Integrating Data Modeling and Dynamic Optimization using Constrained Reinforcement Learning

Naoki Abe, Prem Melville, Chandan K. Reddy[*], Cezar Pendus, David L. Jensen[†]
Mathematical Sciences Department
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

ABSTRACT

In this paper, we address the problem of tightly integrating data modeling and decision optimization, particularly when the optimization is dynamic and involves a sequence of decisions to be made over time. We propose a novel approach based on the framework of constrained Markov Decision Processes, and establish some basic properties concerning modeling/optimization methods within this formulation. We conduct systematic empirical evaluation of our approach on resource-constrained versions of business optimization problems using two real world data sets. In general, our experimental results exhibit steady convergent behavior of the proposed approach in multiple problem settings. They also demonstrate that the proposed approach compares favorably to alternative methods, which loosely couple data modeling and optimization.

1. INTRODUCTION

As the domains and contexts of data mining applications become rich and diverse, the technical requirements for the analytical methods are becoming increasingly complex. In particular, when estimated models are used as input to a subsequent decision making and optimization process, it is sometimes imperative that the constraints that govern the execution of those decisions be taken into account in the estimation process itself. In most application situations today, however, practitioners tend to treat the problems of data modeling and decision optimization independently, either for the benefit of simplified problem formulation, or due to the lack of a unifying theory. In this paper, we address the problem of tightly integrating data modeling and optimization, particularly in the context in which the optimization is dynamic in nature involving a sequence of decisions over time. We propose a specific approach based on a novel variant of the framework of constrained reinforcement learning.

The issue of integrating data modeling (estimation) and decision making has recently attracted considerable attention in the data mining community, with notable examples being the body of work in *cost-sensitive learning* [19, 8, 24, 11, 7] and the emerging topic of *economic learning* [14]. Past works in cost-sensitive learning have established that, in many cases, the estimation process can benefit from incorporating the cost information for the actions taken as a result of the estimation. In economic learning, the cost associated with the acquisition of data used for estimation is also taken into account. In the dynamic problem setting, initial attempts have been made to extend cost-sensitive learning to *sequential cost-sensitive learning*, formulating the problem using the Markov Decision Process (MDP) framework, and invoking reinforcement learning methods [13]. The cost structure treated in the past works, however, was in terms of minimization of a simple cost function. In real world applications, it is often the case that various constraints exist that restrict the space of permissible decisions. It is natural, therefore, to extend the goal of cost minimization to constrained cost optimization, in the context of sequential cost-sensitive learning, and this is exactly what we set out to do in this paper.

For illustrating the ideas set forth, let us consider the familiar application example of targeted marketing. Assume we are given a set of customer data that summarize their behavior over time and possibly the corresponding marketing actions taken on them in the past, as features. Our goal, loosely speaking, is to determine a subset of customers that we should approach with a marketing action (such as mailing), or more generally, choose from a set of possible marketing actions the most appropriate one for each of them, possibly repeatedly over time. Additionally, there may be fixed resources available to carry out the marketing actions, such as personnel in a mailing room or a call center, which may impose constraints on the permissible marketing actions at the population level, at any given point in time.

A standard classification method applied to this problem would try to accurately estimate the subset of the customer base who are likely to respond and generate positive profits. A cost-sensitive learning method would try to estimate the subset of customers who, when targeted, would yield the maximum overall profits (or equivalently with minimum cost) [19, 8, 24, 11, 7]. A sequential cost-sensitive learning method, such as a reinforcement learner, would attempt to come up with a marketing policy which maps each customer to an appropriate marketing action and, when followed consistently over time, would maximize the long term overall benefits [13, 22]. A resource-constrained extension of sequential cost-sensitive learning would have to find a resource allocation policy which maps the customers to marketing resources/actions and, when followed over time, would max-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...$5.00.

imize the long term reward, *while adhering to a resource constraint requirement.*

A natural approach in devising a resource-constrained, sequential cost-sensitive learning method may be to use a standard reinforcement learning method to estimate the values of competing actions, and feed them as input to the resource optimization problem. This approach might suffice for some application scenarios, but in situations involving complex problem formulations, it may be sub-optimal. Recall that the key idea of reinforcement learning is in the way it estimates the value of a current action as the cumulative reward expected when following the policy in question in the future. The key issue with the approach described above is that the look-ahead used in estimating the expected cumulative reward would not take into account the resource constraints that need to be observed in reality. In the context of targeted marketing, for example, one of the marketing actions may be to give a customer a loyalty membership status, with high expected value but with high future demands on the resources. Since giving a loyalty status itself may not immediately require significant resources, applying a standard reinforcement learning method might output a policy that attempts to grant such a status to many more customers than is desired. A constrained reinforcement learning method would not suffer from this shortcoming, precisely because it evaluates the expected cumulative reward with the resource constraints taken into account.

There exist some past works that investigated constrained versions of Markov Decision Process (e.g. see Altman [2]), but our approach differs from this body of work in some key aspects. In particular, in contrast to the earlier general formulations in which the constraints are defined as a cost functional of the entire state trajectory, we formulate the constraints as being fixed along the state trajectory. This stationarity assumption is one that we think is reasonable for the problem setting we consider (batch or population learning), and we leverage the simplified formulation to arrive at a practically viable approach. These proposed methods, based on constrained versions of (batch) reinforcement learning algorithms, would not naturally fall out of the general constrained MDP formulation of the type studied in [2]. We also establish some basic properties about some representative algorithms within this formulation. Specifically, we establish the convergence of a constrained value iteration procedure and a constrained Q-learning algorithm under some technical conditions. We conduct systematic empirical evaluation of our approach on resource-constrained versions of business optimization problems, using some real world data sets and making appropriate modifications. Our experiments show that, on a couple of real world application domains, the proposed approach excels, as compared to an existing alternative approach based on combination of unconstrained MDP and constrained optimization.

