Universidade de São Paulo
Instituto de Matemática e Estatística
Bachalerado em Ciência da Computação

Gabriel Baptista

Aprendizado por Reforço em Lote para a construção de um Sistema de Cobranças

São Paulo
2017

Aprendizado por Reforço em Lote para a construção de um Sistema de Cobranças

Monografia da disciplina
MAC0499 – Trabalho de
Formatura Supervisionado.

Supervisora: Prof[a]. Leliane Nunes de Barros

São Paulo
2017

Agradecimentos

Para a execução deste trabalho gostaria de agradecer em primeiro lugar aos meus pais pelo esforço deles para com minha educação desde quando eu era criança, além do esforço deles para me proporcionar conforto.

Agradeço ao meu irmão por toda a ajuda durante a vida estudantil e em especial pela força dada durante a confecção deste trabalho.

Agradeço à empresa Adimplere pelo fornecimento de dados reais, em especial ao Felipe por estar sempre disponível a me explicar os procedimentos internos da empresa.

Agradeço aos professores que transmitiram conhecimento o suficiente para a execução deste. Em especial à professora Leliane pelo apoio dado ao longo do ano.

Agradeço também à minha família e meus amigos que sempre transmitem alegria para que eu consiga continuar nos momentos mais difíceis.

Resumo

BAPTISTA, G. **Aprendizado por Reforço em lote para a construção de um Sistema de Cobranças**. 2017. Trabalho de Conclusão de Curso - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017.

Neste trabalho, modelamos o problema de cobrança de inadimplentes como um processo de decisão markoviano e usamos a ténica de aprendizado por reforço em lote para aprendermos a melhor política de cobrança, para cada uma das carteiras de inadimplentes.

Para a execução deste trabalho nos foi fornecida uma parte de um banco de dados de uma empresa especializada em cobranças, chamada Adimplere.

Palavras-chave: Aprendizado por Reforço, Sistema de Cobranças, Aprendizado por reforço em lote

Sumário

1 Introdução **2**

2 Processo de Decisão Markoviano **4**
2.1 Política 5
2.2 Valor de Política 5
2.3 Algoritmo Iteração de Valor 6

3 Aprendizado por Reforço **7**
3.1 Q-Learning 7
3.1.1 Q-Valor 7
3.1.2 Q-value update 8
3.1.3 O algoritmo 8

4 Aprendizado por Reforço em Lote **10**
4.1 Algoritmo Fitted Q-Iteration 11

5 O problema do sistema de cobranças **13**
5.1 A empresa de cobrança 15
5.2 A tarefa de cobrança 15
5.3 O sistema automático de cobrança 16

6 Resolução do problema **17**
6.1 Classificação das ações 17
6.2 Classificação dos estados 18
6.3 Recompensa e Custo 20

7 Resultados Experimentais **22**
7.1 K-Fold 22
7.1.1 Leave-one-out 23
7.2 Experimentos para as carteiras da Adimplere 24
7.3 O valor dos estados 26
7.4 A política ótima para o sistema de cobrança 28

8 Discussão sobre a modelagem aplicada **29**
8.1 A dívida 29
8.2 Método anterior de cobrança 30
8.3 Considerações sobre a modelagem do MDP de cobrança 30

9 Próximos projetos **31**

10 Conclusão **32**

1 Introdução

Esse trabalho tem como proposta aplicar técnicas de inteligência artificial para o desenvolvimento de um sistema de tomada de decisão sequencial aplicado ao problema de cobrança de clientes inadimplentes, tendo como objetivo maximizar o número de acordos financeiros para a empresa de cobrança e seus parceiros.

Um primeiro conceito que deve ser considerado é o de planejamento probabilístico, que leva em conta tomadas de decisões sequenciais em que ações implicam em efeitos probabilísticos. No planejamento probabilístico, o problema é modelado como um Processo de Decisão Markoviano *(Markov Decision Process - MDP)* [14], no qual um agente seleciona uma ação para ser executada num ambiente, levando-o de um estado a outro com uma probabilidade associada; para cada ação executada o agente têm um custo para executá-la e também recebe uma recompensa como forma de um *feedback*. O objetivo é acumular o máximo de recompensas e o mínimo dos custos ao longo de suas ações.

Em muitos problemas que envolvem MDPs, não se têm conhecimento completo do ambiente no qual o agente irá atuar e nesse caso, o agente pode realizar uma série de experiências no ambiente e analisar as recompensas recebidas por cada uma, constituindo assim um modelo do ambiente. Essa abordagem permite aprender a melhor sequência de ações que leva o agente a maximizar as recompensas menos os custos acumulados. Este processo recebe o nome de Aprendizado por Reforço *(Reinforcement Learning - RL)* [11].

Um dos principais algoritmos de RL é o Q-Learning [12], considerado ineficiente para problemas reais, pois requer um número grande de interações até alcançar a convergência desejada, isto é, quando alcançada. Um outro caso que podemos enfrentar é quando o agente não pode interagir com o ambiente durante a fase de aprendizado, possuindo apenas uma amostra de interações do passado que são dadas `a priori`. Ao utilizar essa amostra para inferir sobre o ambiente, nem sempre é possível encontrar a sequência de ações ótima, mas a partir desta amostra pode-se encontrar uma sequência de ações que seja a melhor possível para esse conjunto de dados, caracterizando um problema de Aprendizado por Reforço em Lote (Batch Reinforcement Learning - BRL)[13], sendo essa a abordagem adotada nesse trabalho.

No dia-a-dia, é possível aplicar técnicas de planejamento probabilístico e aprendizado por reforço em diversas situações, sendo uma delas o problema de cobranças.

O problema de cobrança de inadimplentes. Uma empresa especializada em cobranças, possui várias companias (parceiros) que disponibilizam dados de seus clientes inadimplentes. Tal empresa deve ser capaz de selecionar a melhor

política de ações de cobrança (também chamadas de campanhas), isto é, propostas de acordos para o pagamento de dívidas, com o objetivo de influenciar os inadimplentes a quitá-las. Todas as campanhas são armazenadas num banco de dados, tal como: os diferentes tipos de cobranças realizadas para os diferentes parceiros (*e-mail*, correio, telefonema, etc) o formato da cobrança (geral, para um grupo de clientes específicos ou personalizada para cada cliente); e diferentes acordos de pagamento (permitindo ou não futuras negociações). Uma das decisões críticas que a empresa de cobrança deve tomar é excluir um cliente de sua lista de inadimplentes, desistindo assim de receber o pagamento daquela dívida. Essa decisão é tomada com base em uma análise do custo acumulado da cobrança mediante a chance de quitação da dívida. Para o problema de cobrança de inadimplentes é possível aplicar técnicas de BRL que use as experiências de campanhas armazenadas pelos diferentes paceiros de uma empresa especializada em cobranças, para aprender a melhor política de cobranças.

Para este estudo, nos foi fornecida uma base de dados que funcionará como nossa amostra dada `a priori`, pertencente a uma empresa chamada Adimplere, especializada em cobranças com empresas parceiras de inúmeros ramos. Ela é responsável por intermediar a comunicação destas empresas com seus inadimplentes e armazenar os dados de suas campanhas de cobrança. O sistema de Aprendizado por Reforço em Lote aplicado a esse conjunto de dados, tem por objetivo aumentar o retorno financeiro para cada parceira em questão, bem como para a própria empresa de cobranças, a Adimplere.

Essa monografia está organizada da seguinte forma. Na seção 2 definimos os Processos de Decisão Markoviano (MDPs), seus principais conceitos e o algoritmo Iteração de Valor que devolve a política ótima para um dado MDP. Na seção 3 definimos os conceitos de Aprendizado por Reforço, bem como o algoritmo *Q-Learning* que é responsável por armazenar o valor esperado acumulado da recompensa menos o custo para cada par estado, ação (função Q). Na seção 4 determinamos um conceito mais específico, o de Aprendizado por Reforço em Lote, também o algoritmo *Fitted Q Iteration*, que é responsável pelo cálculo da função Q para um lote de experiências passadas. Na seção 5, explicamos o problema do sistema de cobranças, definimos alguns conceitos importantes para o seu entendimento, assim como as relações envolvidas no processo. Na seção 6, descrevemos como o problema foi modelado para ser resolvido. Na seção 7, observamos alguns resultados obtidos experimentalmente, bem como alguns conceitos envolvidos para uma análise que é feita ao fim de suas subseções. Na seção 8, há uma discussão sobre a modelagem aplicada para a resolução do problema, embasada nas discussões com a Adimplere. Na seção 9, há uma breve discussão sobre possíveis projetos que podem ser feitos, bem como na seção 10, uma conclusão de nosso trabalho.

2 Processo de Decisão Markoviano

Um Processo de Decisão Markoviano (*Markov Decision Process* - MDP) [14], consiste em um modelo matemático para tomada de decisões sequenciais onde um agente deve escolher uma ação a cada estado do ambiente que o leva a um novo estado com uma dada probabilidade e com uma recompensa e custo associados a essa transição.

