **A** and **B** interested in products X and Y are 90%/10% and 40%/60%, respectively. If a customer of type **A** is recommended and clicks on product Y, the distribution of the types of customers looking at the two products will change, and our model accounts for such dynamics.

Our main contribution in this paper is a novel model for computing interpretable policies as described above, or stated differently, a model for computing policies for MDPs which are constrained to take the same action in subsets of states. We show connections with several other models considered in the context of reinforcement learning and POMDPs and use the relationship to show that the optimal interpretable policies may be stochastic and are NP hard to compute. We then propose a new and simpler nonlinear Mixed Integer Linear Program (MILP) formulation. While these models can also be hard to compute, we show that we can learn optimal interpretable deterministic policies.

We also contribute to the area of recommendation engines with a framework for making recommendations that account for the changing dynamics of the system. While we do not model these dynamics directly, our framework does account for more general dynamics by introducing unobservable dynamics into the model. In other words, our model assumes that customers are changing states but also assumes that we cannot directly observe the manner in which they are changing.

Furthermore, we establish a connection between several apparently unrelated areas of optimization. The underlying model is a Markov decision process with constraints on actions, and is related to several other streams of work. Similar models are studied using aggregation in reinforcement learning, where the motivation is somewhat different. Aggregation is used because the models are too large to be solved or even enumerated. As we discuss in more detail later, the interpretable MDP model is also related to finite state controller optimization on POMDPs (e.g. [2]). Finally, policy search methods from reinforcement learning have been used to compute interpretable policies.

The remainder of the paper proceeds as follows. Section 2 next discusses more related work to our notion of interpretability. Section 3 then formalizes the concept of interpretability and discusses the application to recommender systems, followed by mathematical programming formulations used to learn interpretable policies in Section 4, and corresponding complexity results in Section 5. Our framework is evaluated on data from a major European tour operator in Section 6, where a significant improvement over several practical benchmarks is demonstrated. A final discussion of our findings is given in Section 7.

2 Related Work

In this section, we summarize the existing related work and draw new connections. While there has been little work directly on interpretable policies, it turns out that our model of policy interpretability is closely related to previous results in other fields.

Recent work on interpretable policies in reinforcement learning [17] proposes to use policy search methods. Policy search is a general method that can be leveraged to find policies that are parameterized in ways that make them interpretable. While this is a very natural approach and is demonstrated to work well, it does not study any new methods for computing these interpretable policy. In comparison, our focus is on a simpler model that allows us to more deeply study the computational problems.

The most closely related method to our formulation (discussed in Section 4.1) is state aggregation in reinforcement learning, which is a very simple and classic method (e.g., [25, 19, 28, 3, 10]). The motivation in our work is quite different from that in state aggregation. The main reason for state aggregation is to address a problem that is very large and often cannot be enumerated, and therefore, very little focus has been put on how to compute good policies for an aggregation. Indeed, most work has rather focused on methods for choosing which states should be aggregated.

Since we deal with a smaller number of states, we can develop better methods for computing interpretable policies, and will adapt methods used for state aggregation to this setting.

Our model of interpretability can be also seen as a special case of Partially Observable Markov Decision Processes (POMDPs). POMDPs generalize MDPs to the case where observations do not contain exact information about the current state, and in general, the optimal policy to POMDPs are complex. There do exist certain classes of POMDPs that result in simpler optimal policies, and this space is where our model lies. In the other direction, it can also be shown that our interpretable policy problem generalizes one of these simpler classes of POMDPs, termed optimal finite state controllers in the MDP literature. Hence, we proceed as with general POMDPs, and constrain polices by introducing a concept of observations in the next section.

Another closely related area of work to interpretability is that of implementability of policies. The goal is not to have the policy be understood by humans, but instead requires that. The term *implementable policy* in the context of MDPs was introduced in [29]. However, the problem is a special case of finding *memoryless* policies (i.e., policies that depend only on the current state and not on any history) for POMDPs.

Finally, the analysis of policy optimality in some specialized domains, such as inventory management, queueing, and energy storage is tangentially related to our setting. One can show that the optimal policies in these domains are interpretable; indeed, for example, the optimal policy in inventory optimization domains will be independent of the current inventory. This leads to policies that are easy to interpret and also typically easy to solve. Unlike these settings, we deal with cases in which the optimal policy may not be interpretable, which introduces an additional layer of computational difficulties.

3 Interpretable Policies in MDPs

In this section, we formally define the model and illustrate its application to the dynamic recommendation problem described in the introduction. For the remainder of the paper, we define the following notations. Denote by Δ^d the non-negative simplex in d dimensions, i.e., the set of valid distributions defined by $\{x \in \mathbb{R}^d : \sum_i x_i = 1, x \geq 0\}$. For a matrix X, we define $X(s, \cdot)$ as row s of X.