The remainder of the paper is organized as follows. We first describe the constrained Markov Decision Process and Reinforcement Learning framework we employ in this paper, and establish some basic properties. We then describe a concrete algorithm we employ in our empirical evaluation, also elaborating on the details of the constrained resource optimization formulation. Next we describe detailed business problem formulations we consider in our experiments along with the experimental set-up. Following this we present experimental results and their analyses. We conclude by discussing some issues and open problems.

2. PRELIMINARIES: MARKOV DECISION PROCESSES

We begin with a brief description of the concepts of Markov Decision Process (MDP) [17, 15, 10, 16]. An MDP consists of the state space, S, the action space A, initial state distribution $\mu : S \to \mathcal{R}$, and transition probability function $\tau : S \times A \times S \to [0, 1]$ such that $\forall s \in S, \forall a \in A \sum_{s' \in S} \tau(s'|s, a) = 1$, and the expected reward function $R : S \times A \to \mathcal{R}$. We let $\tau(s'|s, a)$ denote the conditional probability of transition to s' upon taking action a in state s, and $R(s, a)$ denote the expected reward for the same. (We sometimes let $\rho(s, a)$ denote the random variable assuming the reward value also.) For notational convenience, we also let τ denote the random variable that takes on values from S, according to τ, namely

$$\forall s \in S, a \in A,\ \tau(s, a) = s' \text{ with probability } \tau(s'|s, a)$$

Given an MDP, a policy $\pi : S \to A$ determines an agent's behavior in it. In particular, it determines an infinite sequence of state, action, and reward triples, $\langle s_t, a_t, r_t\rangle$, $t = 1, 2, ...$, where the initial state s_1 is drawn according to μ and in general at time t, the action a_t is determined as $\pi(s_t)$ and reward r_t is determined by $R(s_t, a_t)$ and the next state s_{t+1} is determined by the state transition probability $\tau(s_{t+1}|s_t, a_t)$. We also consider stochastic policies, probabilistically mapping states to actions, namely $\pi : S \times A \to \mathcal{R}$ such that $\forall s \in S$, $\sum \pi(s, a) = 1$. Analogously to τ above, for a stochastic policy π, we ambiguously let $\pi(s)$ denote the random variable assuming values from A according to π. That is,

$$\forall s \in S,\ \pi(s) = a \text{ with probability } \pi(s, a)$$

In general, for any stochastic policy π, the value function for π, denoted $V_\pi : S \to \mathcal{R}$, is defined as the expected cumulative reward when starting with a given state and following policy π at every step in the future. Formally,

$$V_\pi(s) = E_{\pi,\tau}[R(s, \pi(s)) + \gamma V_\pi(\tau(s, \pi(s)))]$$

where γ is a discount factor satisfying $0 < \gamma < 1$. It is also customary to define the following two-place variant of value function, as a function of a state and an action.

$$V_\pi(s, a) = E_\pi[R(s, a) + \gamma V_\pi(\tau(s, a), \pi(\tau(s, a)))]$$

More generally, a value function V is any mapping from $S \times A$ to $\mathcal{R}$, and we let $\mathcal{V}$ denote the set of all such value functions. It is well-known that for any MDP, there exists a policy π^* that satisfies Bellman's fix-point equation:

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

and such π^* is optimal, namely

$$\forall \pi,\ \forall s \in S,\ V_{\pi^*}(s) \geq V_\pi(s)$$

It is also the case therefore that

$$\forall \pi,\ E_{s\sim\mu}[V_{\pi^*}(s)] \geq E_{s\sim\mu}[V_\pi(s)]$$

where $E_{s\sim\mu}$ denotes the expectation when s is distributed according to μ. (In general, whenever it is clear from context, we also use E_μ to denote the same.) It also holds that the following alternative form of Bellman's equation for the two-place version of value function is satisfied by the optimal policy.

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

The two-place value function was introduced by Watkins [20], and is also known as the Q-value function. We use both V and Q interchangeably in this paper.

The above fact gives rise to an iterative procedure known as "value iteration", which is stated below for the two-place version of the value function. At step 1, for each state $s \in S$, it initializes the value function estimates using the expected immediate reward function R:

$$V_1(s,a) = R(s,a)$$

At any of the subsequent iterations, t, it updates the value function estimate for each state s and action a according to the following update.

$$V_t(s,a) = E_\tau[R(s,a) + \max_{a' \in A} \gamma V_{t-1}(\tau(s,a),a')]$$

It is known that the above procedure converges, and it converges to the value function of an optimal policy.

3. CONSTRAINED REINFORCEMENT LEARNING

3.1 A Constrained Population MDP

In our formulation, a constrained population MDP is an MDP, in which there exists a prescribed, constrained set Π of permissible policies. Here we consider a version of constrained MDP in which Π is determined with respect to a fixed state distribution μ. More specifically, we consider a *linearly constrained population* MDP, where Π_μ is a set of stochastic policies $\pi : S \times A \to [0,1]$ adhering to a set of n linear constraints of the following form:

$$E_{\mu,\pi}[C_{i,s,a}\pi(s,a)] \leq B_i, i = 1, ..., n$$

where, for each i, the coefficients $C_{i,s,a}$ determine a linear constraint.

This formulation is in contrast with the existing definitions of constrained MDP in the literature (e.g. [2]), in which the constraints are defined in terms of the state trajectory of a policy, rather than being fixed *a priori*. As we will see in the subsequent developments, this formulation results in considerable simplification that motivates, and justifies, a family of algorithms that are relatively straightforward extensions of existing reinforcement learning methods.

The above formulation of constraints admits the following interpretation: Suppose that we are given a population of individuals with associated states, and they are distributed according to μ. At any given point in time, there is a fixed amount of resources, which can be allocated to actions targeting this population. The fact that the constraints are defined with respect to a fixed distribution implicitly relies on the premise that actions taken now will not significantly alter the overall population mixture. While this condition may appear unreasonable for a single agent reinforcement learning scenario, it is quite reasonable for application domains in which the population of states are determined largely by external factors (e.g. the distribution of customers' demographics will remain largely unaffected by a single enterprise's marketing actions.)

