Aprendizado por reforço em lote: um estudo de caso para o problema de tomada de decisão em processos de venda

Denis Antonio Lacerda

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
COMO REQUISITO PARCIAL PARA OBTENÇÃO
DO TÍTULO DE MESTRE
EM CIÊNCIAS

Área de concentração: Ciência da Computação

Orientadora: Prof.ª Dr.ª Leliane Nunes de Barros

— São Paulo, Agosto de 2013 —

Resumo

Planejamento Probabilístico estuda os problemas de tomada de decisão sequencial de um agente (robótico ou de softtware), em que as ações possuem efeitos probabilísticos, e que podem ser modelados como um processo de decisão markoviano (*Markov Decision Process* - MDP). Num MDP o ambiente é modelado como um conjunto de estados sendo que, um agente, ao executar uma ação num determinado estado é levado para um outro estado de acordo com uma dada função de transição probabilística, recebendo do ambiente uma recompensa. Dadas as probabilidades e os valores de recompensa das ações, é possível determinar uma política de ações (i.e., um mapeamento entre estado do ambiente e ações do agente) que maximiza a recompensa esperado acumulada pela sequência de ações.

Nos casos em que o agente não possui conhecimento prévio do ambiente, a melhor política precisa ser aprendida através da interação do agente com o ambiente real. Este processo é chamado de *aprendizado por reforço*. Algumas das limitações conhecidas do aprendizado por reforço são: (1) necessidade de muitas interações com o ambiente real para encontrar a política ótima; (2) comportamento instável em algumas situações; (3) inviável em sistemas em que as ações possuem um custo muito elevado.

Neste trabalho, estudamos técnicas de *aprendizado por reforço em lote* (*Bach Reinforcement Learning* - BRL), em que o aprendizado é feito usando um histórico de interações prévias, armazenadas em um banco de dados de processos. Como um estudo de caso, aplicamos essa técnica no aprendizado de políticas para o processo de venda de impressoras, cujo objetivo é a construção de um sistema de recomendação de ações para vendedores iniciantes.

Palavras-chave: Planejamento Probabilístico, Processo de Decisão Markoviano, Aprendizado por Reforço, Aprendizado por Reforço em Lote, Aprendizado de Processos de Venda

Sumário

1 Introdução **1**

2 Planejamento probabilístico **5**

2.1 Planejamento automático 5

2.2 Planejamento probabilístico 6

2.3 Processos markovianos de decisão 6

2.3.1 Política ótima 8

2.3.2 Algoritmos 9

3 Aprendizado por Reforço **13**

3.1 Definição 13

3.2 Métodos de Resolução 15

3.3 Q-Learning 15

4 Aprendizado por Reforço em Banco de Dados **19**

4.1 Aprendizado por reforço em lote (Batch Reinforcement Learning - BRL) 19

4.2 Fundamentos dos Algoritmos de BRL 20

4.3 Fitted Q Iteration 21

5 Estudo de caso: O Problema do Vendedor de Impressoras **25**

5.1 O Processo de Venda 25

5.2 Impressoras Plotters 27

5.3 Sistema de Acompanhamento de Negociações (Forecast) 28

5.3.1 Estágios do processo de negociação 29
5.4 Sistema de Tomada de Decisão 31
5.4.1 Identificando as Variáveis de Estado 32

6 Modelando o Problema do Vendedor de Impressoras como um MDP 33

7 Resultados Experimentais 43
7.1 Avaliação da política aprendida 43
7.1.1 Teste de convergência 43
7.1.2 Aprendizado por reforço em lote vesus Iteração de Valor 44
7.1.3 Política dos vendedores versus política aprendida 45
7.2 Adaptação às variações do ambiente 48
7.3 Horizonte infinito versus Horizonte finito 48
7.4 Propriedades de convergência 51
7.5 Análise de convergência do aprendizado por reforço em lote com dados simulados 52
7.6 Aprendizado por reforço em Lote com atualizações progressivas versus regressivas 53

8 Conclusão 57

A Aprendizado Supervisionado 59
A.1 Introdução 59
A.2 Aprendizado supervisionado 60
A.3 Aprendizado não-supervisionado 61
A.4 Validação dos sistemas de aprendizado 62
A.5 Taxa de erro 62
A.6 Estimativa da taxa de erro real 63
A.6.1 Treinar e testar 63
A.6.2 Reamostragem 63
A.6.3 Validação cruzada (Cross Validation) 64