Formalmente um Processo de Decisão Markoviano é definido como uma tupla $\langle$S, A, P, R, C$\rangle$, onde:

- S é um conjunto finito de estados do ambiente;
- A é um conjunto finito de ações que o agente pode tomar;
- R é a função de recompensa, definida como $R(s, a)$, representando a recompensa obtida pelo agente ao tomar a ação a no estado s;
- C é a função de custo, definida como $C(s_t, a, s_{t+1})$, representando o custo de tomar a ação a no estado s_t levando ao estado s_{t+1};
- P é a função de transição de estados sobre S que pode ser representada como a probabilidade condicional $P(s_{t+1} \mid s_t, a_t)$ que representa a probabilidade do agente ir para o estado s_{t+1} dado que o agente está no estado s_t e executa a ação a_t.

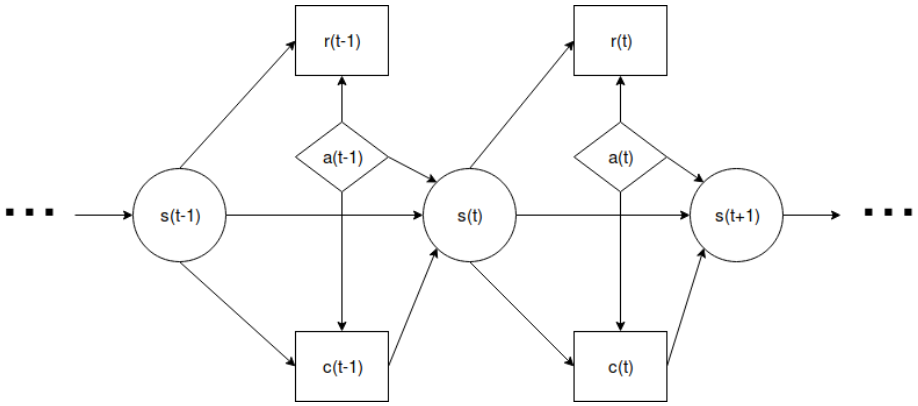


Figura 1: Dependências envolvidas em um Processo de Decisão Markoviano: $r_t(s_t, a_t)$, $c_t(s_t, a_t, s_{t+1})$ e $P(s_{t+1} \mid s_t, a_t)$.

A figura 1 mostra o modelo de execução de um MDP. A cada unidade de tempo t, o agente está no estado s_t e executa a ação a_t, que o leva para o estado s_{t+1}, que produz uma recompensa r_t e um custo c_t.

Este processo tem o nome de Processo Markoviano pois o estado presente depende apenas do anterior, não importando os outros estados visitados no passado e nem os possíveis estados futuros, dessa forma, se comportando como uma cadeia de Markov [4] e podemos escrever:

$$P(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, s_{t-2}, a_{t-2}, ..., s_0) = P(s_{t+1} \mid s_t, a_t), \tag{1}$$

dessa forma, o agente não guarda uma memória de seus passos, importando apenas a última interação com o ambiente.

2.1 Política

A solução de um MDP é uma política, isto é, uma função que mapeia estados em ações, $\pi : S \longrightarrow$ A. Estamos interessados na escolha do melhor mapeamento possível, obtendo assim o que é chamado de política ótima [4]. Uma política ótima chamada de $\pi^* : S \longrightarrow$ A, é responsável por maximizar o valor acumulado esperado da recompensa recebida menos custo, num horizonte infinito.

2.2 Valor de Política

Para analisar uma política π dada, definimos, uma função utilidade que corresponde à recompensa menos o custo acumulado esperado descontado num horizonte infinito de estágios de decisão; ou seja, dado um estado s,

$$V^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^t (r_t - c_t) \mid s = s_0], \tag{2}$$

onde E[x] é a esperança da expressão "x". A função V^{π}(s), $\forall$s $\in$ S, pode ser calculada através de um algoritmo de refinamento iterativo (com valores iniciais arbitrários), baseado em programação dinâmica, sendo que o valor resultante da função V^{π}(s) obedece a equação:

$$V^{\pi}(s) = \sum_{s' \in S} P(s' \mid s, \pi(s))(R(s, \pi(s)) - C(s, \pi(s), s') + \gamma V^{\pi}(s')), \tag{3}$$

onde V^{π}(s) é o valor do estado s, segundo a política π; R(s, a) é a recompensa associada à ação a tomada no estado s; C(s, a, s') é o custo associado à ação a tomada no estado s; γ é uma constante, chamada de fator de desconto, e está definida no intervalo [0,1), e P(s'| s, π(s)) é a probabilidade associada a transição dos estados seguindo π. Podemos definir a política ótima π^* como sendo a política com o valor maximal, considerando todas políticas possíves π de um MDP, isto é, $V^{\pi *}(s) \geq V^{\pi}(s), \forall \pi$. Sendo $V^{\pi *}(s) = V^*(s)$ a função valor ótima do MDP. Assim, a Equação (3) pode ser reescrita da seguinte maneira:

$$V^*(s) = \max_a \sum_{s' \in S} P(s' \mid s, a)(R(s, a) - C(s, a, s') + \gamma V^*(s')), \tag{4}$$

que é conhecida por Equação de Bellman [14]. Note que nessa formalização, usamos a função de recompensa dependendo apenas do estado atual e a ação (s, a), enquanto o custo depende da ação e dos estados atual e próximo (s, a, s').

2.3 Algoritmo Iteração de Valor

Um dos algoritmos clássicos para calcular V^*(s) é chamado de Iteração de Valor (Value Iterarion - VI). Esse algoritmo também é do tipo refinamento iterativo usando programação dinâmica e modifica a Equação (4), da seguinte forma:

$$V^t(s) \leftarrow \max_a \sum P(s' \mid s,a)(R(s,a) - C(s,a,s') + \gamma V^{t-1}(s')), \qquad (5)$$

isto é, fazendo sucessivas atualizações à função valor para todos os estados de S.

O algoritmo VI considera valores arbitrários para V^0(s), $\forall$ s $\in$ S, e a cada iteração t, computa-se V^t(s) através da Equação (5). Em [4], foi provado que VI converge depois de infinitas atualizações, isto é, V^∞(s). Na prática, VI é executado até que o erro residual entre duas iterações, Res(s), seja menor que um dado epsilon, $\forall$ s $\in$ S.

3 Aprendizado por Reforço

O conceito de Aprendizado por Reforço (Reinforcement Learning - RL) pode ser aplicado em problemas que envolvam escolhas sequenciais e que possam ser modelados como um MDP. Para alcançar a política ótima, o agente pode se encontrar em um ambiente parcialmente desconhecido, ou até mesmo totalmente desconhecido, e através de tentativa e erro de suas iterações com o ambiente, ele consegue aprender a política ótima. [3]

Pode-se considerar que no aprendizado por reforço a ideia é introduzir em um ambiente, que desconhece sua dinâmica. Um exemplo é a aquisição de um animal doméstico, um ser irracional que depois de experimentar diversas ações recebendo incentivos ou repreensões de seus donos (funcionando como reforço), aprende quais ações são certas e quais são erradas.

Dado um problema de RL, começamos por modelar o ambiente como um MDP, porém, não são conhecidas a priori, a função de transição P, a função de recompensa R e a função de custo C. A ideia de estimar essas funções em cada experiência, requer um custo computacional muito alto, então a melhor forma de resolver esse problema é estimar a função valor ótimo (V*) diretamente. Para isso, existem inúmeros algoritmos e o mais utilizado e que será apresentado aqui é o *algoritmo Q-Learning*.

3.1 Q-Learning

Q-Learning é um método proposto por [12] que resolve problemas de RL. Ele estima a função valor do par (estado, ação) que determina a esperança das recompensas menos os custos futuros, no momento em que o agente executa uma ação em um determinado estado e segue a política ótima nos passos futuros.

3.1.1 Q-Valor

Q-Valor é uma função que dada a função V(s), calcula a função valor do estado s ao se aplicar a ação a,

$$Q(s,a) = \sum_{s'} P(s' \mid s,a)[R(s,a) - C(s,a,s') + \gamma V(s')], \tag{6}$$

onde Q(s,a) representa o Q-valor associado ao par (estado s, ação a) e cada um dos outros conceitos envolvidos nessa equação são como definidos na seção anterior.

No caso em que conhecemos a função valor de estado ótimo, $V^*(s)$, a função Q-valor ótima é dada por:

$$Q^*(s,a) = \sum_{s' \in S} P(s' \mid s,a)(R(s,a) - C(s,a,s') + \gamma V^*(s')) \tag{7}$$

Como num MDP queremos achar a política ótima, podemos substituir a Equação (7) em (4), e teremos a Equação de Bellman escrita como:

$$V^*(s) = \max_a Q^*(s,a), \tag{8}$$

sendo π^*(s) = argmax$_a$ Q^*(s, a).