Specifically, in the application domain we primarily target in this paper, it is normally the case that there is an existing data set which is to be used for batch learning and optimization. Our premise is that data have been collected for a long enough period of time for a large enough population that the distribution of states found in the data are stable.

3.2 Constrained Value Iteration

Given the above definition of constrained population MDP, we consider the constrained value iteration procedure, analogously to the generic value iteration procedure described in Section 2. At step 1, it initializes the value function estimates for all of the states $s \in S$ and actions as follows.

$$V_1(s,a) = R(s,a)$$

At any of the subsequent iterations, t, it updates the value function estimates for all the states $s \in S$ as follows:

$$V_t(s,a) = E_{\pi_t^*,\tau}[R(s,a) + \gamma V_{t-1}(\tau(s,a), \pi_t^*(\tau(s,a)))]$$

where π_t^* is determined by

$$\pi_t^* = \arg\max_{\pi \in \Pi} E_{\mu,\pi}[R(s,a) + \gamma V_{t-1}(\tau(s,a), \pi(\tau(s,a)))]$$

Note, in the above, that the maximum is taken over policies π restricted to the constrained policy class Π.

We now wish to establish the convergence of the value function estimates by the constrained value iteration procedure described above, subject to certain technical conditions. Specifically, following the intuitive discussion of our premise that the state distribution governing the constraints on the policies is stable, we introduce the following technical condition.

Definition 1. *We say that a state distribution μ and an MDP with transition probability function τ satisfy the strong stationarity condition, if both of the following hold.*

1. *Taking any action $a \in A$ in a state $s \in S$ drawn according to μ and transitioning to the next state $\tau(s,a)$ will not alter the resource constraints, namely,*

$$\Pi_{\tau(\mu(\cdot),a)} = \Pi_\mu$$

2. *The value functions of policies within Π_μ will remain unchanged by the same change of distribution:*

$$\forall V \in \Pi_\mu \; \forall a, a' \in A,$$

$$E_{\tau(\mu(\cdot),a)}[V(s,a')] = E_\mu[V(s,a')]$$

Although these are idealized assumptions that are unlikely to be satisfied exactly, it is reasonable to suppose that they will be approximately satisfied. For simplicity, in much of the theoretical developments to follow, we will assume that they are satisfied exactly.

THEOREM 3.1. *Suppose that the MDP and the distribution μ satisfy the strong stationarity condition. Then for any linearly constrained policy class Π_μ, the value function estimates output by the constrained value iteration procedure converge to the value function for the optimal policy within Π.*

PROOF. The proof essentially relies on the "Banach Fixed-Point Theorem" (Theorem 6.2.3. in [15]), which is restated below.

THEOREM 3.2 ([15]). *Suppose V is a complete linear space (a.k.a. Banach space), and $T : \mathcal{V} \to \mathcal{V}$ is a contraction mapping, namely, there exists λ, $0 \le \lambda < 1$ such that*

$$\forall U, V \in \mathcal{V},\ ||T(V) - T(U)|| \le \lambda ||V - U||$$

Then there exists a unique $V^* \in \mathcal{V}$ such that $T(V^*) = V^*$, and for arbitrary $V_0 \in \mathcal{V}$, the sequence $\{V_n = T(V_{n-1}) = T^n(V_0)\}$ converges to V^*.

For any constrained class of policies Π, we define the associated constrained value iteration mapping, L_Π, mapping $\mathcal{V}$ to $\mathcal{V}$, as follows.

$$\forall s \in S, \forall a \in A,$$
$$L_\Pi(V)(s,a) = E[R(s,a) + \gamma V(\tau(s,a), \pi_V^*(\tau(s,a)))]$$

where π_V^* is the "optimal" policy, namely that which maximizes the right hand side of the above equation in the policy space Π with respect to the state distribution μ:

$$\pi_V^* = \arg\max_{\pi \in \Pi} E_\mu[R(s,a) + \gamma V(\tau(s,a), \pi_V(\tau(s,a)))]$$

Now we define the *expected sup norm* on the space of value functions by taking the expectation over the state space with respect to a fixed distribution μ, of the sup norm over the actions, i.e.,

$$||U - V||_\mu = E_{s\sim\mu}[\max_{a\in A} ||U(s,a) - V(s,a)||]$$

Here the intention is that the stationary empirical distribution will be used as μ. It is easy to see that this is indeed a norm, since it is the expectation of a norm. Then we have the following key lemma.

LEMMA 3.1. *Suppose that the MDP and a state distribution μ satisfy the strong stationarity condition. Then the constrained value iteration map L_Π is a contraction map with respect to the normed space of value functions, with the expected sup norm with μ.*

PROOF. We begin by establishing the following claim.

Claim 1. *For any state distribution μ, for all value functions $U, V \in \mathcal{V}$, and for a set of permissible policies Π, with respect to which π_U^* and π_V^* are defined, we have*

$$E_\mu[||V(s, \pi_V^*(s)) - U(s, \pi_U^*(s))||] \le ||V - U||_\mu$$

Proof of Claim 1

Suppose the claim does not hold. Without loss of generality, we assume that both of the following hold.

$$||V - U||_\mu = \epsilon$$

$$E_{s\sim\mu}[V(s, \pi_V^*(s)) - U(s, \pi_U^*(s))] > \epsilon$$

Then, the latter inequality implies:

$$E_{s\sim\mu}[V(s, \pi_V^*(s))] - \epsilon > E_{s\sim\mu}[U(s, \pi_U^*(s))]$$

But since $||V - U||_\mu = \epsilon$ we must also have

$$E_{s\sim\mu}[U(s, \pi_V^*(s))] \ge E_{s\sim\mu}[V(s, \pi_V^*(s))] - \epsilon$$

Putting together the last two inequalities, we have

$$E_{s\sim\mu}[U(s, \pi_V^*(s))] > E_{s\sim\mu}[U(s, \pi_U^*(s))]$$

But by the "optimality" of π_U^* for U,

$$E_{s\sim\mu}[U(\tau(s,a), \pi_V^*(\tau(s,a)))] \le E_{s\sim\mu}[U(\tau(s,a), \pi_U^*(\tau(s,a)))]$$

and by the strong stationarity condition 2, we have

$$E_{s\sim\mu}[U(s, \pi_V^*(s))] \le E_{s\sim\mu}[U(s, \pi_U^*(s))]$$

leading to a contradiction. This completes the proof of Claim 1.