The interpretable model is based on a Markov decision process (MDP) (e.g., [22]). An *interpretable* MDP is a tuple $(\mathcal{S}, \mathcal{A}, P, r, p_0, \mathcal{O}, \theta)$. Here, $\mathcal{S}$ is a *finite* set of states, $p_0 \in \Delta^{|\mathcal{S}|}$ is the initial distribution, $\mathcal{A}$ is a finite set of *actions*, each of which can be taken in all states. The transition probability matrix for each $a \in \mathcal{A}$ is denoted $P_a \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ and each row lies in the simplex: $P_a(s, \cdot) \in \Delta^{|\mathcal{S}|}$ for all $s \in \mathcal{S}$. The rewards vector for each action $a \in \mathcal{A}$ is denoted $r_a \in \mathbb{R}^{|\mathcal{S}|}$.

As discussed in the introduction, we assume that interpretability of a policy depends on a small number of simple state properties. To capture this property, we augment the model by a set of observations $\mathcal{O}$ and an observation function $\theta : \mathcal{S} \to \mathcal{O}$ that defines a partitioning of states to observations.

A solution to an MDP is a randomized stationary policy from $\Pi_R : \{\mathcal{S} \to \Delta^{\mathcal{A}}\}$ or deterministic stationary policy from $\Pi_R : \{\mathcal{S} \to \mathcal{A}\}$. The set of *interpretable* policies Π_I is defined as:

$$\Pi_I = \{\pi \in \Pi_R \ : \ \theta(s_1) = \theta(s_2) \Rightarrow \pi(s_1) = \pi(s_2)\} \, .$$

In other words, if two states share the same observation then an interpretable policy must take actions with identical probabilities in these states.

Our objective is to compute a policy that maximizes the infinite-horizon γ-discounted return $\rho(\pi)$ by solving

$$\max_{\pi \in \Pi_I} \rho(\pi), \tag{3.1}$$

where $\rho(\pi)$ can be expressed as

$$\rho(\pi) = \sum_{t=0}^{\infty} \gamma^t p_0^{\mathsf{T}} P_\pi^t r_\pi .$$

It is important to note that the constraints imposed by interpretable policies are quite different from the constraints in constrained MDPs [1]. The optimal solution to Eq. (3.1) can be obtained through other formulations. In particular, we offer a Mixed Integer Linear Programming formulation in Section 4.2, and equivalence is given there by Proposition 4.1. Eq. (3.1) offers an easy interpretation for our objective when designing a policy.

To illustrate the concept of interpretability, the following example describes how our model can be used to represent the recommender system formulation from the introduction. This is a simplified version of the model that we use later in the paper for empirical evaluation.

Example 3.1 (Dynamic Product Recommendations)**.** *Consider the online product recommendation setting described in the introduction. Let $\mathcal{M}$ be a set of available products and let $\mathcal{W}$ be a set of customer segments. Assuming that customer segments are known in advance, their browsing behavior and response can be modeled as the following MDP. States $\mathcal{S} = \mathcal{W} \times \mathcal{M}$ represent the* customer type *and* current product displayed*. Actions represent the product sets to recommend during a page view: $\mathcal{A} = \{a \in 2^{\mathcal{M}} : |a| \leq k\}$. Here, k is the maximum number of products to recommend. Transition probabilities are based on whether a customer chooses to follow a product* *recommendation, purchases the currently displayed product, or abandons the purchase. Rewards are accrued by customers purchasing items.*

This recommendation MDP can be solved rather easily, however the policy is likely to be hard to implement. This is because the customer type is not observable, particularly if only a short history of customer interactions is available. Another direction, solving this problem as a POMDP, would yield a complex policy that is hard to interpret and implement in a real-time system. This leads to applying our model of interpretability.

The simplest and most interpretable online product recommendation policy are item-to-item recommendations, and capturing the desire for such a policy in our setting is simple. Let the set of *observations* be equal to the available products ($\mathcal{O} = \mathcal{M}$) and define the *observation mapping* function as $\theta(w, m) = m$ for $w \in \mathcal{W}, m \in \mathcal{M}$ so that observations are independent of the customer type. Typically, item-to-item recommendations are computed based on the customers that typically visit the given product page, i.e, the distribution of customers. However, the recommendations themselves influence this distribution. The MDP implementation with interpretable policies accounts for the change in the distribution, and we formulate and analyze this approach in more detail below.

4 Interpretable MDP Formulations

In this section, we formulate the mathematical programs for learning the MDPs of interest, beginning with a classical MDP formulation, adding interpretability constraints, and then relaxing nonlinearities to obtain a mixed integer linear programming (MILP) formulation that approximates the exact problem.

4.1 Basic Formulation

While it is possible to adapt nonlinear optimization formulations from the POMDP literature, for example [2], we derive a simpler nonlinear formulation first. We first adapt the standard linear program formulation for a Markov decision process. The classic MDP linear program learns a policy that maximizes the expected reward of the system subject to constraints that model the transitions allowed in the system.