3.1.2 Q-value update

Com essa nova maneira (Equação (8)) de expressar a Equação de Bellman (Equação (3)), podemos ainda definir uma nova operação, a *Q-value update*, que para cada par (s,a) atualiza o Q-valor [10], da seguinte maneira:

$$Q^t(s,a) \leftarrow \sum_{s' \in S} P(s,a,s')[R(s,a) - C(s,a,s') + \gamma \max_a Q^{t-1}(s,a)], \tag{9}$$

onde t é um número inteiro que representa uma iteração do *Q-value update*.

Estamos interessados em fazer *Q-value updates* para cada par (s,a) com o objetivo de encontrar a política ótima, porém, como não são conhecidas as funções P, R e C, computa-se uma aproximação dos valores de Q durante a fase de aprendizagem. Para uma experiência de transição (s, a, s', r, c), onde s, s' ∈ S (conjunto de estados), a ∈ A (conjunto de ações), r ∈ R (conjunto de recompensas) e c ∈ C (conjunto de custos), temos:

$$Q(s,a) \approx r - c + \gamma \max_a Q(s',a'). \tag{10}$$

Mas, como estamos interessados na maior recompensa a longo prazo, precisamos ter uma medida sobre todas experiências anteriores, portanto, computa-se o valor Q da seguinte forma:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r - c + \gamma \max_a Q(s',a')], \tag{11}$$

onde $\alpha \in [0,1]$ é a taxa de aprendizagem. Sendo assim, Q funciona como uma média móvel das experiências recebidas a cada passo.

3.1.3 O algoritmo

Com todos os conceitos discutidos anteriormente, nesta seção apresentamos a definição formal do algoritmo de Q-Learning [12]. Ele consiste em aprender a função Q*(s,a) ao invés de $V^*(s)$. Para calcular a política ótima, fazemos:

$$\pi^*(s) = \arg\max_a Q^*(s,a), \tag{12}$$

isto é, para se calcular a política ótima para s ∈ S, é necessário encontrar um argumento que maximiza a função Q-value para todo par (s, a).

De acordo com todos os conceitos apresentados nessa seção, estamos prontos para uma primeira abordagem envolvendo um pseudo-código do algoritmo.

Algorithm 1 Q-Learning(S,A), adaptado de [10]

para i ← 0 **até** |S| **faça**
    para j ← 0 **até** |A| **faça**
        $Q(i, j) = random$ ▷ Inicializa Q(s,a) arbitrariamente
        $j \leftarrow j + 1$
    fim para
    $i \leftarrow i + 1$
fim para
Observa o estado atual s
enquanto Não convergiu **faça**
    Seleciona ação a
    Executa ação a em s
    Recebe r-c ▷ Reforço menos o custo de tomar a em s
    $Q(s, a) = (1 - \alpha)Q(s, a) + \alpha[r - c + \gamma \max a' Q(s', a')]$
    $s \leftarrow s'$
fim enquanto
Retorna $\arg\max_a Q(s, a)$

O algoritmo 1 recebe um conjunto de estados S e um conjunto de ações A e atualiza a função Q(s, a) utilizando a Equação 11, esse algoritmo é executado até a função Q convergir e então devolve a política ótima.

4 Aprendizado por Reforço em Lote

O algoritmo Q-Learning requer muitas iterações para convergir para uma política ótima, tornando sua aplicação inviável no mundo real, pois os recursos computacionais são finitos e muitas vezes não se tem tempo para esperar essa convergência. Em outros casos, o agente não tem interação com o ambiente durante o processo de aprendizado, então a solução é fazer o aprendizado a partir de um número fixo de amostras de transição de interações prévias.

Como o conjunto de experiências prévias geralmente é finito, o objetivo do BRL passa a ser o de encontrar a melhor política possível a partir dos dados que o agente possui.

Alguns algoritmos foram criados para BRL, sendo que a principal ideia é aplicar uma atualização de programação dinâmica para todos elementos de um conjunto de amostras, substituindo assim as atualizações locais. Dessa forma, o algoritmo Q-Learning passa a ter um comportamento estável, com a convergência garantida para uma função valor ótima aproximada [13].

Algorithm 2 BatchReinforcementLearning(S,A,m), adaptado de [6]

```
1:  Q ← Q_0
2:  enquanto Q não convergiu faça
3:      D ← {}
4:      i ← 0
5:      experiencias ← 0
6:      enquanto experiencias ≤ m -1 faça
7:          i ← i + 1
8:          experiencias ← experiencias + 1
9:          s_i ← estadoDoAmbiente()
10:         a_i ← selecionaAção(Q, s_i)
11:         executaAção(a_i)
12:         r_i ← recompensaDoAmbiente(s_i, a_i)
13:         c_i ← custoDoAmbiente(s_i, a_i)
14:         s_{i+1} ← estadoDoAmbiente()
15:         d_i ← (s_i, a_i, s_{i+1}, r_i, c_i)
16:         D ← D ∪ d_i
17:     fim enquanto
18:     Q ← FittedQIteration(D)
19: fim enquanto
```

O algoritmo 2 mostra o pseudo-código do algoritmo BRL. O algoritmo começa com um conjunto vazio de lotes e nas Linhas 6 - 17, um lote de m experiências $d_i = (s_i, a_i, s_{i+1}, r_i, c_i)$ é coletado. Para cada experiência, o agente se encontra num estado s_i, executa a ação a_i, selecionada de maneira gulosa ($\max_a Q(s, a)$), levando-o a um estado s_{i+1} e devolvendo um custo c_i e uma

recompensa r_i. Após a coleta de m experiências no lote D o algoritmo chama a função de treinamento, `Fitted Q-Iteration` (Algoritmo 3). No caso em que após o treinamento a função Q não tenha convergido (Linha 2), um novo lote de m experiências é coletado.

4.1 Algoritmo Fitted Q-Iteration

Seja um conjunto fixo D = {(s_t, a_t, s_{t+1}, r_t, c_t | t = 1, ..., m)} de m experiências (s, a, s', r, c) e um valor inicial para Q por exemplo, $q^0 = 0$, para todo (s, a). A cada iteração, podemos fazer uma melhor aproximação da função Q através dos seguintes passos [6]:

1. Dada a aproximação Q^{i-1}, para cada experiência $(s_j, a_j, s_{j+1}, r_j, c_j) \in$ D, calcula-se:

$$T_j^{i+1} = r_j - c_j + \gamma \max_a \hat{Q}^i(s', a), \tag{13}$$

   onde T_j é o valor aproximado alvo (*target*) para a função Q (como no algoritmo Q-Learning);

2. Usa-se aprendizado supervisionado para extrair uma nova aproximação de Q^i, isto é, ajustar a aproximação de T_j.

O Algoritmo 3, chamado de `Fitted Q Iteration` [13] implementa as ideias descritas acima:

Algorithm 3 FittedQIteration(D,m), adaptado de [6]

```
1: i ← 0
2: Q_i ← Q_0
3: enquanto i ≤ m faça
4:     para j ← 1 até tamanhoLote(D) faça
5:         T_j ← r_j − c_j + γ max_a Q_{i−1}(s_{j+1}, a) ▷ experiência (s_j, a_j, r_j, c_j, s_{j+1})
6:         j ← j + 1
7:     fim para
8:     \\ Ajusta a aproximação alvo (T_j) usando aprendizado supervisionado
9:     para iteracao ← 1 até k faça
10:        para j ← 1 até tamanhoLote(D) faça
11:            Q_i(s_j, a_j) ← Q_i(s_j, a_j) + α(T_j - Q_i(s_j, a_j))
12:            j ← j + 1
13:        fim para
14:        iteracao ← iteracao + 1
15:    fim para
16:    i ← i + 1
17: fim enquanto
18: retorna Q
```

O algoritmo 3 recebe um lote D de experiências, com |D| múltiplo de m, que é responsável por limitar o número de iterações no aprendizado. O algoritmo também leva em conta os dois passos descritos anteriormente. No laço de linhas (3-6) o algorimo atualiza a função Q (como na Equação de Bellman) a partir do lote D, com o cálculo de uma função T_j para cada experiência. No laço das linhas (11-16) gera uma nova aproximação para Q (como no *Q-Learning*), fazendo um aprendizado supervisionado, em no máximo k iterações. Dessa forma, k deve ser bem escolhido para evitar um *overfitting* ou *underfitting* da função. Devolve a função Q atualizada de acordo com o lote recebido inicialmente.

5 O problema do sistema de cobranças

Vamos imaginar que uma pessoa deseja realizar a compra de um televisor numa loja de eletrodomésticos e decide efetuar o pagamento via boleto ou cartão de crédito em 12 vezes, com um contrato pré-definido. Caso ocorra a quebra do contrato, ela entra para uma lista de inadimplentes da loja.