Now recalling that

$$L_\Pi(V)(s,a) = E[R(s,a)] + \gamma E[V(\tau(s,a), \pi_V^*(\tau(s,a)))]$$

we have, for any $a \in A$,

$$\begin{aligned}
& E_{s\sim\mu}[L_\Pi(V)(s,a) - L_\Pi(U)(s,a)] \\
= \ & \gamma E_{\mu,\tau}[V(\tau(s,a), \pi_V^*(\tau(s,a)))] \\
& -\gamma E_{\mu,\tau}[U(\tau(s,a), \pi_U^*(\tau(s,a)))] \\
= \ & \gamma E_{\mu,\tau}[V(\tau(s,a), \pi_V^*(\tau(s,a))) \\
& -U(\tau(s,a), \pi_U^*(\tau(s,a)))] \\
\le \ & \gamma E_{\mu,\tau}[||V - U||] \\
= \ & \gamma E_\mu[||V - U||]
\end{aligned}$$

The second to last inequality is obtained by applying Claim 1 on $\Pi_{\tau(\mu(\cdot),a)}$, which is possible since $\Pi_{\tau(\mu(\cdot),a)}$ is equal to Π_μ by the first strong stationarity condition (Definition 1.1), and the last inequality follows by the second condition (Definition 1.2). Hence,

$$||L_\Pi(V) - L_\Pi(U)||_\mu \le \gamma ||V - U||_\mu$$

□

Now it is left only to establish that the space of value functions for a set of policies with linear constraints is a Banach space, with respect to the expected sup norm.

LEMMA 3.2. *The normed space of value functions for the set of linearly constrained stochastic policies with the expected sup norm is a linear complete space.*

PROOF. It is easy to see that the normed space of value functions is linear irrespective of whether there is a constraint, since for any value function V and a scalar α, $||\alpha V|| = \alpha ||V||$ holds. Now for showing completeness, recall that a set of stochastic policies with linear constraints is defined as the set of stochastic policies $\pi : S \times A \to [0,1]$ satisfying a set of linear constraints. So Π is a simplex in $[0,1]^{S\times A}$ and hence is compact. Consider the mapping $\Phi : \Pi \to V$, mapping any policy π to its value function V_π. Then it is easy to see that Φ is continuous, since any incremental change in π, say by ϵ, will translate to a bounded change in the corresponding value function with respect to the expected sup norm, specifically by $R_{max}\epsilon$, where R_{max} is an upper bound on the expected reward for any state-action pair. Since the image of a compact set under a continuous function is also compact, it follows that the set of value functions for stochastic policies in a linear constrained policy space is compact, and hence complete. □

3.3 Constrained Q-Learning

Loosely based on the constrained value iteration procedure described in the previous section, we can derive constrained versions for many known reinforcement learning methods. Of these, we specifically consider the constrained version of Q-learning.

The constrained Q-learning algorithm is essentially expressed by the following update equation, for each observed state, action, reward and the next state sequence (s, a, r, s'):

$$\begin{aligned} Q_k(s,a) &= (1-\alpha_k)Q_{k-1}(s,a) \\ &\quad +\alpha_k E_{\pi_k^*}[r+\gamma Q_{k-1}(s',\pi_k^*(s'))] \end{aligned} \quad (1)$$

where

$$\pi_k^* = \arg\max_{\pi\in\Pi} E_\pi[r+\gamma Q_{k-1}(s',\pi(s'))]$$

and k denotes the number of times the given state-action pair (s, a) has been observed and updated. Note that Q-value function is nothing but the two-place value function we introduced in the previous section, but here we denote it as "Q", following the convention in describing Q-learning.

For this formulation of the constrained Q-learning method, we can establish the following convergence property, by a straightforward application of the contraction property of the corresponding constrained value iteration operator (Lemma 1) to a theorem in stochastic approximation.

THEOREM 3.3. *Suppose that the MDP and the distribution μ satisfy the strong stationarity conditions. Then the Q-value function estimates, Q_k, output by the constrained Q-learning algorithm converge to the Q-value function of the optimal policy in the constrained class of policies Π_μ with probability 1, provided (1) S and A are finite; (2) $\sum_{k=1}^\infty \alpha_k = \infty$ and $\sum_{k=1}^\infty \alpha_k^2 < \infty$; and (3) $Var[\rho(s,a)] < \infty$.*

PROOF. The proof relies on the following theorem, which is an application of stochastic approximation to stochastic DP algorithms, due to Jaakkola et al. [9], which is re-stated here in a slightly simplified and weaker form.

