Optimizing Debt Collections Using Constrained Reinforcement Learning

Naoki Abe
Prem Melville
Cezar Pendus
Chandan K. Reddy[*]
David L. Jensen
IBM Research
Yorktown Heights, NY 10598

Vince P. Thomas
James J. Bennett
Gary F. Anderson
Brent R. Cooley
Global Business Services
IBM Corporation
Armonk, NY 10504

Melissa Kowalczyk
Mark Domick
Timothy Gardinier
State of New York
Dept. of Tax. and Fin.
W. A. Harriman Campus
Albany, NY 12227

ABSTRACT

In this paper, we propose and develop a novel approach to the problem of optimally managing the tax, and more generally debt, collections processes at financial institutions. Our approach is based on the framework of constrained Markov Decision Process (MDP), and is unique in the way it tightly couples data modeling and optimization techniques. We report on our experience in an actual deployment of a tax collections optimization system based on the proposed approach, at New York State Department of Taxation and Finance. We also validate the competitive advantage of the proposed methodology using other data sets in a related application domain.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms

Keywords

Constrained Markov Decision Process, Reinforcement Learning, Debt collection, Business analytics and optimization

1. INTRODUCTION

The problem of optimally managing the collections process by taxation authorities is one of prime importance, not only for the revenue it brings but also as a means to administer a fair taxing system. The analogous problem of debt collections management in the private sector, such as banks and credit card companies, is also increasingly gaining attention. With the recent successes in the applications of data analytics and optimization to various business areas, the question arises to what extent such collections processes can be improved by use of leading edge data modeling and optimization techniques. In this paper, we propose and develop a novel approach to this problem based on the framework of constrained Markov Decision Process (MDP), and report on our experience in an actual deployment of a tax collections optimization system at New York State Department of Taxation and Finance (NYS DTF).

The tax/debt collections process is complex in nature and its optimal management will need to take into account a variety of considerations. At a high level, the collections process management problem is that of determining the answers to the following questions: (1) which of the debtors should be approached; (2) which of the possible collection actions are to be taken onto them; (3) who should take those actions; and (4) when they should be taken. The answer to each of these questions will depend on a number of factors. The answer to the first question will depend on the information the collection agency may have on the debtors, such as the demographics and the amount of debt owed. The answer to the second will also depend on the nature and status of the debtor, such as how collectible they appear to be, since actions of varying severity may be appropriate for debtors in different categories. The third will additionally depend on what resources are available within the collection organization. With regard to the fourth, there are several complications that impact the optimal timing to take actions on debtors. For example, in the tax collection case, there are certain legal requirements that govern the sequential course of collection actions. The prime example is the requirement that a warrant need be issued before collection actions such as levy and seizure are performed. Also, there are business considerations that affect the appropriate timing of a collections action in general.

Due to the high complexity involved in the collection process, and particularly because of the legal and business constraints, it is common practice to follow rigid manually constructed rules to guide the collection activities. Even in state-of-the-art rule based systems for collections, the role of data analytics is typically limited to that of augmenting the rules engine with scoring functions that they can refer to. In the present work, we develop a collection process au-

[*]This author's current affiliation: Department of Computer Science, Wayne State University, Detroit, MI 48202.

tomation system that is *primarily* based on data modeling and optimization, and accepts input from a rules engine as constraints on the optimization process.

Specifically, we propose a novel approach based on the framework of constrained Markov Decision Processes (MDP), which addresses all of the considerations that were touched upon in the foregoing discussion. The MDP formulation is particularly applicable here because of its treatment of the sequential dependencies between actions via the introduction of states. For example, the legal requirement that a warrant precedes levy and other collection actions can be addressed naturally by introduction of states that correspond to warranted cases and the unwarranted cases. The value of issuing a warrant in an unwarranted state can then be fairly assessed by considering the long term rewards obtained in future (warranted) states, via subsequent collection actions such as levy. The use of long term rewards as the objective employed in MDP is therefore essential for satisfactory formulation of the collection process.

The constrained MDP formulation, beyond the standard MDP, allows us to take into account the various constraints that govern the actions under consideration. For example, the value of a warrant now will likely depend on the available resources for performing subsequent corresponding actions such as levy. Estimating the value of an action in a constrained MDP involves looking ahead into the resources available at the future states, and the resulting value function and corresponding optimized policy will be better guided with respect to resource consumptions. In addition to the resource constraints, the business and legal constraints can also be handled by including them as an additional type of constraints that define the set of legal policies in the constrained MDP.