First define the following variables. Let $u \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ represent the policy we are trying to learn, which is defined in every state as $u(s,\cdot)/\sum_a u(s,a) \in \Delta^{|\mathcal{A}|}$. Note that, for variable u, we denote $u(s,a)$ as the (s,a) entry of matrix u in the following formulations, along with similar notation for other variables. Summations over s or a are meant as shorthand for $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Then, in formal terms, a deterministic policy means that $u(s,a) = 1$ for a single $a \in \mathcal{A}$ for each $s \in \mathcal{S}$, whereas a randomized policy simply means that $\sum_a u(s,a) = 1$ for each $s \in \mathcal{S}$.

Given our notation, the linear programming formulation for the classic MDP is

$$\begin{aligned}
\max_{u} \quad & \sum_{s,a} u(s,a) r(s,a) \\
\text{s.t.} \quad & \sum_{a} u(s,a) \\
& \quad = p_0(s) + \sum_{s',a} \gamma P_a(s,s')\, u(s',a) \quad \forall s \in \mathcal{S} \\
& u(s,a) \geq 0 \quad \forall s \in \mathcal{S}, a \in \mathcal{A}.
\end{aligned} \tag{4.1}$$

The summation in the objective is the expected reward for a policy given by u. Note that u can also be interpreted as a state-action occupancy measure (e.g. [4]) which is the cumulative visitation probability over all state-action pairs when the discount (for computing the current value of future rewards) is interpreted as a probability of leaving the system. Then the first set of constraints can be seen as a form of the Bellman equations in terms of a state-action occupancy measure rather than state value. It is known that an optimal policy must satisfy these equations. The final inequalities are needed since measures must be nonnegative. For the optimal solution, these values represent the optimal policy.

We next want to adapt Problem 4.1 in order to enforce interpretability. Interpretability constraints can be directly added by introducing a new optimization variable, $\psi \in \mathbb{R}^{|\mathcal{O}|\times|\mathcal{A}|}$, which defines the interpretable policy for each observation as $\psi(o,\cdot) \in \Delta^{|\mathcal{A}|}$. The new formulation is

$$\begin{aligned}
\max_{u,\psi} \quad & \sum_{s,a} u(s,a) r(s,a) \\
\text{s.t.} \quad & u(s,a) = \psi(\theta(s),a) \sum_{a'} u(s,a') \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \\
& \sum_{a} u(s,a) \\
& \quad = p_0(s) + \sum_{s',a} \gamma\, P_a(s',s)\, u(s',a) \quad \forall s \in \mathcal{S} \\
& u(s,a) \geq 0 \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \\
& \sum_{a} \psi(o,a) = 1 \quad \forall o \in \mathcal{O}.
\end{aligned} \tag{4.2}$$

In this formulation, we interpret u as a state-action occupancy measure (as defined above). The first set of constraints dictates that the interpretable policy is equivalent to a normalized state-action occupancy measure (as is the policy in the classic MDP above) and furthermore that the policy at states in the same observation is the same. The second and third set of constraints remains the same. The last set of constraints puts the interpretable policy at each observation in the simplex (since nonnegative is already implied by nonnegativity of u).

The optimal policy is directly constructed from the optimal ψ for (4.2). The following proposition shows its correctness.

Proposition 4.1. *The optimal solution to* (4.2) *is also optimal in* (3.1).

Proof. Follows from the equivalence in Theorem 6.9.1 in [22], the optimality in Theorem 6.9.4 in [22], and the equivalence of return in terms of the occupancy frequency. The constraint on u from ψ is correct due to the construction of the policy from u. $\square$

Note that formulation (4.2) is no longer linear or convex because of the terms $\psi(\theta(s), a)u(s, a')$ in the first set of constraints. It could be solved using a non-linear solver, such as IPOPT. We take a different approach, which can guarantee solution optimality, in the next subsection.

4.2 Mixed Integer Linear Program

This section describes a new mixed integer linear programming (MILP) formulation for learning an interpretable policy within the MDP framework. A recent paper describes a MILP formulation for finite state controllers [9], however, our formulation is simpler and our derivation is more straightforward.

As we have defined a state-action occupancy frequency $u(s, a)$ above, we can similarly define a state occupancy frequency $d \in \mathbb{R}^{|\mathcal{S}|}$ by $d(s) = \sum_a u(s, a)$ for each state $s \in \mathcal{S}$. From a modeling viewpoint, including d gives a new interpretation of state-action occupancy frequency as the fraction of state occupancy frequency determined by the optimal interpretable policy. From an optimization viewpoint, including d greatly reduces the number of nonlinear functions in the first set of constraints in a trade-off for $|\mathcal{S}|$ additional equality constraints. This trade-off is important regarding the relaxation below that we use to get an MILP formulation. Introducing the state occupancy frequency to the formulation results in the problem

$$\begin{aligned}
\max_{u,d,\psi} \quad & \sum_{s,a} u(s,a)r(s,a) \\
\text{s.t.} \quad & u(s,a) = \psi(\theta(s),a)d(s) \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \\
& d(s) = p_0(s) + \sum_{s',a} \gamma\, P_a(s',s)\, u(s',a) \quad \forall s \in \mathcal{S} \\
& d(s) = \sum_a u(s,a) \quad \forall s \in \mathcal{S} \\
& u(s,a) \geq 0 \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \\
& \sum_a \psi(o,a) = 1 \quad \forall o \in \mathcal{O}.
\end{aligned} \tag{4.3}$$

Note that the third set of constraints are the $|\mathcal{S}|$ new equality constraints, and that the number of nonlinear terms in the first set of constraints has been reduced from $|\mathcal{S}| \cdot |\mathcal{A}|^2$ to $|\mathcal{S}| \cdot |\mathcal{A}|$ nonlinear terms.