THEOREM 3.4 (JAAKKOLA ET AL [9]). *A random process defined by $\Delta_{n+1}(x) = (1-\alpha_n(x))\Delta_n(x) + \alpha_n(x)F_n(x)$ converges to zero with probability 1, provided the following assumptions are satisfied. (1) The state space is finite. (2) $\sum_{k=1}^\infty \alpha_k(x) = \infty$ and $\sum_{k=1}^\infty \alpha_k(x)^2 < \infty$ uniformly over x with probability 1; (3) $E[||F_n(x)||_W] \le \gamma||\Delta_n||_W$; and (4) $Var[F_n(x)] \le C(1+||\Delta_n(x)||)^2$ for some constant C. Here $||\cdot||_W$ denotes a weighted maximum norm.*

First we define

$$\Delta_k(s,a) = Q_k(s,a) - Q^*(s,a) \quad (2)$$

$$F_k(s,a) = L_\Pi(Q_k)(s,a) - Q^*(s,a) \quad (3)$$

where we used L_Π to denote the constrained value iteration operator defined earlier, and Q^* to denote the Q-value function corresponding to the optimal policy within Π. Now, if we formulate the constrained Q-learning update as a random process in terms of ρ and τ, the update is expressible in terms of L_Π.

$$\begin{aligned} & Q_{k+1}(s,a) \\ = \; & (1-\alpha_k)Q_k(s,a) \\ & +\alpha_k E_{\pi_k^*}[\rho(s,a)+\gamma Q_k(\tau(s,a),\pi_k^*(\tau(s,a)))] \\ = \; & (1-\alpha_k)Q_k(s,a) + \alpha_k L_\Pi(Q_k)(s,a) \end{aligned}$$

Now subtracting $Q^*(s, a)$ from both sides of the equality, we have

$$\Delta_{k+1}(s,a) = (1-\alpha_k)\Delta_k(s,a) + \alpha_k F_n(s,a) \quad (4)$$

In applying Theorem 3.4 to this random process, it is easy to see that conditions (1), (2) and (4) are implied by the assumptions in Theorem 3.3. As for (3), by taking the expected sup norm $E_{s\sim\mu}||\cdot||$ as our weighted norm $||\cdot||_W$, we have

$$\begin{aligned} ||F_k||_W &= E_{s\sim\mu}\max_{a\in A}[||F_k(s,a)||] \\ &= E_{s\sim\mu}[\max_{a\in A}||L_\Pi(Q_k)(s,a) - Q^*(s,a)||] \\ &= E_{s\sim\mu}[\max_{a\in A}||L_\Pi(Q_k)(s,a) - L_\Pi(Q^*)(s,a)||] \\ &\le \gamma E_{s\sim\mu}[\max_{a\in A}||Q_k(s,a) - Q^*(s,a)||] \\ &\quad \text{by Lemma 3.1} \\ &= \gamma||\Delta_k||_W \end{aligned}$$

□

4. A CONCRETE ALGORITHM

4.1 Constrained Advantage Updating

While the generic description of constrained reinforcement learning methods given in the foregoing section served as a theoretical basis, they require some modifications and extensions to be useful in real world applications. One critical issue is that of dealing with variable time intervals between actions. Among the body of past works that addressed the problem of extending Q-learning and other related learning methods to variable time intervals and continuous time setting [4, 6], the *Advantage Updating* algorithm, due to Baird [4], is particularly attractive and has proven effective in past applications [1].

Advantage updating is based on the notion of *advantage* of an action a relative to the optimal action at a given state s, written $A(s, a)$:

$$A(s,a) = \frac{1}{\Delta t_s}(Q(s,a) - \max_{a'} Q(s,a')) \quad (5)$$

In the above, Δt_s denotes the time interval between the state s and the subsequent one. The notion of advantage is useful because it factors out the dependence of the value function on the time interval (by division by Δt_s), and relativizes the influence of the state (by subtraction of $\max_{a'} Q(s, a')$).

Given this notion of advantage, *advantage updating* is an on-line learning method that learns this function iteratively, by a coupled set of update rules for the estimates of A and V, and a normalization step for $A^*(s, a)$ which drives $\max_{a'} A^*(s, a')$ towards zero. Although superficially it differs from the canonical Q-learning method, its central step still involves choosing an action that maximizes the A-value estimate. So, given the standard version of this algorithm, its *constrained* version can be derived in a straightforward manner by replacing the maximization by the appropriate constrained optimization. We present pseudo-code for the constrained (and batch) version of this algorithm in Figure 1.