No caso de grandes redes de lojas, surge a necessidade do gerenciamento de sua lista de inadimplentes (essa lista pode conter milhares ou até milhões de clientes, como uma das empresas consideradas nesse trabalho que possui 400000 inadimplentes) e, muitas vezes, a rede não consegue conciliar a tarefa de cobrança de inadimplentes e o processo natural de vendas. Por isso, existem empresas de cobrança que formam parceria com as grandes redes de lojas e são responsáveis somente pela tarefa de cobrança.

Dentre os principais conceitos envolvidos em um sitema de cobrança estão:

- *Empresa de cobrança*: Empresa responsável pela cobrança das dívidas de clientes de empresas parceiras que a contratou.
- *Parceiro*: Para a empresa de cobrança, o parceiro é a empresa que vende produtos e gera contratos com os clientes, que podem ser quebrados gerando uma lista de inadimplentes, que deve ser fornecida à empresa de cobrança.
- *Cliente*: Para a empresa de cobrança, o cliente é a pessoa que se encontra na lista de inadimplentes.
- *Carteira*: Uma lista de inadimplentes de um parceiro, um parceiro pode possuir mais de uma carteira.
- *Aumento da Carteira*: Durante o processo de cobrança, a lista de inadimplentes pode ser atualizada com a adição de novos clientes fornecidos pela empresa parceira, constituindo um aumento da carteira.
- *Exclusão de clientes da Carteira*: Durante o processo de cobrança, a lista de inadimplentes também pode ser atualizada com a exclusão de clientes que não respondem às propostas de acordos e que provavelmente não quitarão suas dívidas. Essas exlusões também são realizadas para minimizar os custos envolvidos nos envios das campanhas.
- *Acordo*: Denota o fim do processo de negociação cliente - empresa; é o momento em que o cliente paga sua dívida, ou gera boletos, para pagá-la.
- *Acordo Válido*: Este conceito aplica-se no caso em que o pagamento da dívida se dá durante um período (pagamento à prazo), dizemos que um acordo é válido se o pagamento das parcelas de um cliente não passou do prazo permitido.

- *Acordo Inválido*: Se o boleto vence e o pagamento não for identificado até 2 dias úteis após o vencimento, o acordo passa de válido para inválido e é considerado como quebrado e o cliente volta à carteira.
- *Campanha*: Ação de cobrança aos clientes de uma determinada carteira através da empresa de cobrança. Tem por objetivo que o maior número possível de pessoas negocie a sua dívida e gere acordos para quitação de suas dívidas.
- *Histórico da Carteira (h)*: Armazena todo o processo das campanhas realizadas para uma carteira.

O processo representado na Figura 2 pode ser usado para sintetizar o problema da Cobrança. O parceiro, em algum momento do processo de venda de seus produtos, mantém uma lista de inadimplentes que é passada para a empresa, constituindo uma carteira. A empresa de cobrança gera campanhas (ações de cobrança) com o objetivo de recuperar o valor das dívidas e com isso, o cliente é notificado e pode vir a fechar um acordo para pagamento de sua dívida.

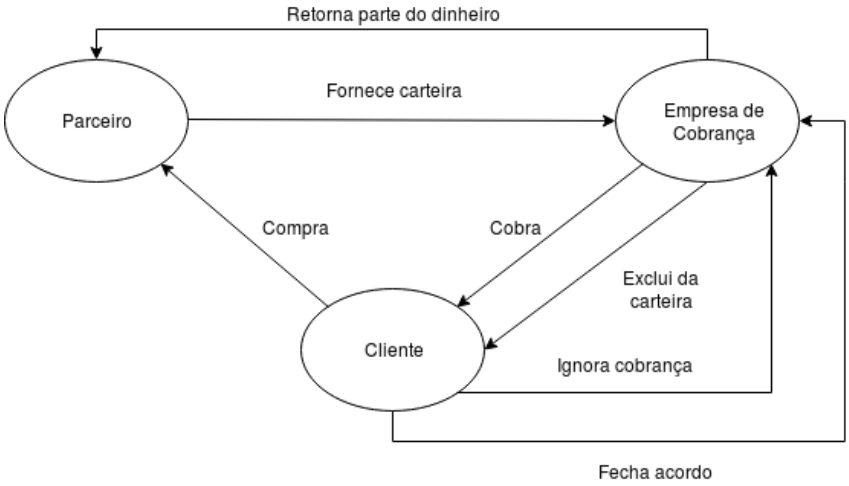


Figura 2: Ilustração do processo de cobrança envolvendo um Parceiro, Clientes e a Empresa de Cobrança. Ovais representam as entidades do problema e as linhas rotuladas com setas indicam as relações entre entidades.

O parceiro vende seus produtos para diversos clientes e alguns deles acabam não pagando pelos mesmos, formando assim uma lista de inadimplência. Essa lista é disponibilizada à empresa de cobrança, constituindo uma carteira, que fica responsável por cobrar os clientes através de campanhas de propostas de acordos. O cliente pode vir a ignorar sucessivas campanhas de cobrança, o que pode causar sua exclusão da lista de inadimplentes. Isso significa uma confirmação da perda do valor devido. Outros clientes podem fechar acordos para quitar

suas dívidas, e nesse caso, a empresa de cobrança deve repassar parte da dívida recuperada para a empresa parceira e ficar com a parte que lhe é cabida. Assim, tanto a empresa de cobrança como seus parceiros têm interesse em definir uma política ótima de campanhas.

5.1 A empresa de cobrança

Para o parceiro, quase que majoritariamente o lucro obtido vem dos pagamentos convencionais de venda de seus produtos. As empresas especializadas em cobrança são responsáveis única e exclusivamente por cobrar clientes inadimplentes. Como o parceiro muitas vezes não tem o foco nesses clientes, a empresa de cobrança acaba ficando com uma porcentagem alta dos valores recuperados. Portanto, surge a necessidade de um aperfeiçoamento na relação deste tipo de empresa com os clientes inadimplentes, visando assim um aumento no retorno financeiro.

O primeiro contato (que chamamos de campanha) da empresa de cobrança com o cliente serve como um aviso ao mesmo. Uma vez que o cliente está ciente, ele pode responder e gerar um acordo, as campanhas subsequentes com os clientes já levam em conta alternativas com o objetivo de convencer o maior número de clientes a fechar acordos rapidamente, oferecendo diferentes formas de pagamento permitindo negociações de acordo.

Campanhas são feitas de diversas maneiras, como por exemplo: *Short Message Service* (SMS), *whatsapp*, correio (utilizado em locais mais afastados, como em regiões rurais e onde os meios mais tecnológicos não chegaram), telefone (utilizado quando o fechamento do acordo se torna primordial para a continuidade da carteira, pois é um meio de custo alto) e o mais importante e mais utilizado são os *e-mails*. Eles serão o foco neste estudo por conta da facilidade de armazenamento de informações e pela quantidade de campanhas já enviadas por *e-mail* pela empresa de cobrança fornecedora dos dados, a Adimplere.

Cada *e-mail* contém as informações necessárias para que o cliente negocie sua dívida. Nele está contido o valor inicial da dívida, sua data de efetuar os pagamentos e o período de inadimplência, bem como o desconto oferecido. Além disso, são disponibilizados *links* que redirecionam o cliente para *sites* onde existe mais informações sobre a dívida, negociações e acordos. Caso o cliente não utilize nenhuma das opções anteriores, ele pode fazer a negociação respondendo o próprio *e-mail*.

5.2 A tarefa de cobrança

A empresa de cobrança envia *e-mails* para todos os clientes de determinada carteira sob a forma de campanhas. O *e-mail* tem um esqueleto pré-definido, e

é personalizado com os dados de cada cliente (nome, valor da dívida, desconto), porém a forma de cobrança é a mesma para todos clientes.

A estrutura de um *e-mail* pode ser descrita com uma sequência de caracteres pré-definidas, onde o título do *e-mail* vem seguido de uma saudação ao cliente com seus dados (nome, CPF/ CNPJ), detalhes da dívida com o seu valor e o desconto oferecido, indicação de *sites* para iniciar a negociação da dívida, informações adicionais sobre a proposta: nome e contato do parceiro, quantidade de parcelas e o prazo que é válida e por fim o contato da empresa de cobrança

Neste trabalho, estamos interessados em analisar quais campanhas têm mais sucesso que as outras, aprendendo assim qual seria a melhor forma de abordar o cliente para uma maior recuperação dos valores devidos.

5.3 O sistema automático de cobrança

Para aplicarmos BRL, estamos interessados nos dados referentes às campanhas e iremos utilizar: *Identity* (ID) da campanha (está associado a uma data), nome da carteira, título da campanha, número de inadimplentes, número de acordos gerados no dia[1].