While we can try to solve the nonlinear optimization problem (4.3), any solution we get will have no guarantee of optimality. We now develop a mixed integer linear program formulation based on the so-called McCormick inequalities. This approach relaxes the constraints in (4.3). The idea is to bound the terms $\psi(\theta(s), a)d(s)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$ by making use of upper and lower bounds on $\psi(\theta(s), a)$ and $d(s)$. McCormick inequalities are defined in the lemma below.

Lemma 4.2 (McCormick Inequalities, e.g., [13])**.** *Assume that $a_L \leq a \leq a_U$ and $b_L \leq b \leq b_U$. Then:*

$$a\, b_L - a_L\, b_L + a_L\, b \leq a\, b \leq a\, b_U - a_L\, b_U + a_L\, b$$
$$a\, b_U - a_U\, b_U + a_U\, b \leq a\, b \leq a\, b_L - a_U\, b_L + a_U\, b$$

It is important to note that the equalities are attained for the extreme points of the intervals (for either one of the two variables).

The McCormick inequalities are easy to derive; for example, the first lower bound on ab is attained by multiplying the bounds $a - a_L \geq 0$ and $b - b_L \geq 0$. The MILP can then be constructed as follows. The constraints on ψ are box constraints of the form $0 \leq \psi(\theta(s), a) \leq 1$ and the bounds on state occupancy frequencies are $0 \leq d(s) \leq \bar{d}(s)$ where $\bar{d}(s)$ must be estimated. Therefore, we can relax the nonlinear equality $u(s, a) = \psi(\theta(s), a)d(s)$ in problem 4.3 by the following inequalities

$$\bar{d}(s)\,(\psi(\theta(s),a) - 1) + d(s) \leq u(s,a) \leq \bar{d}(s)\,\psi(\theta_s,a)$$

Note that all four McCormick inequalities are implied by these two constraints.

The relaxed optimization problem becomes:

$$\begin{aligned}
\max_{d,u,\psi} \quad & \sum_{s,a} u(s,a)\, r(s,a) \\
\text{s.t.} \quad & \bar{d}(s)\,(\psi(\theta(s),a) - 1) + d(s) \\
& \quad \leq u(s,a) \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \\
& u(s,a) \leq \bar{d}(s)\,\psi(\theta(s),a) \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \\
& d(s) = p_0(s) + \sum_{s',a} \gamma\, P(s',a,s)\, u(s',a) \quad \forall s \in \mathcal{S} \\
& d(s) = \sum_a u(s,a) \quad \forall s \in \mathcal{S} \\
& \sum_a \psi(o,a) = 1 \quad \forall o \in \mathcal{O} \\
& u(s,a) \geq 0 \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \\
& \psi(o,a) \in \{0,1\} \quad \forall o \in \mathcal{O}, a \in \mathcal{A}
\end{aligned} \tag{4.4}$$

We next note that the last statement of Lemma 4.2 implies that we can compute an optimal *deterministic* policy (under interpretability constraints) as well as an optimality gap for suboptimal policies, which is summarized in the following proposition.

Proposition 4.3. *Optimal solution to* (4.4) *is also an optimal solution to* (3.1).

Proof. Since the McCormick inequalities are tight for $\psi(o, a) \in \{0, 1\}$, the optimal solution to this MILP problem will be the optimal *deterministic* implementable policy for the problem. □

It may be also possible to solve (4.4) without the integrality constraints, which means optimizing over randomized policies. In that case, this is simply a linear program which can be easily solved, however, the solution may not be very good due to the relaxation given by the McCormick inequalities.

Remark 4.4. As we hinted, there is a connection between finite state controllers and the interpretability constraints. Consider a POMDP with m states, a actions, o observations, and computing a finite state controller with n nodes. [9] shows that their MILP formulation has $n \cdot a + n^2 \dot{o}$ integral variables. It can be readily seen that applying our formulation would have $n^2 \cdot a \cdot o$. This is more than [9]. The number of variables can reduced in a setting such as the finite state controller in which the action set can be decomposed as follows. Assume that $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$. Then introduce additional variables $\psi_i : \mathcal{O} \times \mathcal{A}_i \to \{0, 1\}$ for $i = 1, 2$. We add constraints $\sum_{a \in \mathcal{A}_i} \psi(o, a_i) = 1$ and $\psi(o, (a_1, a_2)) \le \min\{\psi_1(o, a_1), \psi(o, a_2)\}$.