4.2 Coupling constrained optimization with linear modeling

In a typical real world application, such as targeted marketing, the state space is represented by a feature space involving tens, if not more, of features. (See Section 5.) It is therefore practical to use *function approximation* in the estimation involved in batch reinforcement learning (c.f. [18, 21]). This corresponds to the use of a base regression method (Base) in the description of constrained batch advantage updating procedure in Figure 1.

```
Procedure Constrained Advantage Updating
Premise:
  A base learning module, Base, for regression is given.
Input data: $D = \{e_i | i = 1, ..., N\}$ where
    $e_i = \{\langle s_{i,j}, a_{i,j}, r_{i,j}, t_{i,j} \rangle | j = 1, ..., l_i\}$
    ($e_i$ is the $i$-th episode, and $l_i$ is the length of $e_i$.)
  1. For all $e_i \in D$
    1.1 For $j = 1$ to $l_i$, $\Delta t_{i,j} = t_{i,j+1} - t_{i,j}$
  2. For all $e_i \in D$
    $D_i^{(0)} = \{\langle (s_{i,j}, a_{i,j}), \frac{r_{i,j}}{\Delta t_{i,j}} \rangle | j = 1, ..., l_i\}$
  3. $A^{(0)} = \mathrm{Base}(\bigcup_{i=1,...,N} D_i^{(0)})$
  4. For all $e_i \in D$ and for $j = 1$ to $l_i - 1$, initialize
    4.1 $A_{i,j}^{(0)} = A^{(0)}(s_{i,j}, a_{i,j})$
    4.2 $\pi^*_{(0)} = \arg\max_{\pi \in \Pi} \sum_{i,j} A^{(0)}(s_{i,j}, \pi(s_{i,j}))$
    4.3 $Aopt_{i,j}^{(0)} = A^{(0)}(s_{i,j}, \pi^*_{(0)}(s_{i,j}))$
    4.4 $V_{i,j}^{(0)} = Aopt_{i,j}^{(0)}$
  5. For $k = 1$ to $K$
    5.1 Set $\alpha_k$, $\beta_k$ and $\omega_k$, e.g. $\alpha_k = \beta_k = \omega_k = \frac{1}{k}$
    5.2 For all $e_i \in D$
      For $j = 1$ to $l_i - 1$
        $A_{i,j}^{(k)} = (1 - \alpha_k) A_{i,j}^{(k-1)}$
            $+ \alpha_k (Aopt_{i,j}^{(k-1)} + \frac{r_{i,j} + \gamma^{\Delta t_{i,j}} V_{i,j+1}^{(k-1)} - V_{i,j}^{(k-1)}}{\Delta t_{i,j}})$
        $D_i^{(k)} = \{\langle (s_{i,j}, a_{i,j}), A_{i,j}^{(k)} \rangle | j = 1, ..., l_i - 1\}$
    5.3 $A^{(k)} = \mathrm{Base}(\bigcup_{i=1,...,N} D_i^{(k)})$
    5.4 For all $e_i \in D$ and for $j = 1$ to $l_i - 1$, update
      $A_{i,j}^{(k)} = A^{(k)}(s_{i,j}, a_{i,j})$
      $\pi^*_{(k)} = \arg\max_{\pi \in \Pi} \sum_{i,j} A^{(k)}(s_{i,j}, \pi(s_{i,j}))$
      $Aopt_{i,j}^{(k)} = A^{(k)}(s_{i,j}, \pi^*_{(k)}(s_{i,j}))$
      $V_{i,j}^{(k)} = (1 - \beta_k) V_{i,j}^{(k-1)}$
          $+ \beta_k (\frac{Aopt_{i,j}^{(k)} - Aopt_{i,j}^{(k-1)}}{\alpha_k} + V_{i,j}^{(k-1)})$
    5.5 For all $e_i \in D$ and for $j = 1$ to $l_i - 1$, normalize
      $A_{i,j}^{(k)} = (1 - \omega_k) A_{i,j}^{(k)} + \omega_k (A_{i,j}^{(k)} - Aopt_{i,j}^{(k)})$
  6. Output the final advantage model, $A^{(K)}$.
```

Figure 1: Constrained reinforcement learning based on advantage updating.

The use of a segmented linear regression algorithm (e.g. [3], [12]) for function approximation in the present context leads to a practically viable method. In the case of the constrained advantage updating algorithm, the advantage values, A, are estimated using a segmented linear regression model. That is, the A-model consists of a finite number of segments, each defined by a conjunctive condition on the features, and a linear regression model for that segment. Using this model, the constrained optimization procedure within blocks 4.2 and 5.4 in the algorithm description in Figure 1 can be formulated as follows.

Let $D = \{(s, a, r)\}$ denote the state-action-reward triples in the input data. Let *seg* denote the segmentation function of the model, mapping states to their segments. Let X denote the set of segments in the model. For each segment $x \in X$, the advantage function is estimated as a linear regression of the following form (denoted R to avoid confusion with the set of actions A.)

$$R = \sum_{a \in A} R(x, a) \cdot M(x, a)$$

where $R(x, a)$ is the linear coefficient for action a, and $M(x, a)$ is the number of action a allocated to segment x. Then the objective of optimization is to maximize

$$\sum_{x \in X} \sum_{a \in A} R(x, a) \cdot M(x, a)$$

subject to the following constraints (6), (7) on the resources associated with the actions, namely,

$$\sum_{(s,a,r) \in D} \sum_{a' \in A} C(seg(s), a') \cdot M(seg(s), a') \leq B \qquad (6)$$

where $C(seg(s), a')$ denotes the cost of allocating a single action a' to segment $seg(s)$ and B is a resource bound, and

$$\forall x \in X, \sum_{seg(s)=x} \sum_{a \in A} M(seg(s), a) = \sharp(seg(s)) \qquad (7)$$

where $\sharp(seg(s))$ denotes the size of the segment $seg(s)$. Note that the above is an equality because we consider *inaction* as one of the actions. Given a solution to this optimization problem, namely the action allocations M to each segment-action pair, one can define the corresponding stochastic policy π^* as follows:

$$\forall s \in S, \forall a \in A, \pi^*(s, a) = \frac{M(seg(s), a)}{\sharp(seg(s))}$$

Thus, the linear regression formulation naturally reduces the optimization problem to a linear program, resulting in a practically viable algorithm that realizes constrained reinforcement learning. In our experiments, we used ProbE: IBM's segmented linear regression engine [3, 12] as our function approximator.

5. EXPERIMENTS

We applied our constrained reinforcement learning framework, instantiated with the concrete algorithm described in the last section, and evaluated its performance on two real world problems. These data sets are briefly described below.

5.1 KDD Donation Data

For our first domain, we use the well-known donation data set from KDD Cup 1998, which is available from the UCI KDD repository [5]. This data set contains information from direct mail promotions soliciting donations, and contains demographic data as well as promotion history of 22 monthly campaigns, conducted over a two year period.

For each campaign, the data records whether an individual was mailed a solicitation for a donation, and whether she responded with a donation. The data also records the amount and date of a donation if one was made. We used a randomly sampled subset of 10 thousand individuals from the training data portion of the original data set, which contains data for approximately 100 thousand selected individuals.[1] We preprocessed this data in the same way as done in [13]. In particular, we only select the age and income bracket from the set of demographic features provided. We also generated a number of temporal features to capture the *state* of an individual at the time of each campaign, e.g., the number of donations in the last 6 months, frequency of donations,

[1]This is contained in "cup98lrn.zip" on the URL "http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html".