As described above, the proposed approach based on the constrained MDP framework provides a comprehensive solution to the problem of collections optimization, which tightly couples data modeling and constrained optimization in a unified manner. Furthermore, it can accept input from rules engine as constraints, and the output policy can be fed into a rules engine. The approach thus allows us to invert the typical roles played by the rules engine and analytics/optimization engine, and provide a nearly automated system in which rules are fed as input to and output from the analytics/optimization engine.

The remainder of the paper is organized as follows. We first review some related work in the intersection of data modeling and decision making. We will then describe the constrained Markov Decision Process and Reinforcement Learning framework we employ in our approach. Next we present the concrete algorithm we employ in our work, and also elaborate on the constrained optimization problem formulation. We then describe the business problem we solve using this approach in the actual deployment at NYS DTF and give some evaluation results. Finally, we present further empirical evaluation of the proposed methodology, using data sets from a different but related application domain.

2. RELATED WORK

The technical problem addressed in the present work is that of tightly integrating data modeling and decision making/optimization. When estimated models are used as input to a subsequent decision making process, it is sometimes desirable that the constraints that govern the execution of those decisions be taken into account in the estimation process itself. In the knowledge discovery and data mining literature, this issue has attracted considerable attention, with notable examples being the body of work known as *cost-sensitive learning* [16, 7, 20, 9, 6] and the emerging topic of *economic learning* [12]. 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 the dynamic problem setting, attempts have been made to extend cost-sensitive learning to *sequential cost-sensitive learning*, formulating the problem using the Markov Decision Process (MDP) framework [11]. The cost structure treated in the past works, however, was in terms 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 what we do in the present work.

There exist some past works that investigated constrained versions of Markov Decision Process (e.g. Altman [2]), but our approach differs from this body of work in a key aspect. 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 and known at each learning iteration. This 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].

3. PRELIMINARIES

3.1 Markov Decision Processes

We begin with a brief description of the concepts of Markov Decision Process (MDP) [13, 8, 14]. An MDP consists of the state space, S, the action space A, initial state distribution $\mu : S \rightarrow \mathcal{R}$, and transition probability function $\tau : S \times A \times S \rightarrow [0,1]$ such that $\forall s \in S, \forall a \in A \ \sum_{s' \in S} \tau(s'|s,a) = 1$, and the expected reward function $R : S \times A \rightarrow \mathcal{R}$. 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. 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 $\pi : S \rightarrow A$ determines an agent's behavior in it. In particular, it determines an infinite sequence of state, action, and reward triples, $\langle s_t, a_t, r_t \rangle$, $t = 1, 2, \ldots$, 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 $\pi : S \times A \rightarrow \mathcal{R}$ such that $\forall s \in S, \sum \pi(s,a) = 1$. Analogously to τ above, for a stochastic policy π, we interchangeably 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.

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

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 [18], 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.

3.2 Constrained MDP

A constrained MDP is an MDP, in which the policies under consideration must belong to a set Π of permissible policies. In the standard constrained MDP, [2], Π is determined with respect to a set of bounds on cumulative discounted "costs", defined analogously to the cumulative rewards. That is, Π_μ is a set of stochastic policies $\pi : S \times A \to [0,1]$ adhering to a set of n constraints of the form:

$$E_{s\in\mu,\pi}[\sum_{t=0}^{\infty} \gamma^t C_{i,s,a}\pi(s,a)] \leq B_i, i = 1,...,n$$

where γ is some fixed positive discount factor, and for each i, $C_{i,s,a}$ denotes a cost incurred by taking action a in state s.

The application domain that we target in the present context possesses some characteristics that motivate us to modify the above, standard formulation. We are given a historical data set for a population of individuals with associated states, distributed according to a certain distribution, and are to use this data set to optimize the present day collections resources. Furthermore, this state distribution will likely remain relatively stable, since collection actions taken now will not significantly alter the overall population mixture of taxpayers. At any given point in time, there is a fixed amount of resources, which can be allocated to actions targeting this population, and these constraints need be observed *at all times*, rather than cumulatively through the course of the state trajectory.

These characteristics together motivate a modified formulation of the constrained MDP, which is applicable to the batch, population learning setting we consider. Formally, we assume that the initial state distribution, μ, used in determining the constraints, is in fact the stationary distribution (for policy π) and arrive at the following simplified formulation of the costs:

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

where in general the B' are distinct from the B in the earlier formulation.