5 Properties of Optimal Interpretable Policies

In this section, we prove basic properties of optimal interpretable policies. Recall that an interpretable policy takes the same action for all states within a single observation and is computed by solving problem (3.1).

We first address the computational complexity of computing an optimal interpretable policy.

Proposition 5.1. *Solving* (3.1) *for either randomized or deterministic policies is NP hard, and is in NP as well for deterministic policies.*

Proof. Hardness is proven by a reduction from computing a memoryless policy for a POMDP [14] (which is known to be NP-hard to compute). The construction is simple and we only outline it here. Consider a POMDP with states $\bar{\mathcal{S}}$ and observations $\bar{\mathcal{O}}$. Then construct a cross-product MDP (e.g. [2]) for a finite node controller with $\bar{\mathcal{O}}$ nodes. The state space of the cross-product MDP will have $\mathcal{S} = \bar{\mathcal{S}} \times \bar{\mathcal{O}}$. The set of actions in the cross-product MDP is the same as the actions in the POMDP. The transitions in the cross-product MDP are a product of the POMDP state transition according to the action taken and the observation observed. Consider an interpretable policy for this problem in which the observations depend on $\bar{\mathcal{O}}$ and the observation mapping is $\theta((s, o)) = o$. It can be now readily seen that computing such an interpretable policy will correspond to a memoryless policy in the POMDP and vice versa. An optimal interpretable policy for this problem will therefore be also an optimal memoryless policy in the POMDP. The hardness for stochastic policies follows similarly from [34]. It follows that solving (3.1) for deterministic policies also in NP since the set of deterministic policies can be enumerated. □

The problem of solving (3.1) over randomized policies is not known to be in NP and there is evidence that the problem may in fact be harder [34].

We next study the structure of optimal policies. It can be readily seen that the optimal policy to an MDP with interpretability constraints may be history dependent. See, for example, the MDP in Fig. 1 in which the optimal policy will be $a_2, a_1, a_2, \ldots$. No deterministic interpretable policy can achieve such a return. However, we are interested in finding a stationary policy. The following proposition shows that the optimal policy may need to be randomized.

Proposition 5.2. *There may be no optimal deterministic interpretable policy.*

We prove Proposition 5.2 by an example in which a randomized policy can be arbitrarily better than the best deterministic policy.

Example 5.3. *Consider an example MDP depicted in Fig.* 2 *in which the optimal policy will be randomized. One possible interpretation in the context of product recommendations is as follows. States of the MDP represent a customer state. In particular, s_1 means that the customer is not yet interested in a product, and action a_1 represents a good recommendation. Taking this action moves the customer to state s_2 which means considering a product they are interested in. Taking action a_2 will cause the customer to consider another interesting product. Action a_2 represents suggesting an irrelevant product. In state s_1, the customer remains uninterested, but in s_2 the customer is already considering an interesting product and since the recommendation is not useful, they will simply purchase the product and a reward is received. Since we do not observe the internal customer state, s_1 and s_2 share the same observation and the interpretable policy will have to take the same action in both states. Clearly, any deterministic policy will receive return* 0 *(since the system starts in state s_1), while a policy that randomizes a_1 and a_2 with equal probability will receive return of at least $\gamma 0.5$ for discount factor γ. Looking*

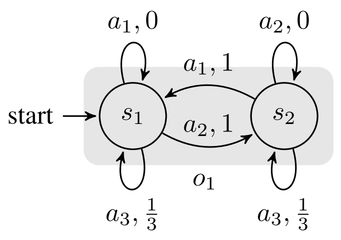

Figure 1: Markov decision process in Example 5.3. Shaded areas show states that share the same observation. Edge labels denote action name and the reward.

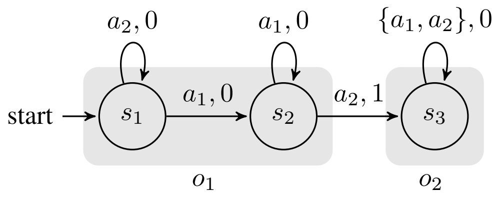

Figure 2: Markov decision process in Example 5.3. Shaded areas show states that share the same observation. Edge labels denote action name and the reward.

back at Fig. 1, we have an even simpler example in which the optimal policy will be randomized.

Note that our MILP formulation (4.4) only learns deterministic policies, and that the binary constraints need to be relaxed in order to learn randomized policies. Initial experiments showed poor performance with this relaxation, likely due to the McCormick inequalities being too loose an approximation to the other nonlinearities.

6 Case Study: Interpretable Product Recommendations

In this section, we describe the experimental evaluation of our interpretable MDP model in an online product recommendation setting. As mentioned above, most research on recommender systems has focused on developing methods for fitting a customer preference model to data. We, in contrast, study optimization methods that use the customer preference model to make better recommendations.