Um ponto crucial é manter o *Internet Protocol*(IP) da empresa limpo, pois o envio repetitivo de cobranças para *e-mails* que o julguem como *spam* acaba sujando o IP da empresa, fazendo com que os próprios servidores de *e-mail* julguem todos *e-mails* com esse IP como *spam* e os enviem para a caixa de *spam* do cliente, prejudicando assim a cobrança. Portanto, um serviço que é constantemente utilizado pela empresa de cobrança é o de limpeza dos clientes da carteira, excluindo alguns clientes da lista de inadimplentes. Como um efeito colateral desse processo de exclusão de clientes, a empresa de cobrança considera perdido os valores devidos por esses clientes (o que pode ser visto como um custo da aplicação da campanha).

Outro fato que pode se observar no sistema é que os acordos gerados em um dia não necessariamente são referentes aos estímulos enviados sob forma de campanha do mesmo. De fato, é possível observar que após uma dada campanha, um número maior de acordos firmados do que os de *e-mails* clicados, indicando que campanhas anteriores podem gerar acordos em dias futuros.

Nesse trabalho, uma vez que estamos modelando esse problema com um processo de decisão markoviano, vamos assumir que os acordos observados em um instante t depende somente da campanha atual.

[1]Vale ressaltar que os dados utilizados nesse trabalho foram fornecidos pela Adimplere e que algumas informações são omitidas por serem confidenciais.

6 Resolução do problema

Para resolver o problema, foi necessário um pré-processamento dos dados para aplicar o algoritmo de BRL, isto é, como modelar um MDP a partir dos dados fornecidos pela empresa de cobrança, Adimplere. Essa empresa nos forneceu dados de 7 carteiras diferentes, cada uma com um número variado de clientes e campanhas.

6.1 Classificação das ações

Como descrito nas seções anteriores, iremos nos concentrar em aprender como os tipos de campanhas influenciam na quantidade de acordos gerados. Nesse trabalho, vamos nos restringir às campanhas feitas através do envio de *e-mails*.
Observamos que os títulos dos *e-mails* podem ser usados para caracterizar os tipos de campanha, uma vez que os títulos enviados nas campanhas seguem um padrão definido pela Adimplere.

{PRIMEIRO_NOME}: {PARCEIRO} - Feirão de Regularização
{PRIMEIRO_NOME}, quite seu débito {PARCEIRO} com {DESCONTO}% de desconto!
{PRIMEIRO_NOME}, negocie com até {DESCONTO}% de desconto por tempo limitado!
{PRIMEIRO_NOME}, negocie com até {DESCONTO}% de desconto por tempo limitado!
{PRIMEIRO_NOME}, evite transtornos. Não perca essa oportunidade.
{NOME}, evite transtornos. Não perca essa oportunidade.
{PRIMEIRO_NOME}, negocie com até {DESCONTO}% de desconto por tempo limitado!
{PRIMEIRO_NOME}, ainda dá tempo de aproveitar os {DESCONTO}% de desconto!
O tempo está se esgotando, {PRIMEIRO_NOME}. Você não irá fazer nada?
Campanha de regularização - {PARCEIRO}
{PRIMEIRO_NOME}, quite seu débito {PARCEIRO} com {DESCONTO}% de desconto.
Campanha de regularização - {PARCEIRO}
Campanha de regularização - {PARCEIRO}
{PRIMEIRO_NOME}, liquide seu débito com {DESCONTO}% de desconto e evite maiores transtornos!
{PRIMEIRO_NOME}, evite transtornos. Não perca essa oportunidade
Campanha de regularização - {PARCEIRO}
Campanha de regularização - {PARCEIRO}
{PRIMEIRO_NOME}, liquide seu débito {PARCEIRO} com {DESCONTO}% de desconto!
{PRIMEIRO_NOME}, o tempo para quitar o seu débito {PARCEIRO} com {DESCONTO}% de desconto está se esgotando
Campanha de regularização - {PARCEIRO}

Figura 3: Parte da amostra de títulos de *e-mails* enviados para um dos parceiros da Adimplere.

Como podemos ver na Figura 3, existem algumas palavras-chaves no título, que servirão para a nossa classificação. Palavras como {PRIMEIRO_NOME}, {NOME} e {DESCONTO} se referem a uma característica própria de cada cliente, {PARCEIRO} se refere ao nome do parceiro. Note que alguns *e-mails* são enviados com uma conotação de tempo: "Último dia para quitar seu débito

com {DESCONTO}% de desconto", é um exemplo enviado em algumas campanhas. Enquanto que outros são enviados com mais personalização ao cliente: juntamente com o nome do mesmo, estão associadas informações de seus documentos, o RG ou seu CPF/ CNPJ (condensadas em uma espécie de ID para mantermos a segurança dos dados confidenciais).

Ação	Título
a_1	GENÉRICO
a_2	NOME
a_3	NOME + DESCONTO
a_4	DESCONTO
a_5	NOME + ID
a_6	NOME + ID + DESCONTO
a_7	NOME + ID + TEMPO
a_8	NOME + TEMPO
a_9	DESCONTO + TEMPO

Tabela 1: A ação é o título do *e-mail*.

A Tabela 1 mostra o conjunto de 9 ações que compõe nosso modelo. Elas são responsáveis por agrupar o conjunto de títulos dos *e-mails* de todas as campanhas da Adimplere. Por exemplo, a ação a_1 corresponde aos *e-mails* enviados sem nenhuma informação a respeito do cliente, a ação a_2 corresponde aos *e-mails* enviados contendo **apenas** o nome (primeiro nome está contido aqui) do cliente, a ação a_9 possui informações do desconto *e também* uma conotação de tempo, já a ação a_5 contém além da informação do nome do cliente o ID associado ao mesmo. As outras ações são derivadas das que já foram descritas.

6.2 Classificação dos estados

Como o objetivo da empresa de cobrança é gerar mais acordos, isto é, diminuir a quantidade de inadimplentes a cada campanha enviada, é natural que a quantidade de clientes seja uma variável que descreva os estados do MDP que representam o processo de cobrança.

	CLIENTES	ACORDOS	EXCLUSÕES
1	237398	49	10829
2	226520	88	193
3	226239	90	3189
4	215833	38	7076
5	225598	94	0
6	219853	81	5651
7	218511	73	2247
8	216191	78	3664
9	212449	87	0
10	310723	203	55493
11	255027	180	276
12	254298	115	158
13	250788	113	3397
14	259926	82	0

Tabela 2: Parte dos dados de uma carteira, a primeira coluna representa a quantidade de clientes inadimplentes de determinada campanha, a segunda a quantidade de acordos que ocorreram e a terceira a quantidade de exclusões de clientes.

A Tabela 2 mostra os dados de uma carteira para 14 campanhas, extraídas de um histórico de 56 (extraídas do que chamamos de Carteira C). A primeira coluna indica a quantidade de clientes inadimplentes no instante de envio da campanha, a segunda indica o número de acordos e a terceira o número de clientes excluídos após determinada campanha.

Na Tabela 2 observamos alguns comportamentos interessantes, como um aumento repentino na quantidade de clientes (entre as linhas 9 e 10), indicando um aumento da carteira e logo depois uma diminuição nesse número (entre 10 e 11), este fato se deve à necessidade de manter o IP da empresa de cobrança limpo, excluindo alguns clientes com *e-mails* que não respondiam às campanhas.

Para não termos uma quantidade muito grande de estados, o que poderia dificultar a convergência do algoritmo BRL [15], consideramos faixas de quantidades de clientes, que correspondem a uma partição do espaço de estados de cada carteira.

Estado	Faixa de valores
s1	0 .. 220000
s2	220001 .. 230000
s3	230001 .. 240000
s4	240001 .. 250000
s5	250001 .. 310723

Tabela 3: Partição do espaço de estados da Carteira C.

Um exemplo de divisão dos dados por faixas de quantidades da variável CLIENTES está na Tabela 3, cada faixa representa um estado da carteira em questão. Essa mesma ideia foi aplicada em todas as carteiras da Adimplere.

6.3 Recompensa e Custo

Como a empresa de cobrança deseja a recuperação de dinheiro, que vem através dos acordos, associarmos a quantidade de acordos com a recompensa recebida após a execução de uma campanha em um dado estado. Dada uma experiência (s, a, s', r, c) a função de recompensa é definida como:

R(s, a) = # de acordos efetuados após a campanha a em s.