With this simplified formulation, the constraints can now be computed at each iteration of batch learning, *provided* we have the knowledge of the distribution μ. Given the above assumption of stability, μ can indeed be reasonably estimated using the sampling distribution. As we will see in the subsequent developments, this simplified formulation motivates a family of algorithms that are relatively straightforward extensions of existing reinforcement learning methods, and that are well suited to the present application scenario.

4. METHODOLOGY

4.1 Constrained Value Iteration and Q-Leanring

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 3. 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, which emphatically is known and computable at each iteration, due to the simplified formulation.

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'):

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

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.

4.2 A Concrete Algorithm: Constrained Advantage Updating

While the generic description of constrained reinforcement learning methods given in the foregoing section serves to motivate a family of methods, 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, 5], 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 $A(s, a)$:

$$A(s,a) = \frac{1}{\Delta t_s}(Q(s,a) - \max_{a'} Q(s,a')) \tag{2}$$

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.

```
**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.

4.3 Coupling constrained optimization with linear modeling

In a typical real world application, such as debt collection, the state space is represented by a feature space involving tens, if not more, of features. It is therefore practical to use *function approximation* in the estimation involved in batch reinforcement learning (c.f. [15]). This corresponds to the use of a base regression method (Base) in the description of constrained batch advantage updating procedure in Figure 1.

The use of a segmented linear regression algorithm (e.g. [3], [10]) 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.)

$$R = \sum_{a \in A} R(x,a) \cdot M(x,a) + R(x,0)$$

where $R(x,a)$ is the regression coefficient for action a, $R(x,0)$ is the intercept, 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 (3), (4) on the resources associated with the actions, namely,

$$\sum_{(s,a,r) \in D} \sum_{a' \in A} C(seg(s), a') \cdot M(seg(s), a') \leq B \qquad (3)$$

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

$$\forall x \in X, \sum_{seg(s)=x} \sum_{a \in A} M(seg(s), a) = \sharp(seg(s)) \qquad (4)$$

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 practical algorithm that realizes constrained reinforcement learning. In our deployment and in evaluation experiments, we used IBM's scalable segmented linear regression engine, ProbE [3, 10], as our function approximator.

5. DEPLOYMENT AT NEW YORK STATE DTF

A tax collection optimization engine based on the approach proposed in the present paper was implemented and deployed as part of a larger collection management solution at New York State Department of Taxation and Finance (NYS DTF). In this section, we describe some aspects of this deployed solution in detail, and discuss technical enhancements we devised in the actual deployment.

5.1 Business Problem Specification

We begin by briefly describing how the collections process works at NYS DTF. The collection process starts on a taxpayer when an upstream process (e.g. auditing) makes the determination that a certain taxpayer has an outstanding debt and an assessment is created on that taxpayer. At that point, the taxpayer is unassigned, but soon following the maturing of the assessment, the taxpayer is sent by default to the call center (CC). The case will belong to CC for a certain period of time, and various contact and collection actions can be taken onto them (e.g. mailing, phone call, warrant, levy, etc) either manually by CC staff or sometimes automatically (for some of the actions). When determination is made that the collection process should be elevated for a given case, for one reason or another, the case is moved to a more specialized organization. These organizations are categorized into those that handle certain specialized types of cases, or the district offices (DO's) that handle elevated cases belonging to specific regions.