The motivating business case for this application was to improve customer experience and promote conversions for a major European tour operator. For the purposes of this experimental study, we apply the tools of this paper to design a recommendation engine that targets conversions, i.e., customer purchases.

6.1 Customer Model

To evaluate the quality of recommendations, we simulate customer purchase behavior and interactions with an online catalog. Our focus is on simulating online sessions in which customers browses different products and at some point either make a purchase or abandon the session.

Customer behavior is modeled using a mixed logit customer choice model [31] with 10 discrete segments. The logit model is used to predict customer behavior during a product purchase session. In particular, the model decides whether a customer purchases the currently viewed product, takes a recommendation, or abandons the search.

The logit model assumes that customers from any segment c assign some value, denoted $\eta(c, p)$, to product p. This represents the value gained by purchasing the product and is used to determine the probability of purchasing the given product. As a baseline, we denote by $\eta_N(c)$ the value of no purchase, which is the standard approach. We describe how the values $\eta(c, p)$, for each segment and product, and $\eta_N(c)$, for each segment, translate to probabilities below.

We fit the parameters of the logit model to data which comes from an adventure travel brand of a major European tour operator that specializes on sailboat rentals. Our dataset, which is a subset of the entire clickstream, consists of 22803 individual website visits, 1100 individual customers, and 75 products. Customer segments are identified using a standard low-rank matrix decomposition method and logit parameters are fit to maximize likelihood.

The MDP that models the interaction of customers with the online system is defined as follows. The state set is $\mathcal{S} = \mathcal{C} \times \mathcal{P}$, where $\mathcal{C}$ is the set of customer segments, and $\mathcal{P}$ is the set of products. In other words, the state represents a customer and a product currently being considered. Assume that the goal is to recommend n products. Action set $\mathcal{A} = \{(p_1, \ldots, p_n) \; : \; p_i \in \mathcal{P}\}$ determines the set of recommended products in a given state.

The transition probabilities in the MDP for some state (c, p) and products $(p_1, \ldots, p_n)$ are given according to the mixed logit choice model as follows. In particular, let $\alpha = \exp(\eta(c, p)) + \cdot \sum_{j=1}^{n} \exp(\kappa_R \cdot \eta(c, p_j)) + \sum_{p' \in \mathcal{P}} \exp(\kappa_N \cdot \eta(c, p')) + \exp(\eta_N(c))$ be a normalization constant. Here, $\kappa_R \leq 1$ represents the propensity for taking a recommendation and $\kappa_N \leq 1$ represents the propensity of choosing another product directly from the catalog.

Given the normalization constant α, the probability of a customer purchasing a product is $\exp(\eta(c, p))/\alpha$, the probability of taking recommendation p_j is $\exp(\kappa_R \cdot \eta(c, p_j))/\alpha$, the probability of using the menu to choose another product p' is $\exp(\kappa_N \cdot \eta(c, p'))/\alpha$, and finally the probability of abandoning the session is $\exp(\eta_N(c))/\alpha$. The rewards in the MDP model are 1 when a purchase is made and 0 otherwise. This essentially assumes that the margins are constant, but the model could be easily extended to weighted margins.

Note that the states in the MDP above describe both the product and customer segment. While such an MDP can be solved the policy cannot be implemented because the customer segment is not observed. This problem could also be solved using POMDP techniques; however, as suggested in the Introduction, the final solution would be hard to interpret and also difficult to implement in a real-time setting. Instead, we seek to find good *interpretable* policies.

Perhaps the simplest interpretable policy in the product recommendation setting is the so-called product-to-product recommendation. In this setting, recommendations are static and are solely a function of the currently viewed product. The simplicity of this method makes it popular in practice and is a good fit with our model of interpretable policies. It can be readily seen that the set of policies can be constrained to product-to-product policies by defining observations as $\mathcal{O} = \mathcal{P}$ and $\theta_{(c,p)} = p$, i.e., the action is independent of the customer.

6.2 Simulation Results

The goal of the empirical evaluation is to determine the possible benefit from using interpretable policies in making product recommendation as compared with more traditional product-to-product recommendations.

We compare three methods: *Static*, *Iterated*, *IMDP*. The *static* method is the simplest one and entails simply making the recommendation that is most likely to be appreciated by types of people looking at the current product. These standard methods—as described in the introduction—reflect how recommendations for other products can impact the distribution of customers considering the current product. The *iterated* method expands on this idea by re-optimizing the recommendations three times (i.e., implementing static recommendations using simulation, recomputing customer distributions, and repeating three times). The *IMDP* method uses the MILP formulation to compute an interpretable policy for the MDP described above. As solvers for MILP can be quite computationally expensive, we apply two versions of *IMDP*: Methods *IMDP(3)* and *IMDP(50)* represent results after 3 and 50 minutes of computation using CPLEX on an AMD Phenom II X6 machine.