Referências Bibliográficas 65

Índice Remissivo **68**

Lista de Figuras

2.1 Modelo conceitual da interação entre o planejador, o controlador e o sistema (ambiente) que será controlado. A construção de um agente envolve os módulos planejador e controlador. 6

3.1 Modelo teórico de aprendizado por reforço apresentado por [Sutton and Barto, 1998]. 14

4.1 Problema geral do aprendizado por reforço em lote [Sascha Lange and Riedmiller, 2012]. As três fases distintas: 1) Coleta de transições com uma estratégia de amostragem arbitrária. 2) Aplicação de algoritmos de aprendizagem (em lote), a fim de aprender a melhor política possível a partir do conjunto de transições. 3: Aplicação da política aprendido. Nesta abordagem a exploração não é parte da tarefa de aprendizagem. Durante a fase de aplicação, que também não faz parte da tarefa de aprendizagem, as políticas não são melhoradas. 20

4.2 Classificação dos algoritmos de Aprendizado por reforço batch vs. não-batch de acordo com a perspectiva de uso dos dados e interação ([Sascha Lange and Riedmiller, 2012]). 23

5.1 Ciclo de Vendas: etapas básicas de um processo de venda. 27

5.2 Ciclo de Vendas do Programa Big Impression. Na etapa 08 (Standby) o vendedor aguarda por um tempo antes de retomar o processo de negociação. 29

7.1 Avaliação dos algoritmos de Aprendizado BRL vs. IV usando Validação Cruzada adaptada com k=5 45

7.2 Avaliação dos algoritmos de Aprendizado BRL vs. IV usando Validação Cruzada adaptada com k=10 46

7.3 Adaptação à política dos vendedores ao longo dos anos. 49

7.4 Adaptação à política dos vendedores ao longo dos anos. 49

7.5 Adaptação à política dos vendedores ao longo dos anos. 50

7.6 Adaptação à política dos vendedores ao longo dos anos. 50

7.7 Avaliação dos algoritmos de Aprendizado BRL vs. IV modelando o problema como um MDP de horizonte finito ou infinito. 52

7.8 Teste de convergência dos algoritmos de Aprendizado BRL vs. IV modelando o problema como um MDP de horizonte finito. 53

7.9 Teste de convergência do algoritmo de Aprendizado por Reforço em Lote usando dados simulados. 54

7.10 Teste de convergência do algoritmo de Aprendizado por Reforço em Lote usando dados simulados. 55

A.1 Tipos de aprendizado. 60

A.2 Cross Validation. Método K-fold 64

Capítulo 1

Introdução

Planejamento Probabilístico é a área que trata problemas de tomada de decisão sequencial em que as ações possuem efeitos probabilísticos. No planejamento probabilístico os problemas são normalmente modelados como um processo de decisão markoviano (Markov Decision Process - MDP) [Bellman, 1957]. Um MDP modela a interação entre um agente e um ambiente como um processo de tempo discreto. A ação executada por um agente em um dado estado possuem efeitos probabilísticos e leva o agente para outro estado com uma certa probabilidade. Para cada ação executada o agente recebe uma recompensa e o objetivo do agente é maximizar a recompensa esperada sobre uma sequência de ações.

O objetivo de um planejador probabilístico é encontrar uma *política ótima*, isto é, uma função que mapeia para cada estado do sistema qual ação o agente deve escolher, de forma que a recompensa esperada seja máxima. Para isso, o agente precisa levar em consideração o *horizonte*, isto é, o número limite de passos (escolhas de ações) disponíveis para o agente. O horizonte pode ser: *finito* quando o número de ações que o agente deve tomar é fixo e conhecido; *infinito* quando a tomada de ações deve ser feita repetidamente e sem possibilidade de parar ou *indefinido* semelhante ao finito porém, o número de ações a serem tomadas é desconhecido.

No *planejamento probabilístico*, nem sempre é possível ter uma descrição completa do problema e construir uma política baseando-se no conhecimento completo do ambiente. Neste caso, o agente precisa aprender a melhor política através de tentativa e erro usando suas interações com o ambiente [Kaelbling et al., 1996]. O nome que se dá a esse processo é *aprendizado por reforço* [Sutton and Barto, 1998].