Given this overall flow of the collections process, the task of the recommendation engine, based on our modeling and optimization engine, is divided into the modeling phase and the scoring (action allocation) phase. In the scoring phase, it is given as input: (1) a set of collection actions under consideration; (2) a set of modeling feature vectors for taxpayers of interest; (3) a set of constraint feature vectors for the same taxpayers, specifying for each taxpayer which of the actions are permissible (obtained by applying business and legal rules to their profile data); (4) a description of the available resources in terms of man hours in each of the multiple organizations; (5) a list of additional constraints in terms of upper and lower bounds on the type of actions that are performable; it is to output an optimized action allocation, mapping each of the taxpayers in (2) to a recommended action.

In the modeling phase, the modeling/optimization engine is given analogous input as above, except the modeling feature data (2) are historical sequences for a generally distinct set of taxpayers than for action allocation, and it is to output a series of models. In the scoring phase, the best model(s) are chosen and used.

5.2 Actions and Resource Constraints

In the legacy system, the overall collections process is guided via the notion of case assignment. When the assessment is created, the case is initially "unassigned." It will then be assigned to Call Center (CC) when the assessment matures. If the case is subsequently moved to any one of the organizations mentioned above, the case is then considered "org assigned" (ORG). When the collection process is completed on a case, the case is then "closed" (CLO). When all possible means are attempted to no avail, then the case can be "completed" (COM). In our deployment, we respect the notion of case assignment, and consider cases in CC and ORG to be owned by a particular organization. We have accordingly designed the set of actions: We consider direct collection actions that can be taken at one or the other of the assigned organizations, and indirect or movement actions that move cases from CC to one of the other organizations. The set of actions being considered are listed in Table 1. The first and third groups consist of direct actions and the second the movement actions. The third group (i.e. "perform field visit") is special in that it can only be performed in district offices.

Resources are equated to the man hours available in various organizations for performing the collection actions being considered. For each of the actions we consider we have an estimate of the expected amount of time required to perform them from historical data (column 3 of Table 1), and these are used in conjunction with the man hours in organizations to determine the resource constraints. It is important to note that direct actions consume resources from the organization that the case currently is assigned to, whereas indirect actions, in and of themselves, do not consume any resources. They will incur resources in their future organizations, however, which will be taken into account in evaluating the value of a move to action, via the look-ahead mechanism inherent in the constrained MDP framework. Additionally, we pro-

action	description	hours	bound
Collection Actions			
cntct_tp_ml	contact taxpayer by mail	0.01	5000
cntct_tp_phn	contact taxpayer by phone	0.14	2000
crt_wrrnt	create warrant	0.01	5000
crt_ie	create income execution	0.01	10
crt_lvy	create levy	0.09	5000
Movement Actions			
mv_to_do	move to district office	0	5910
mv_to_hivl	move to high value team	0	330
mv_to_cvs	move to collection vendors	0	340
mv_to_ice	move to indiv. case enf.	0	100
Org Specific Action			
prfrm_fld_vst	perform field visit	0.625	5000
No Action			
no_actn	take no action	0	100000

Table 1: Collection actions being considered, with the hours it takes to perform them, and upper bounds on their allocations specified per day.

vide a form of constraints to explicitly control the number of allocations of move-to actions, which we call "move to upper bounds." (Table 1, column 4.) Since there is one unified action *mv_to_do* for the DO's (unlike other specialized organizations), we have an additional type of bounds to be specified per DO. Table 2 lists some examples of the organizations we consider, with their associated resource limits, as well as the move to upper bounds discussed above.

organization	hours	bound
High Value Team	97.5	
Call Center	225	
Collection Vendor Support (CVS)	60	
Individual Case Enforcement (ICE)	82.5	
DO 1	30	45
DO 2	105	150
DO 3	172.5	120
DO 4	67.5	75
etc...		

Table 2: Example resource and move-to action constraints, specified per day.

5.3 Modeling Features

Given data consisting of the taxpayers' background information, their complete history of transactions (payments) and collection actions taken onto them by DTF (contact and collection actions), we generate, for each taxpayer/case, a time stamped sequence of feature vectors at multiple sampling (or evaluation) time steps, to be used as training data in the constrained reinforcement learning procedure. In the deployed system, we used approximately 200 modeling features in all, some concrete examples of which are listed in Table 3.