Table 1: Features in our data based on the KDD Cup 98 donation data

Features	Descriptions
age	indivual's age
income	income bracket
nGiftAll	number of gifts to date
numProm	number of promotions to date
frequency	nGiftAll / numProm
recency	number of months since last gift
lastGift	amount in dollars of last gift
rAmntAll	total amount of gifts to date
nRecProms	num of recent promotions (last 6 months)
nRecGifts	num of recent gifts (last 6 months)
totRecAmt	total amount of recent gifts (6 months)
recAmtPerGift	recent gift amount per gift (6 months)
recAmtPerProm	recent gift amount per prom (6 months)
promRecency	num of months since last promotion
timeLag	num of months between first prom and gift
recencyRatio	recency / timeLag
promRecRatio	promRecency / timeLag
action	whether mailed in current promotion (0,1)
inaction	whether not mailed in current promotion

etc. A complete list of the features we use for modeling is presented in Table 1.

In addition to the mailing *action*, we model the effect of not taking any action — we capture this by explicitly adding *inaction* as one of the possible actions in the data.

5.2 Saks Marketing Data

The other data set we use for our evaluation is the (proprietary) marketing data from Saks Fifth Avenue, which was used in a prior work [1]. We refer the reader to that reference for further details, but for completeness sake we will briefly summarize this data set here, and describe the enhancements we made in order to test the additional aspects of resource optimization we address in this paper.

The data set was generated by joining together a variety of data sources, consisting of customer data, transaction data, marketing campaign data, and product data. Using these, we generated time stamped sequences of feature vectors containing summarized information on marketing and transaction history of the customers. We used a random sample of 5,000 customers out of 1.6 million customers in the original data set, and generated a sequence of 68 states for each of them, corresponding to 68 marketing campaigns, totaling 340,000 data records, half of which was used for training and the other half reserved for evaluation.

The features can be categorized into four types: customer features, transaction features, campaign features, product group specific campaign features. The customer features are simply a subset of the customer information in the customer data. The transaction features were calculated by joining the store transaction history (p.o.s.) data with the customer data. The campaign features were constructed using the per campaign mailing lists, again joining them with the customer data. The product group specific campaign features were synthesized using both campaign data and taxonomy information in the product data. As in the KDD Cup data we also explicitly represent *action* and *inaction* in the data.

5.3 Resource Constraints

The KDD Cup and Saks data sets do not explicitly provide information on constraints on actions. However, in both cases, there are naturally limits on the number of mailings/marketing actions that can be executed. In particular, we introduce the notion of departments (or groups); where each individual (customer) in the data is assigned to a department based on her cumulative value to date. In the case of the KDD Cup data the value of an individual is defined as the sum of the donations made so far, upto the date in question. Note therefore that a given customer can move from one department to another in the observed period. For the Saks data, the value of a customer is the sum of the cumulative revenue i.e., the sum of purchases, minus the merchandise and marketing costs Resource constraints are then defined as bounds on the number of actions that can be performed by each department per unit time. Table 2 presents the department definitions for the KDD Cup data, along with the number of instances in the training data assigned to each department, and the average reward generated by customers in each department. The last column shows the amount of resource bounds we assigned to each department in our experiments. Assuming that each action takes unit time to execute, we represent the resources in terms of the number of actions permissible by each department. We assume inaction does not consume any resources. The resource bounds were selected to be approximately proportional to the department sizes and limit the number of actions that can be taken to roughly 3 out of 8 customers. Table 3 presents the analogous information for the Saks data.

Table 2: Summary of resources for KDD Cup data

Dept. def. in terms of cumul. reward (r)	Size	Average reward($)	Resource bound
$r < 10$	45,226	2.05	1,700
$10 \leq r < 40$	37,068	0.53	1,200
$r \geq 40$	3,962	1.00	150

Table 3: Summary of resources for Saks data

Dept. def. in terms of cumul. reward (r)	Size	Average reward($)	Resource bound
$r < 10$	114,807	3.87	11,480
$10 \leq r < 250$	28,605	6.89	2,860
$r \geq 250$	26,588	33.38	2,660

5.4 Evaluation Methodology

A challenge in evaluating the performance of a data-driven business optimization methodology is that we are typically required to do so using only historical data, which was presumably collected using a different (sampling) policy. Here we employ an evaluation method based on bias correction that allows us to do so, essentially following [1].

A useful quantity to estimate is the expected R-value for a new policy π' with respect to an old (or sampling) policy π and state distribution μ, written $R_{\pi,\mu}(\pi')$ and defined as follows.

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

We can estimate the above quantity using the sampling policy, with appropriate *bias correction* (c.f. [23]), in which the observed reward is multiplied by the ratio between the probabilities assigned to the observed action by the respective policies[2]

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

Note, in the above, that $\pi'(a|s)$ is known since it is the stochastic policy generated by the constrained reinforcement learning procedure, but $\pi(a|s)$ needs to be estimated from the data, since we do not know the sampling policy explicitly. In our evaluation, we estimate π using Laplace estimates in each of the segments that were output by the segmented linear regression algorithm used for function approximation in the learning procedure.

5.5 Experimental Results

A natural alternative to our constrained reinforcement learning approach is to use a standard reinforcement learning algorithm to estimate the values of competing actions, and to use optimization only at the time of application to select the actions to maximize expected reward given the resource constraints. Another more straightforward alternative is to use a standard regression modeing to estimate (possibly observed cumulative) expected rewards for each state, action pair, and then apply constrained optimization using the estimated model, in place of those obtained by reinforcement learning, constrianed or unconstrained. We evaluate thse approaches, along with the proposed constrained reinforcement learning approach and investigate their relative merits.

Our data derived from the KDD Cup data consists of 10,000 episodes, each consisting of 16 states, which we divide into two equally-sized sets — one for training and one for validation. This translates to a training data size of 80,000 state-action-reward tuples for reinforcement learning. Since the evaluation is stochastic in nature, we average the results over 10 evaluation runs, using a different random number seed for each run.