Existe um custo em manter o IP da empresa ativo e em boas condições para o envio de campanhas futuras, sendo essa a principal razão da exclusão de clientes que não demonstram interesse pelas campanhas que vem sendo efetuadas pela empresa de cobrança. A exclusão de clientes além de implicar na manutenção de IP, significa a confirmação da perda de possíveis acordos. Por exemplo, na Tabela 2, é possível observar que após a campanha J, houve uma exclusão de 55493 clientes. Assim, consideraremos a quantidade de clientes excluídos como o custo de uma campanha a ser realizada no estado s e levar para o estado s', calculada da seguinte forma:

$$C(s, a, s') = \#clientesexcluidos = CLIENTES(s) - R(s, a) - CLIENTES(s') \tag{14}$$

	s	a	s'	r	c
1	s3	a2	s2	49	10829
2	s2	a1	s2	88	193
3	s2	a2	s1	90	3189
4	s1	a3	s2	38	7076
5	s2	a3	s1	94	0
6	s1	a4	s1	81	5651
7	s1	a5	s1	73	2247
8	s1	a1	s1	78	3664
9	s1	a9	s5	87	0
10	s5	a4	s5	203	55493
11	s5	a7	s5	180	276
12	s5	a8	s5	115	158
13	s5	a1	s5	113	3397
14	s5	a2	s5	82	0

Tabela 4: Parte dos dados de uma carteira, a primeira coluna representa o estado da campanha após a classificação por faixas, a segunda a campanha enviada, a terceira o próximo estado, a quarta a recompensa (acordos) e a quinta o custo (exclusões de clientes).

A Tabela 4 representa as experiências utilizadas pelo sistema modelados como descrito nesse capítulo, em cada linha as experiências são da forma (s, a, s', r, c).

7 Resultados Experimentais

As experiências extraídas do banco de dados da empresa Adimplere são da forma: $(s_i, a_i, s_{i+1}, r_i, c_i)$, sendo s_i o estado (#clientes) em que a campanha a_i foi enviada resultando no estado s_{i+1} (#clientes no instante em que a próxima campanha foi enviada); a recompensa r_i o #acordos fechados após a_i e início da próxima campanha e o custo c_i é o número de clientes excluídos após a_i e início da próxima campanha.

No problema do sistema de cobranças, estamos interessados em encontrar a política que maximize a esperança das recompensas menos custos recebidos ao longo das campanhas (Equação (12)).

Para analisarmos o desempenho do algoritmo de aprendizado por reforço em lote aplicado ao banco de experiências da empresa Adimplere, iremos utilizar a técnica conhecida por *k-fold*, uma técnica de validação cruzada [6], que explicaremos a seguir.

7.1 K-Fold

O *k-fold* consiste em uma técnica de validação cruzada no aprendizado de máquina. No nosso caso, essa técnica avalia a capacidade de estimar a política ótima a partir de uma amostragem de experiências em um sistema dinâmico.

O método consiste em particionar a amostra em K partes independentes entre si, com tamanhos iguais (ou próximos). Como ilustrado na Figura 4, dado um conjunto de experiências, faz-se uma partição de tamanho K, sendo que uma parte servirá para executar **teste** e as outras K - 1 serão usadas no **aprendizado**, o BRL neste caso.

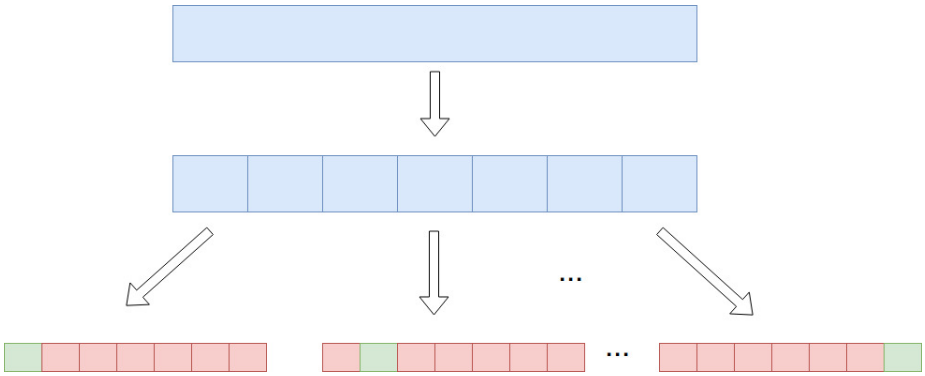


Figura 4: Esquema representando um k-fold, com k = 7, as partições vermelhas são as utilizadas para fazer o aprendizado e as verdes para os testes.

Para quantificar esse processo, consideremos:

- $\hat{Q}(s, a)$: O valor aprendido esperado com o lote de experiências, para a função Q no estado s ao executar a ação a, a partir das K - 1 partições responsáveis pelo aprendizado;
- $\hat{V}(aprendido(s))$: A função valor aprendida a partir de $\hat{Q}$ para o estado s.
- *V(h(s_i))*: O valor da recompensa menos o custo acumulado do histório de uma carteira, a partir do estado s_i.
- $T = \{t_1, t_2, ..., t_n\}$, T $\subseteq$ D: Conjunto de testes de n tuplas da forma $t_i = (s_i, a_i, s_{i+1}, r_i, c_i)$, onde $s_i, s_{i+1} \in$ S, $a_i \in$ A, $r_i \in$ R e $c_i \in$ C que indica que partindo do estado s_i, tomando uma ação a_i, o agente chega no estado s_{i+1} com uma recompensa r_i e custo c_i associados a essa transição.

Desta forma, para cada um dos estados s $\in$ S, é possível determinar qual o valor de $\hat{V}(s)$, a partir da equação 7, considerando a função $\hat{Q}$ aprendida, ou seja:

$$\hat{V}^*(s) = \max_a \hat{Q}(s, a) \tag{15}$$

As Equações 13 e 14 funcionam para cada uma das iterações dentro da técnica *k-fold*, ao final dessas k execuções, podemos calcular uma medida de dispersão entre estes valores da seguinte maneira:

$$DISPERSAO(s_i) = \frac{\hat{V}^*(s_i)}{k} - \frac{V(h(s_i))}{k} \tag{16}$$

E em termos percentuais, podemos escrever essa medida como:

$$ERRO(s_i) = 1 - \frac{V(h(s_i))}{\hat{V}^*(s_i)} \tag{17}$$

A partir da Equação 16, obtemos informações mais confiáveis para o nosso sistema de cobranças, podendo utilizar essas informações para as futuras campanhas.

7.1.1 Leave-one-out

O método *leave-one-out* é um caso particular do *k-fold*, onde k = N, com N sendo o número de dados. Nesse método, calcula-se N vezes o erro.

Este método funciona como um bom estimador no caso em que se possui poucos dados, pois tem um custo computacional alto, e iremos utilizá-lo para gerar alguns testes, pois a maior carteira possui apenas 65 campanhas.

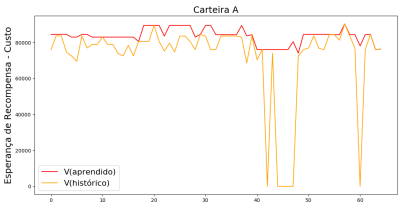
7.2 Experimentos para as carteiras da Adimplere

Como cada carteira tem comportamentos distintos, o aprendizado foi realizado para cada carteira separadamente. A Tabela 5 mostra as 7 carteiras e o número de campanhas realizadas para cada uma pela Adimplere, que corresponde ao número de experiências usadas no aprendizado.

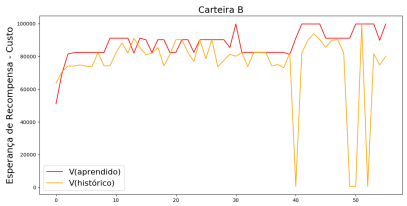
Carteira	Campanhas
A	65
B	56
C	42
D	43
E	30
F	30
G	63

Tabela 5: Descrição do número de campanhas em determinada carteira.

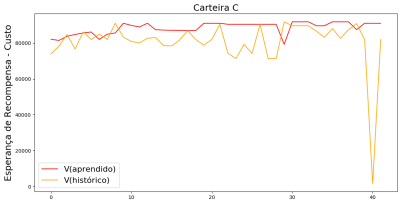
Assim, para cada carteira, utilizamos a técnica *leave-one-out*, e comparamos os valores de $V^*(S_0)$ aprendidos com os de $V^h(S_0)$ a partir dos históricos, sendo S_0 o estado inicial do histórico de cada carteira. Os resultados podem ser observados na Figura 5:



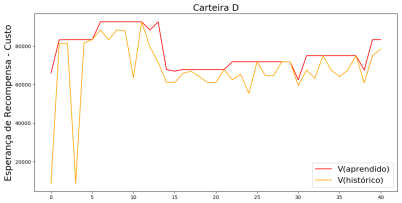
(a) Carteira A.



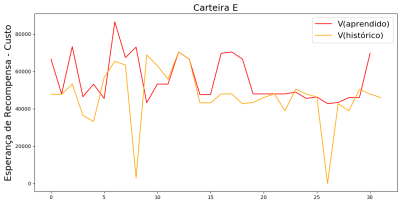
(b) Carteira B.



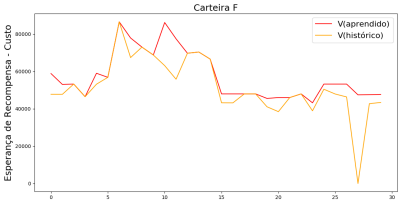
(c) Carteira C.