5.4 Legal and Business Constraints

In addition to the modeling features, the engine makes use of another group of features called *action constraints*, which are binary features specifying, for each of the actions considered, whether or not the action is allowed on the case at that time point, according to the business and legal rules. The generation of these features are done by feeding the modeling feature vectors to the rules engine containing a catalog of approximately 300 business and legal rules that have been carefully constructed by the users. Table 4 shows a small number of example action constraint rules in this catalog.

feature	description
Taxpayer features	
num_non_rstrctd_fin_srcs	num non-restricted financial sources
st_inactv_ind	sales tax inactive indicator
num_bnkrptcy_flngs	number of bankruptcy filings
Liability features	
ttl_liability_blnc	total liability balance
sum_cllct_asmts	sum of collectible assessments
sum_asmts_avail_to_wrrnt	sum assessments available to warrant
Transactional features	
tax_pd_lst_yr	tax paid last year
num_pymnts_snc_lst_actn	num of payments since last action
sum_pymnts	sum of payments to date
sum_pymnts_lst_yr	sum of payments last year
Collections features	
num_opn_pfrctd_wrrnts	number of open perfected warrants
dys_snc_lst_wrrnt_pfrctd	days since last warrant perfected

Table 3: Some example modeling features.

Rule #	Rule contents
Contact rules	
502.12	A collection letter should not be sent to a taxpayer whose mailing address is invalid
2000.1	A contact action should only occur for a taxpayer with at least one open mature assessment
2005.9	A contact by mail must not be made for a taxpayer with an active promise-to-pay 30 days.
Levy rules	
2601	A levy is not allowed for a taxpayer unless the taxpayer has at least one perfected warrant

Table 4: Example action constraint rules.

5.5 Micro-segments and Resource Optimization

We modify the generic optimization problem formulation given in Section 4 to the present scenario involving organizations and action constraints as follows. First, we let *valid action* be the bit vector obtained by concatenating all the action constraint features, for each (case, time stamp) pair. We then introduce the notion of *micro-segment*, which is defined as a quadruple consisting of the modeling segment, the organization owning the case at the time, the DO the case would be sent to, and the valid action vector. We then modify the optimization problem described in Subsection 4.3 by re-defining the $seg(\cdot)$ function to map state features to the corresponding *micro-segment*, rather than the modeling segment, and modify some of the constraints to depend on the micro-segments. Here, micro-segments inherit the coefficients of the corresponding modeling segments.

5.6 Enhancements to the Modeling Engine

In deploying our engine, we made a few extensions and enhancements to the modeling engine. One extension that is worth mentioning has to do with the way the segmentation is done in the modeling engine. The regression tree modeling engine we use (ProbE [3]) performs tree-based automatic segmentation of the feature space into a number of segments, which are uniform with respect to the regression of the objective function and large enough to admit sufficient statistical significance. While automatic segmentation is critical for our purposes, there is a motivation to guide the segmentation process with some coarse segmentation that is derived from the domain knowledge. For example, the stages within the collections process, such as whether the case is in the call center or in a district office, or whether the case has been warranted, have a major impact on the way they should be handled. We do exactly this, by forcing the top level branchings according to a specified set of categorical variables, and then given this initial segmentation, the rest of segmentation is performed automatically. Specifically, we force the initial branching according to a tailored categorical variable we call "state", which is defined by combining the notion of case assignment and collection stage. See Table 5 for the definitions of some example states.

State #	Definition
CCN	Assinged to Call Center and unwarranted
CCW	Assinged to Call Center and warranted
DON	Assinged to district office and unwarranted
DOW	Assinged to district office and warranted
CVS	Assinged to CVS
CLO	Closed
COM	Complete
etc...	

Table 5: The "state" variable for enforced branching.

5.7 Output Segments and Allocations