To determine the improvement, we compare these product recommendation methods on several randomly generated scenarios, each of which is restricted to 25 randomly selected products and a single product recommendation. To make the results comparable across scenarios, we normalize them using lower (baseline) and upper bounds on the conversion rate. Our baseline is a policy that makes no recommendations, and the upper bound is a clairvoyant policy. The clairvoyant assumes that the precise customer types are known, solves the MDP, and implements that policy.

Table 1 depicts the normalized experimental results. The percentages are computed as follows. Let l be the conversion rate of the baseline policy, u be the conversion rate of the clairvoyant policy, and q be the conversion rate of the tested policy. Then the improvement is computed as: $(q - l)/(u - l)$. Fig. 3 depicts the average improvement over 50 runs of the three methods.

#	Static	Iterated	IMDP(3)	IMDP(50)
1	56.8%	56.8%	**72.9%**	**72.9%**
2	19.8%	10.0%	43.9%	**45.9%**
3	24.2%	16.2%	54.3%	**55.2%**
4	31.6%	25.0%	65.9%	**66.2%**
5	17.0%	17.8%	54.0%	**54.2%**
6	44.5%	44.5%	**75.2%**	**75.2%**
7	32.7%	39.7%	73.1%	**73.2%**
8	**66.7%**	**66.7%**	55.8%	57.3%
9	25.0%	25.0%	62.9%	**63.0%**
10	19.4%	16.0%	53.2%	**53.6%**

Table 1: Percentage of improvement in conversion rate over no recommendation compared with clairvoyant policy for 10 problem instances.

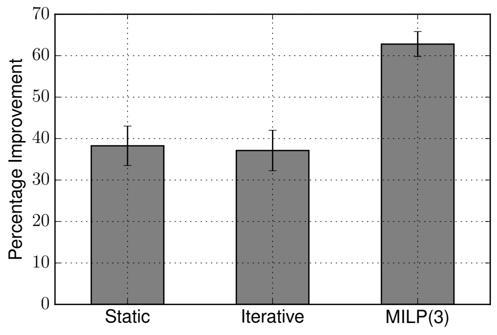

Figure 3: Average percentage improvement in conversion rate with 95% confidence interval.

There are several interesting observations that can be inferred from our results. First, IMDP significantly outperforms the standard recommendation techniques. At least in our scenario, considering the affected distribution of customers appears to be very important. Second, surprisingly, iterating the recommendations often not only does not increase the conversion rates, but actually decreases them (i.e., the *Static* method sometimes outperforms the *Iterated* method). Finally, while 3 minutes are not enough to compute an optimal MILP solution, our results indicate that the interpretable policy achieves over 50% of the benefit of a clairvoyant policy while being much simpler.

We also investigated recommending more than a single product. Unfortunately, the number of integer variables in our formulation scales exponentially with the number of products. It can be readily seen, however, that the action constraints can be decomposed in a manner that is similar to the discussion in Remark 4.4. This reduces the number of variables to be linear in the number of products, but our empirical results indicate that this approach does not significantly reduce the computation time. Regardless, our experiments with a single recommendation already demonstrate the usefulness of our notion of interpretability.

7 Concluding Remarks

In this paper, we have derived and implemented a novel framework for making dynamic product recommendations. The key insight is that modeling customer states is often quite challenging due to lack of knowledge about the customer, but that underlying dynamics (i.e., customers changing their state) can still be accounted for by using a concept of interpretability. Namely, recommendations are restricted to being identical for customers that are in the same observation, where each observation is a union of customer states that are not observable. Restricting the recommendations in this manner makes the recommendation policy interpretable (i.e., not complex and easily understandable). We use tools from Mixed Integer Nonlinear Programming in conjunction with a relaxation using McCormick inequalities to learn interpretable policies, but there are other directions not pursued in this work, such as combining a semidefinite programming (SDP) relaxation with Reformulation Linearization Technique (RLT) inequalities (which generalize McCormick inequalities) used to tighten (rather than relax) the SDP relaxation.

As noted above, regarding scalability, there is much room for improvement since the number of binary variables in our MILP formulation grows exponentially with the number of products recommended (i.e., the dimension of the actions). Aside from the decomposition described above, a different direction is a heuristic that fixes all but one of the recommendation action dimensions and iteratively optimizes over individual recommendations. This alternating minimization scheme has efficient iterations (which we have already demonstrated with our single product recommendations above), so performance depends on the number of iterations required for a good solution.

Note also that uncertainty in the model parameters, rewards and transition matrices, have an important impact on policy performance. While we have not addressed such uncertainty here, it is an important topic for future work. Finally, there are several dynamics to the model that we have not yet addressed. For example, reviews written by customers and product ratings affect the distribution of customers looking at products.

References

[1] E. Altman. *Constrained Markov Decision Processes*. 1998.

[2] C. Amato, D. S. Bernstein, and S. Zilberstein. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*, 21(3):293–320, 2009.

[3] D. P. Bertsekas and D. A. Castanon. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transations on Automatic Control*, 34(6):589–598, 1989.