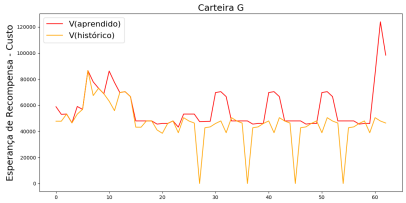
(d) Carteira D.



(e) Carteira E.



(f) Carteira F.



(g) Carteira G.

Figura 5: No eixo Y temos os valores aprendidos e valores dos históricos de cada carteira de acordo com o estado inicial S_0 e o eixo X representa o *fold* correspondente do *leave-one-out*.

A Figura 5 mostra os valores (eixo Y), $V^*(S_0)$ e $V^h(S_0)$, para cada *fold* do *leave-one-out* (eixo X), isto é, rodando o algoritmo BRL usando o conjunto de experiências excluindo apenas uma delas. Note que um pico invertido corresponde à exclusão de uma experiência que traria uma recompensa significativa para o valor de $V^h(S_0)$, enquanto $V^*(S_0)$ computado com a ausência daquela

mesma experiência, manteve o valor esperado maior. Uma explicação possível para isso é que, ao verificarmos os históricos, as experiências excluídas foram responsáveis pela exclusão ou inserção abrupta de clientes nas carteiras correspondentes. Também podemos observar que os valores $V^*(S_0)$ e $V^h(S_0)$ apresentam comportamentos similares sendo que na maioria dos casos, $V^*(S_0)$ é maior que o $V^h(S_0)$, o que era previsto, uma vez que $V^*(S_0)$ computa o valor esperado acumulado a partir de S_0.

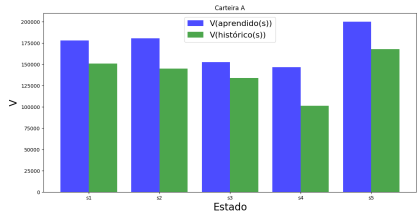
Carteira	ESTADO INICIAL (S_0)	ERRO	DISPERSÃO
A	s_2	0,35	25747,06
B	s_1	0,38	14755,90
C	s_1	0,37	26881,72
D	s_2	0,30	30401,37
E	s_5	0,38	9068,48
F	s_5	0,25	6942,86
G	s_5	0,44	45534,84

Tabela 6: Medidas, erro e dispersão, calculadas a partir de um estado inicial, S_0, para cada carteira.

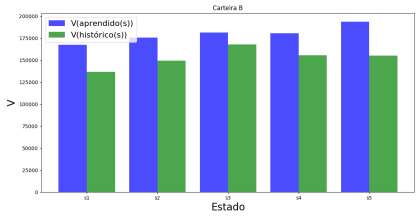
A Tabela 6 mostra as medidas de Erro e Dispersão calculas pelas Equações (15) e (16), para cada uma das carteiras. Observamos que a Carteira G apresentou as maiores medidas de Erro e Dispersão, enquanto a Carteira F apresentou as menores medidas. Isso pode ser explicado através da Figura 5, uma vez que essas foram as carteiras com a maior e a menor quantidade de picos, respectivamente.

7.3 O valor dos estados

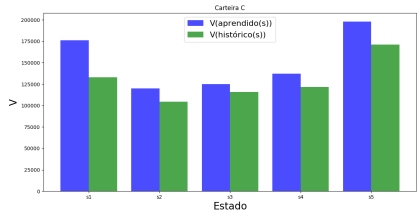
Uma vez que o modelo adotado possui um número pequeno de estados (5), na Figura 6, mostramos os valores $V^*(s_i)$ e $V^h(s_i)$ para cada carteira.



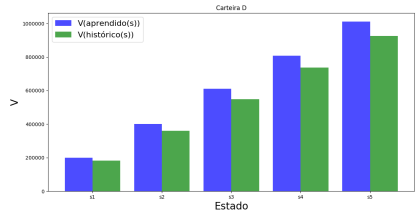
(a) Carteira A.



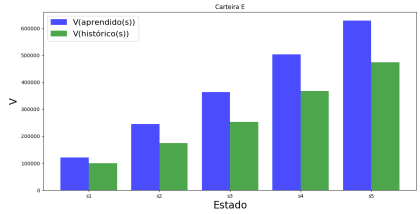
(b) Carteira B.



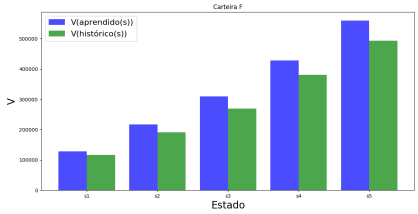
(c) Carteira C.



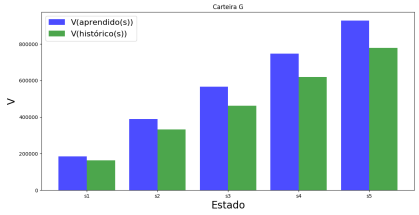
(d) Carteira D.



(e) Carteira E.



(f) Carteira F.



(g) Carteira G.

Figura 6: Comparação entre valores aprendidos ($V^*(s_i)$) e valores dos históricos ($V^h(s_i)$) para todos os estados de cada carteira.

Nessa análise, foram consideradas todas as campanhas para cada carteira e comparadas aos valores dos históricos também considerando todas as campanhas. A Figura 6 mostra que os estados com maiores quantidades de clientes inadimplentes, são os que possuem os maiores valores V^* e V^h, o que indica que o uso da política ótima pode de fato trazer um maior retorno financeiro para as empresas.

7.4 A política ótima para o sistema de cobrança

Considere a função $Q^*(s_i, a_i)$, com ela é possível extrair a política ótima, a partir da Equação (9), para cada carteira.

Carteira	$(s_1,\ s_2,\ s_3,\ s_4,\ s_5)$
A	$(a_2,\ a_5,\ a_4,\ a_1,\ a_5)$
B	$(a_3,\ a_5,\ a_5,\ a_4,\ a_5)$
C	$(a_7,\ a_7,\ a_1,\ a_5,\ a_9)$
D	$(a_2,\ a_5,\ a_3,\ a_3,\ a_1)$
E	$(a_4,\ a_2,\ a_1,\ a_1,\ a_1)$
F	$(a_4,\ a_5,\ a_2,\ a_2,\ a_5)$
G	$(a_4,\ a_4,\ a_3,\ a_6,\ a_2)$

Tabela 7: Tabela com as políticas ótimas aprendidas para cada carteira. A segunda coluna representa a política para os 5 estados; a_i representa a melhor ação a ser tomada no estado s_i.

As políticas observadas na Tabela 7 são aquelas extraídas da função $Q^*(s_i, a_i)$ para cada carteira. Com isso, se a empresa se encontrar no estado s_i e tomar a ação a_i, é possível garantir que essa ação é a melhor que a empresa poderá tomar naquele momento.

8 Discussão sobre a modelagem aplicada

8.1 A dívida

A dívida possui uma idade, que nada mais é que o período em que um cliente se encontra numa lista de inadimplência. Numa dívida com idade baixa (menor que um ano), a empresa responsável pela loja se esforça para que o cliente venha a pagar, pois nela se cobra juros, multas e correções.

Entretanto, quanto maior a idade da dívida, a tendência é de que o retorno financeiro seja pequeno, quase nulo. Esse fato foi observado pela empresa de cobrança, que nos forneceu essa informação. Sendo assim, podemos montar um gráfico, figura 7, que represente esse comportamento das dívidas.

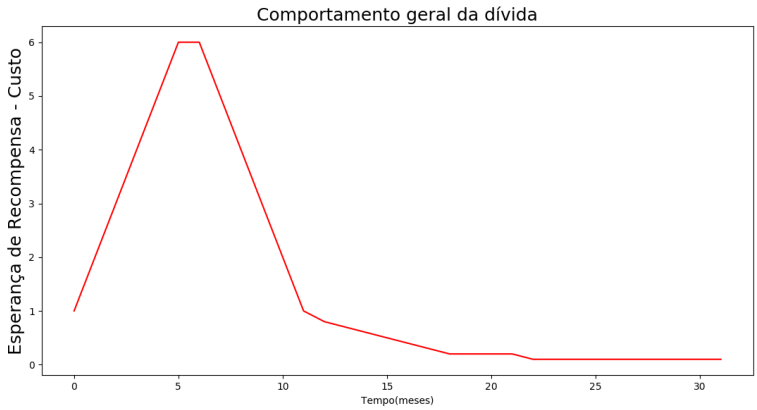


Figura 7: Comportamento do valor esperado da recompensa menos custo num período de tempo.

Neste gráfico podemos observar que a dívida inicia com um valor, no começo do processo sofre um aumento devido aos juros e depois o valor esperado começa a cair . Após um ano de dívida, o valor fica menor do que o inicial, até chegar numa fase onde esse valor fica quase nulo.