Some example segments output by the engine are exhibited in Table 6. Some interesting observations can be made. First, notice that the first segment (Segment 212), for which many warrant actions are allocated, includes the condition that *num_non_rstrctd_fin_srcs* (number of non-restricted financial sources) is at least 1. This is in fact one of the conditions for levy to be possible, and the fact that it is included for a segment for which warrant is recommended suggests that the look ahead mechanism of our reinforcement learning method is working properly. Also note that "state" equals "CCN", which means the case has not been warranted. Next, in the second segment (Segment 437) for which mailing is recommended, we see the condition that *tax_pd_lst_yr* (tax paid last year) exceeds a certain threshold amount. A plausible interpretation of this is that this segment consists of taxpayers who have been paying and will likely pay without having to take elevated and costly collection actions. The third segment (Segment 341) is a little more involved. We see that it is relatively high valued (sum_asmts_avail_to_wrrnt exceeds a certain amount), and have paid last year but not recently (tax_pd_lst_yr exeeds a threshold and num_pymnts_snc_lst_actn is less than 1), but a significant debt amount remains and they appear collectible (num_non_rstrctd_fin_srcs is at least 1, st_inactv_ind is 0, and sum_asmts_avail_to_wrrnt exeeds a threshold.) It seems reasonable that the engine recommends move to DO within resource limits and no action to the rest of this segment.

5.8 Further Deployment Details

The architecture of the deployed system is schematically depicted in Figure 2. The overall solution is deployed and executed within a WebSphere process server (WPS) application server. Data sources include the collections data, tax return data, as well as additional external data. They are used

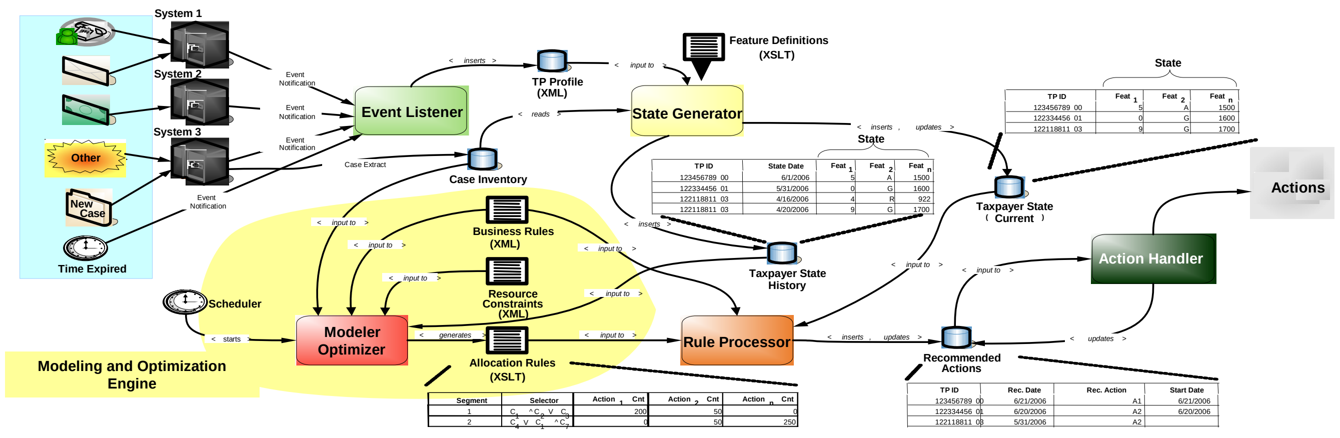
Segment Definition	Action Allocation
Segment 212	
state = CCN and tax_pd_lst_yr < \$ X and 1 <= num_non_rstrctd_fin_srcs and 1 <= st_inactv_ind and num_pymnts_snc_lst_actn < 1 etc...	5103 crt_wrrnt 152 no_actn
Segment 437	
state = DON and \$ Y <= tax_pd_lst_yr and sum_pymnts < \$ Z and \$ S <= sum_pymnts_lst_yr and ttl_liability_blnc < \$ T etc...	1004 cntct_tp_ml 77 mv_to_cvs 141 no_actn
Segment 341	
and state = CCN and \$ X <= tax_pd_lst_yr and num_pymnts_snc_lst_actn < 1 and \$ V <= sum_cllct_asmts and 1 <= num_non_rstrctd_fin_srcs and st_inactv_ind < 1 and \$ W <= sum_asmts_avail_to_wrrnt	201 mv_to_do 1424 no_actn

Table 6: Example output model segments.

to compute summary information on the taxpayers, which are stored as an intermediate representation referred to as *taxpayer profiles*. The taxpayer profile exists as a single xml document for each taxpayer in a DB2 database (indicaed as 'TP Profile' in Figure 2.) The profile records the history of the taxpayer states using the temporal data model approach suggested by Wang and Zaniolo [17], using transactional semantics with a granularity of a day. A simple transform can generate the state of the taxpayer as it was on any particular day.

Numerous legacy systems generate events which are passed to the engine by work-flow. Each event has an XSLT2 transform that updates the taxpayer profile with the new information. Each week, taxpayers that have had an event, and those whose next review becomes due, are considered for scoring (action allocation). They pass through a number of transforms which check their suitability for scoring, apply prescriptive rules and generate the state features, giving rise to the scoring data. An analogous process is used to generate the training data of the constrained MDP procedure, except in the latter case a sequence of state feature vectors per taxpayer are generated, at a number of evaluation points in the past. The two types of state feature data are respectively indicated as 'Taxpayer State (Current)' and 'Taxpayer State (History)' in Figure 2. In total, taxpayer profiles are computed for over 2 million taxpayers, but typically as training data for modeling, we use feature data for a subset of them, in the order of a couple of hundred thousand taxpayers, each with 5 to 10 state feature vectors per taxpayer on the average, totally approximately 1 million data records.

Following the modeling and scoring processes by the engine, a few more steps happen prior to the actual action recommendations, which are performed by the module called 'Action Hanlder' in the figure. One complication of the present domain is that actions on a taxpayer may depend upon the state of other related taxpayers. When a profile indicates a relationship, the related profiles are brought in under an associations tag so that the models and rules can



		State		
TP ID	State Date	Feat $_1$	Feat $_2$	Feat $_n$
123456789 00	6/1/2006	5	A	1500
122334456 01	5/31/2006	0	G	1600
122118811 03	4/16/2006	4	R	922
122118811 03	4/20/2006	9	G	1700

	State		