For the KDD Cup data, Figure 2 compares the expected reward obtained by each method, plotted as a function of the number of learning iterations performed. What is plotted in the graph is actually the percentage improvement over the expected reward of the sampling policy in the data. We observe a steady improvement in performance for consecutive iterations of both algorithms. Furthermore, we see that constrained reinforcement learning begins to outperform unconstrained reinforcement learning in later iterations. The difference in expected rewards for the final model, at approximately 4 % of the sampling policy's rewards, is indeed statistically significant based on a paired t-test ($p < 10^{-7}$).

During training, unconstrained reinforcement learning does not account for constraints on resources, which are only enforced during application of the learned policy. As such, even though unconstrained reinforcement learning learns to effectively estimate the value of competing actions, it leads to a myopic policy, that is unable to foresee the infeasibility of some actions in the future due to limits on resources. This shortcoming is clearly overcome in constrained reinforcement learning by evaluating the expected cumulative reward with the resource constraints taken into account.

Analogously, Figure 3 exhibits the percentage improvement in expected reward by the two approaches for the Saks data. It is clear that both of these approaches out-perform the straightforward approach of combining data modeling and constrained optimization, which happens to be approximately equal to the sampling policy. It is also seen that the constrained reinforcement learning has a significant advantage over the unconstrained one. The differences in performance by constrained reinforcement learning and unconstrained reinforcement learning, at iterations 3, 4, and 5, are very statistically significant based on a paired t-test ($p < 10^{-8}$). We also note that the resource constraints used in our experimentation on the Saks data are close to what is observed in the data. It is therefore fair to say that our method would likely translate to a significant lift in revenue, if deployed.

[2]In our empirical evaluation, we actually extend this formulation by choosing to look ahead a number of steps (2 steps to be precise) following the action in question.

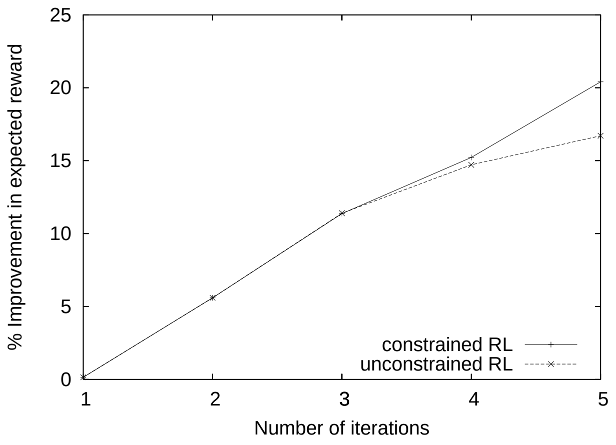

Figure 2: Expected rewards achieved by the two approaches: Percentage improvement over the sampling policy in expected rewards for each method is plotted for five learning iterations.

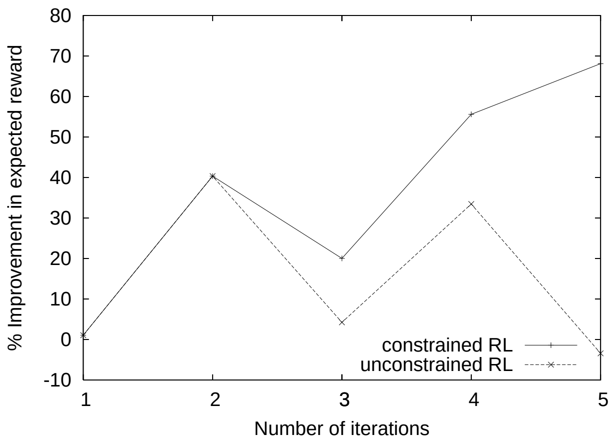

Figure 3: Expected rewards achieved by the two approaches: Percentage improvement over the sampling policy in expected rewards for each method is plotted for five learning iterations.

6. CONCLUDING REMARKS

Some important open problems that we wish to address in the future include further characterization of conditions under which the proposed constrained reinforcement learning strategy significantly outperforms alternative approaches. Relaxing the conditions on the convergence of these methods is also an important theoretical challenge.

Acknowledgments

We would like to thank the following people for their contributions to this work in a variety of ways: Rick Lawrence, Edwin Penault, Chid Apte, Andre Elisseeff, Bianca Zadrozny and Naval Verma.

7. REFERENCES

[1] N. Abe, N. Verma, C. Apte, and R. Schroko. Cross channel optimized marketing by reinforcement learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2004.

[2] Eitan Altman. Asymptotic properties of constrained markov decision processes. Technical Report RR-1598, INRIA, 1993.

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

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

[5] S. D. Bay. UCI KDD archive. Department of Information and Computer Sciences, University of California, Irvine, 2000. `http://kdd.ics.uci.edu/`.

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

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

[8] C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, August 2001.

[9] T. Jaakkola, M. Jordan, and S. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201, 1994.

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

[11] D. Margineantu. On class probability estimates and cost-sensitive evaluation of classifiers. In *Workshop Notes, Workshop on Cost-Sensitive Learning, International Conference on Machine Learning*, June 2000.

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

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

[14] Foster Provost, Prem Melville, and Maytal Saar-Tsechansky. Data acquisition and cost-effective predictive modeling: Targeting offers for electronic commerce. In *Proceedings of the Ninth International Conference on Electronic Commerce*, 2007.

[15] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and sons, Inc., 1994.

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

[17] J. N. Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine Learning*, 16:185–202, 1994.

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

[19] P. Turney. Cost-sensitive learning bibliography. Institute for Information Technology, National Research Council, Ottawa, Canada, 2000. `http://extractor.iit.nrc.ca/bibliographies/cost-sensitive.html`.

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

[21] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[22] Q. Yang and H. Cheng. Planning for marketing campaigns. In *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003)*, pages 174–184. AAAI, 2003.

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

[24] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.