[4] E. A. Feinberg and U. G. Rothblum. Splitting Randomized Stationary Policies in Total-Reward Markov Decision Processes. *Mathematics of Operations Research*, 37(1):129–153, 2012.

[5] J. Hannon, M. Bennett, and B. Smyth. Publisher Recommending Twitter Users to Follow using Content and Collaborative Filtering Approaches. In *ACM Conference on Recommender Systems*, 2010.

[6] L.-p. Hung. A personalized recommendation system based on product taxonomy for one-to-one marketing online. *Expert Systems with Applications*, 29(2):383–392, 2005.

[7] P. Kazienko and M. Adamski. AdROSA: Adaptive personalization of web advertising. *Information Sciences*, 177(11):2269–2295, June 2007.

[8] D. Konopnicki, D. Shmueli-Scheuer, M. Cohen, B. Sznajder, J. Herzig, A. Raviv, N. Zwerling, H. Roitman, and Y. Mass. A statistical approach to mining customers' conversational data from social media. *IBM Journal of Research and Development*, 57(3/4):14:1 – 14:13, 2013.

[9] A. Kumar and S. Zilberstein. History-Based Controller Design and Optimization for Partially Observable MDPs. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 156–164, 2015.

[10] T. J. Lambert, M. A. Epelman, and R. L. Smith. Aggregation in Stochastic Dynamic Programming. Technical report, University of Michigan, 2004.

[11] B. Letham, C. Rudin, T. H. Mccormick, and D. Madigan. An Interpretable Stroke Prediction Model Using Rules and Bayesian Analysis. (609):65–67, 2010.

[12] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2013.

[13] J. Linderoth. A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. *Mathematical Programming, Series B*, 103(2):251–282, 2005.

[14] M. Littman. Memoryless policies: Theoretical limitations and practical results. In *International Conference on Simulation of Adaptive Behavior*, 1994.

[15] X. Liu and K. Aberer. SoCo: a social network aided context-aware recommender system. *Proceedings of the World Wide Web Conference*, pages 781–791, 2013.

[16] H. Ma, D. Zhou, C. Liu, M. Lyu, and I. King. Recommender systems with social regularization, 2011.

[17] F. Maes, R. Fonteneau, L. Wehenkel, and D. Ernst. Policy Search in a Space of Simple Closed-form Formulas: Towards Interpretability of Reinforcement Learning. *Discovery Science*, 7569(1):37–51, 2012.

[18] D. M. Malioutov and K. R. Varshney. Exact rule learning via Boolean compressed sensing. In *International Conference on Machine Learning (ICML)*, pages 1802–1810, 2013.

[19] R. Mendelssohn. An iterative aggregation procedure for Markov decision processes. *Operations Research*, 30(1):62–73, 1982.

[20] V. Metsis, I. Androutsopoulos, and G. Paliouras. Spam filtering with naive Bayes - which naive Bayes? In *Third Conference on Email and Anti-Spam*, 2006.

[21] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *ACM conference on Digital libraries*, pages 195–204, 2000.

[22] M. L. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., 2005.

[23] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized Markov chains for next-basket recommendation. In *International conference on World wide web (WWW)*, pages 811–820, 2010.

[24] F. Ricci, L. Rokach, and B. Shapira. *Recommender Systems Handbook*. Springer US, Boston, MA, 2011.

[25] D. F. Rogers, R. D. Plante, R. T. Wong, and J. R. Evans. Aggregation and disaggregation techniques and methodology in optimization. *Operations Research*, 39(4):553–582, 1991.

[26] C. Rudin. Algorithms for interpretable machine learning. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, page 1519. ACM, 2014.

[27] N. Sahoo. A Hidden Markov Model for Collaborative Filtering A Hidden Markov Model for Collaborative Filtering. *MIS Quarterly*, 36(4):1329–1356, 2012.

[28] P. J. Schweitzer, M. L. Puterman, and K. W. Kindle. Iterative Aggregation-Disaggregation Procedures for Discounted Semi-Markov Reward Processes. *Operations Research*, 33(3):589–606, 1985.

[29] Y. Serin and V. Kulkarni. Implementable policies: discounted cost case. In *Computations with Markov Chains*, pages 283–306. 1995.

[30] P. V. Singh. Seeking Variety : A Dynamic Model of Employee Blog Reading Behavior. 2010.

[31] K. E. Train. *Discrete Choice Methods with Simulation*. 2003.

[32] S. Velusamy, L. Gopal, S. Bhatnagar, and S. Varadarajan. An efficient ad recommendation system for TV programs. *Multimedia Systems*, 14(2):73–87, 2008.

[33] P. Viappiani and C. Boutilier. Recommendation Sets and Choice Queries: There Is No Exploration/Exploitation Tradeoff! In *National Conference on Artificial Intelligence (AAAI)*, 2011.

[34] N. Vlassis, M. Littman, and D. Barber. On the computational complexity of stochastic controller optimization in POMDPs. *ACM Transactions on Computation Theory*, V(212):1–7, 2012.