TP ID	Feat $_1$	Feat $_2$	Feat $_n$
123456789 00	5	A	1500
122334456 01	0	G	1600
122118811 03	9	G	1700

Segment	Selector	Action $_1$ Cnt	Action $_2$ Cnt	Action $_n$ Cnt
1	$C_1 \wedge C_2 \vee C_3$	200	50	0
2	$C_4 \vee C_1 \wedge C_7$	0	50	250

TP ID	Rec. Date	Rec. Action	Start Date
123456789 00	6/21/2006	A1	6/21/2006
122334456 01	6/20/2006	A2	6/20/2006
122118811 03	5/31/2006	A2	

Figure 2: Overall collections system architecture.

see the full picture. This also means that, after a recommendation is generated for each taxpayer, conflicts may need be consolidated with these relationships in mind. For example, it may not be desirable to send related taxpayers to different organizations. This adjustment is part of what is performed by the module named 'Rule Processor' in the figure. Additionally, there is a mechanism in place that can be used to deploy two models in an incumbent/challenger paradigm, until such time as there is sufficient statistical significance to call a winner. The loser will be replaced by the winner or a new model when available.

There are three types of users at NYS DTF. The first type is the members of the IT department within DTF, who are responsible for operating and maintaining the collections optimization system. The second type is the project team, of size ranging from 5 to 15, that represents the users of the collections optimzation system, who perform validation of the rules and output allocations, and oversee the process of setting resource constraints and other inputs to the system. The third is the actual collections agents who are directly affected by the system's recommended actions. Of these, 30 to 40 call center agents receive the recommended actions via an application screen, whereas the hundreds of agents in the specialized organizations get them indirectly via the engine's allocations to the respective organizations.

The engine itself is a result of multiple years of research and development, and with all of its components, the relevant efforts sum to about 5 million dollars in research investment, although some of the components (modeling and optimization engines) are generic. Additionally, NYS DTF invested approximately 4 million dollars for this engagement. The software maintenance planning is underway by IBM Global Business Services, as part of the assetization and commercialization of the engine.

5.9 Lift Evaluation

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 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 cumulative reward for a new policy π', when sampled with respect to the old (or empirical) 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. [19]), in which the observed reward is multiplied by the ratio between the probabilities assigned to the observed action by the respective policies

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

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.

Figire 3 plots the expected rewards of the collections policies output by our engine in a typical modeling run, as compared to the sampling or DTF's historical policy, as a function of the number of learning iterations. The advantage over the sampling policy is significant, and the models obtained over the learning iterations exhibit a steady improvement.

6. EVALUATION ON OTHER DATA SETS

In order to assess the robustness of our proposed methodology, we applied our proposed method and evaluated its performance on two other real world data sets.

6.1 The Data Sets

The first data set we use is based on the well-known donation data set from KDD Cup 1998, which is available from the UCI KDD repository.[1] This data set contains information from direct mail promotions soliciting donations, and

[1]`http://kdd.ics.uci.edu/`

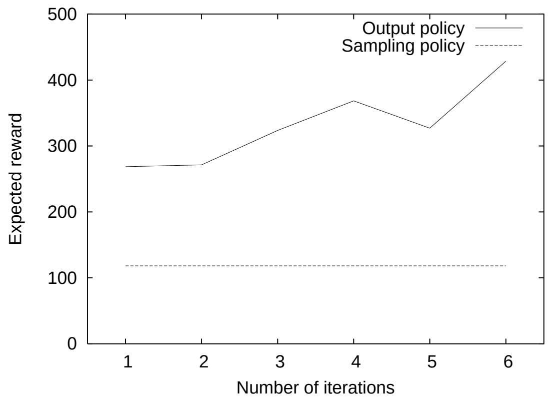


Figure 3: Normalized expected rewards achieved by the deployed method and the sampling policy are plotted for six learning iterations.

contains demographic data as well as promotion history of 22 monthly campaigns, conducted over a two year period. Following [11], we preprocessed this data set to obtain a sequence of feature vectors that capture the *state* of an individual at the time of each campaign, resulting in many features not explicitly present in the original data.

The second 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. 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.

6.2 Resource Constraints

The KDD Cup and Saks data sets do not explicitly provide information on constraints on actions, so we provided them for our experimentation. In order to parallel the multi-organization setting of our problem, 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. Resource constraints are then defined as bounds on the number of actions that can be performed by each department per unit time.