Sendo assim, toda empresa ao efetuar um planejamento financeiro contabiliza além dos custos e lucros próprios do negócio, os riscos envolvidos nele, isto é, a inadimplência. Toda transação financeira oferece um risco, quanto maior esse risco, maior o custo para "encará-lo", contudo maior é o retorno no caso de sucesso.

Porém, a inadimplência é um risco considerado como certo em negociações, deixando praticamente de ser um risco para se tornar um custo. Com isso, as

empresas investem em previsões em seus negócios. Dentro de seus planejamentos anuais já deixam separado um percentual de inadimplência, que é definido baseado em resultados anteriores, análises de mercado e projeções futuras, portanto, se a inadimplência daquele ano não ultrapassar a margem prevista, a empresa registrará lucro, pois irá cobrir as despesas geradas por esses inadimplentes, e essa carteira de inadimplentes poderá, no ano seguinte, ser trabalhada para recuperação destas receitas. E tudo o que retornar será lucro direto para a empresa, pois o custo destes clientes já foi coberto no ano anterior.

Alguns clientes conhecem esse processo e preferem ficar devendo por um período de tempo considerável (mais de um ano), pois sabem que existem diversas formas de desconto oferecidos para quitar suas dívidas e isso pode vir a ser vantajoso. Entretanto, as vezes eles precisam ter o nome ”limpo”para fazer empréstimos, negociações à prazo e com isso estão interessados em fazer uma negociação de dívidas antigas.

8.2 Método anterior de cobrança

A empresa de cobrança antigamente mandava *e-mails* de acordo com uma tabela referente à idade da dívida, nela constava quanto podia ser dado de desconto a cada cliente a partir da idade de sua dívida. Os cobradores procuravam o desconto mais baixo possível para fechar o acordo, similar com um comportamento ótimo. Contudo, encontrar o melhor desconto em cada caso é muito complicado para ser feito e por isso a empresa mudou a forma de cobrar, optando sempre por oferecer o maior desconto possível para dívidas antigas.

8.3 Considerações sobre a modelagem do MDP de cobrança

Na solução do problema, para formar o MDP descrito nas seções anteriores, consideramos o comportamento acumulativo natural dos acordos ao longo das campanhas para representar as recompensas, isto é, um acordo gerado hoje não necessariamente é referente às campanhas enviadas hoje, mas além delas, das anteriores.

Com isso, foi natural gerar a relação natural recompensa - acordo, entretando, como cada carteira possui uma quantidade de clientes com ordens de grandeza diferentes umas das outras, foi necessário efetuar uma normalização nas recompensas a cada campanha. Ela foi feita de forma simples, bastou dividir a quantidade de acordos do dia pela quantidade de clientes de determinada campanha. Podemos extender esse raciocínio às funções de custo ao longo das campanhas.

9 Próximos projetos

Como descrito na seção anterior, um possível projeto é o de aprender o desconto ótimo que pode ser dado ao longo de uma carteira, decidindo em qual momento se deve aumentar o desconto oferecido a cada cliente de acordo com a idade de sua dívida.

Também é possível aprender como proceder para cada tipo de cliente, isto é, será que variáveis como idade, região que mora, status (empregado, desempregado) dos clientes influenciam na geração dos acordos? O objetivo aqui se torna descobrir como lidar com cada um desses casos.

A empresa de cobrança se comunica através de diversas maneiras, como *e-mails*, *whatsapp*, mensagens de texto no celular, cartas, entre outros. Um possível aprendizado é o de descobrir quando fazer a transição entre as opções de comunicação para determinado cliente otimizando o retorno financeiro.

E por fim, um projeto que a própria Adimplere tem é o de aprender linguagem natural nos *e-mails*, e com isso, de acordo com respostas dos clientes automatizar totalmente o processo de negociação da dívida.

10 Conclusão

Nesse trabalho, foi modelado o problema de cobrança como um processo de decisão markoviano e foi usada a técnica de Aprendizado por Reforço em Lote para o aprendizado da melhor política de cobranças a partir de dados reais fornecidos pela empresa Adimplere.

Considerando as experiências fornecidas sobre campanhas passadas, foi possível encontrar políticas ótimas para o processo de cobrança. A análise do processo de aprendizado sugeriu as seguintes conclusões:

1. Há uma similaridade entre as linhas da Figura 6, que representam os valores $V^h(s_0)$ e $V^*(s_0)$, indicando que o aprendizado foi não-viesado e de fato é uma boa estimativa da função V.

2. Foi possível observar que a hipótese feita pela Adimplere: "campanhas com informações personalizadas geram mais acordos", é verdadeira quando realizadas após o envio de um certo número de campanhas genéricas ou não-personalizadas, o que reflete uma certa demora na respostas do cliente às campanhas realizadas pela Adimplere.

3. Estados com maiores quantidades de clientes são os que possuem os maiores valores de V^h e V^*, o que indica que o uso da política ótima pode de fato trazer um maior retorno financeiro para as empresas.

Apesar da quantidade limitada de dados, espera-se que ao extendermos esse processo incluindo experiências futuras sobre novas campanhas, utilizando a melhor política aprendida neste trabalho, o processo de aprendizado apresentarão medidas melhores de Erro e Dispersão com o passar do tempo.

Observamos também uma quantidade enorme de aplicações distintas de BRL nesse mundo de cobranças, por exemplo, com mais experiências e um modelo mais detalhado de estados, recompensas e custos, ou até mesmo para outras abordagens de cobrança.

Referências

[1] KLEIN, D.; ABBEEL, P., Lecture 8: MDPs I. Disponível em: http://ai.berkeley.edu/lecture_slides.html. Acesso em: 20 jun. 2017.

[2] KLEIN, D.; ABBEEL, P., Lecture 9: MDPs II. Disponível em: http://ai.berkeley.edu/lecture_slides.html. Acesso em: 20 jun. 2017.

[3] COELHO, L. Processo de decisão markoviano e aprendizagem por reforço. Technical report, Laboratório de Técnicas Inteligentes. Disponível em: <http://www.lti.pcs.usp.br/> USP, 2011. Acesso em: 17 ago. 2017.

[4] M. L. Puterman. Markov Decision Processes—Discrete Stochastic Dynamic Programming. John Wiley Sons, Inc., 1994.

[5] MAUÁ, D. D., Sequential Decision Making, Disponível em: https://www.ime.usp.br/~ddm/mac425/aulas/mdp.pdf. Acesso em: 20 jun. 2017.

[6] LACERDA, D. A., Batch reinforcement learning: a case study for the problem of decision making in sales processes. In: Biblioteca Digital USP. Disponível em: <http://www.teses.usp.br/teses/disponiveis/45/45134/tde-03072014-101251/pt-br.php> . Acesso em: 10 mar 2017.

[7] LACERDA, Denis Antonio; DE BARROS, Leliane Nunes. Aprendizado por reforço em lote para o problema de tomada de decisão em processos de venda.

[8] MANNION, P.; DUGGAN, J.; HOWLEY, E. Parallel Reinforcement Learning for Traffic Signal Control. In: INTERNATIONAL WORKSHOP ON AGENT-BASED MOBILITY, TRAFFIC AND TRANSPORTATION MODELS, METHODOLOGIES AND APPLICATIONS (ABMTRANS), 4., 2015, Londres. Artigo, Londres:Elsevier, 2015. p. 956-961

[9] REINFORCEMENT learning. In: "Wikipédia: the free encyclopedia". Disponível em: <https://en.wikipedia.org/wiki/Reinforcement_learning> Acesso em: 20 jun 2017.

[10] DE BARROS, N. L., Reinforcement Learning I. In: "MAC0425/5739 Inteligência Artificial - PACA 1sem2017". Disponível em: <http://paca.ime.usp.br/pluginfile.php/107189/course/section/17629/Aula-ReinforcementLearning-2017.pdf> Acesso em: 20 jun 2017.

[11] R.S. Sutton and A.G. Barto. Reinforcement Learning: An Introduction. Adaptive Computation

[12] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. Machine Learning, 8(3-4):279–292, 1992.

[13] Damien Ernst, Pierre Geurts, Louis Wehenkel, and L. Littman. Tree-based batch mode reinfor- cement learning. Journal of Machine Learning Research, 6:503–556, 2005. and Machine Learning. Mit Press, 1998. ISBN 9780262193986.

[14] R. E. Bellman. Dynamic Programming. Princeton University Press, USA, 1957.

[15] ANTOS, A., Munos, R., Szepesvari, C.: Fitted Q-iteration in continuous action-space MDPs. In: Advances in Neural Information Processing Systems, vol. 20, pp. 9–16 (2008).

[16] Bonarini, A., Caccia, C., Lazaric, A., Restelli, M.: Batch reinforcement learning for controlling a mobile wheeled pendulum robot. In: IFIP AI, pp. 151–160 (2008).