Table 7 provides information on the group definitions and resource bounds we used for both data sets, 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. We note that the resource bounds were selected to be approximately proportional to the department sizes.

6.3 Evaluation Results

The main comparison method is that of using a standard reinforcement learning algorithm to estimate the values of competing actions, and then using optimization only at the time of action allocation.

Figure 4 exhibits the results of this comparison for both data sets. What is plotted in the graph is actually the percentage improvement over the sampling policy as a function of the number of learning iterations. Since the evaluation is stochastic in nature, we averaged the results over 10 randomized runs. In both cases, we observe a significant improvement in performance for consecutive iterations of both algorithms. This means that both of these approaches outperform the canonical approach of combining data modeling and constrained optimization (without reinforcement learning), which is equivalent to the first iteration in both methods.[2] Furthermore, we see that constrained reinforcement learning out-performs unconstrained reinforcement learning in later iterations in both cases. For the KDD Cup 98 data, 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 ($p < 10^{-7}$). For the Saks data, the differences in performance by the two methods, at iterations 3, 4, and 5, are very statistically significant based on a paired t-test ($p < 10^{-8}$).

We remark that the reward values in the experiments involving KDD cup 98 data are *not* directly comparable to those for the original cup task, due to different sets of features involved and the presence of resource constraints. We emphasize, however, that the first iteration of our method is similar to the approach used by the cup winner, and hence we have a competitive baseline.

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

(a) KDD Cup 98 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

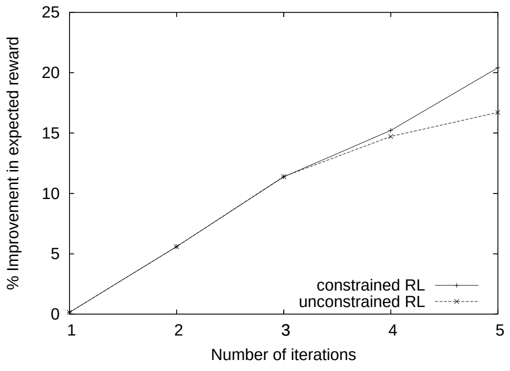
(b) Saks data

Table 7: Summary of resources for KDD Cup data and Saks data.

7. CONCLUDING REMARKS

We have presented a novel approach to the debt collections optimization problem, and described an actual deployment. The system went live in December of 2009 and there has been significant press coverage and attention (e.g. CNN Money, NY Channel 1, etc.) It will take more time to assess the actual monetary benefits, but the state expects savings of about 100 million dollars in the next three years. This figure is arrived at by a relatively conservative estimate of the *lift* at a few percentage points for the relevant portion of the collections that are affected by the system, and considering that the state's overall annual collections revenue is in the order of billion dollars. Non-monetary benefits should also be emphasized - Since the deployed system is based on data modeling, it will be able to adapt to environment changes without significant extra labor or costs. Furthermore, its

[2]In our experiments, the value function estimates were initialized by the cumulative rewards observed in the data. So the first iteration actually corresponds to modeling the observed long term rewards and running constrained optimization using the estimated models.



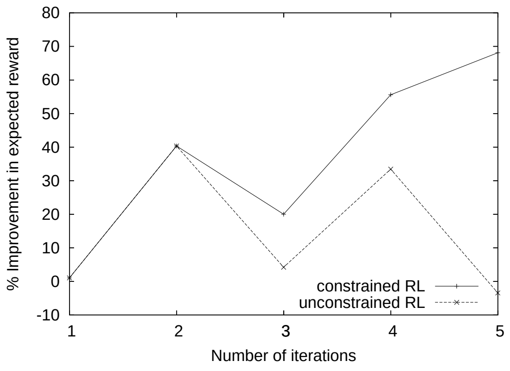


Figure 4: Expected rewards achieved by the two approaches on the KDD Cup 98 data (above), and Saks data (below): Percentage improvement over the sampling policy in expected rewards for each method is plotted for five learning iterations.

effectiveness is expected to improve over the years as more and more data are collected.

Acknowledgments

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

8. 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 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 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. 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.

[6] 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.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] F. Provost, P. Melville, and M. 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.

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

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

[15] 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.

[16] 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`.

[17] F. Wang and C. Zniolo. Xbit: An xml-based bitemporal data model. In *Proceedings of the 23rd International Conference on Conceptual Modeling*, pages 810–824, 2004.

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

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

[20] 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.