Algoritmos básicos de aprendizado por reforço, como Q-learning [Watkins and Dayan, 1992], geralmente precisam de muitas interações até a convergência para boas políticas, tornando sua aplicação para problemas reais impossíveis. Além disso, há situações críticas em que o agente não pode interagir com o ambiente durante o processo de aprendizado, e para isso precisa aprender baseando-se em um número fixo de amostras de interações do passado conhecidas *a priori*. Para estas situações são utilizadas adaptações das técnicas de aprendizado por reforço conhecidas como aprendizado por reforço em lote (Batch Reinforcement Learning - BRL) [Ernst et al., 2005]. Os algoritmos modernos de Aprendizado por Reforço em Lote também tratam problemas conhecidos do Aprendizado por Reforço, garantindo estabilidade e aproximação da política ótima (ou V^*).

Um exemplo interessante em que é possível aplicar técnicas de planejamento probabilístico e aprendizado por reforço é o problema do **Vendedor de Impressoras**. Neste problema, uma série de ações sequenciais são executadas por um vendedor com o objetivo de influenciar o cliente à compra. O resultado das ações executadas é incerto e depende de inúmeros fatores, em geral desconhecidos pelo vendedor. No entanto, com acesso a um banco de dados de experiências anteriores é possível aplicar técnicas de aprendizado por reforço em lote para a construção de um sistema de suporte a tomada de decisão num processo de vendas.

Para aplicar métodos de aprendizado em problemas reais, como processos de venda, modelados como MDP, algumas questões precisam ser respondidas e alguns problemas precisam ser tratados:

- O problema a ser aprendido possui uma função de transição *estacionária* ou *não-estacionária*? Por exemplo, os dados sofrem alterações com o tempo?
- Como validar o sistema de aprendizado criado?
- Qual é a quantidade de esperiências anteriores necessárias para aprender uma política ótima?

No Problema do Vendedor de Impressoras as informações coletadas, e usadas no processo de aprendizado, são provenientes tanto de vendedores experientes quanto de vendedores inexperientes. O sistema precisa ser capaz de aprender com os casos de sucesso, mas também com os casos de fracasso, e após o aprendizado é desejável que seja possível identificar os vendedores que precisam ser treinados. No problema do vendedor de impressoras o cliente é modelado

apenas pela região onde reside e como pessoa física ou jurídica. Podemos assumir que as outras características do cliente, pertinentes para a decisão, são modeladas pela função de transição de estados aplicando-se as ações dos vendedores?

Neste trabalho utilizamos técnicas de aprendizado por reforço em lote e planejamento para aperfeiçoar processos de vendas, fornecendo políticas que minimizam o custo do processo e maximizam o sucesso das vendas. Para isso utilizamos um banco de dados com histórico de negociações de impressoras de grande formato (*plotters*). A base de dados pertence ao Programa Big Impression, grupo de revendas e parceiros localizados em todo território brasileiro, e que conta com o registro de mais de 20 mil negociações.

Capítulo 2

Planejamento probabilístico

2.1 Planejamento automático

Planejamento é um processo de escolha de sequencias de ações através da previsão de seus efeitos com o intuito de satisfazer objetivos previamente definidos. Planejamento automático é a sub-área da *Inteligência Artificial* que estuda métodos para automatizar esse processo através de algoritmos independentes de domínio. Um problema de planejamento pode ser descrito formalmente pela tupla $\mathcal{P} = (\mathcal{S}, s_0, S_G, \mathcal{A}, E, T, \mathcal{O}, O)$ [Ghallab et al., 2004] onde:

- $\mathcal{S}$ é um conjunto de estados,
- $s_0 \in \mathcal{S}$ é o estado inicial,
- $S_G \subset \mathcal{S}$ é o conjunto (não vazio) de estados meta,
- $\mathcal{A}$ é o conjunto de ações sendo $\mathcal{A}(s)$ o conjunto de ações aplicáveis no estado s,
- E é o conjunto de eventos exógenos, ou seja, ações externas que não podem ser controladas pelo planejador,
- $T : \mathcal{S}\text{x}\mathcal{A}\text{x}E \rightarrow 2^{\mathcal{S}}$ é a função de transição de estados sendo que para cada estado $s \in \mathcal{S}$, ação $a \in \mathcal{A}(s)$ e evento $e \in E$, devolve um subconjunto $S_r \subset \mathcal{S}$ que representa os possíveis estados resultantes,
- $\mathcal{O}$ é o conjunto de observações que o agente pode fazer do ambiente,

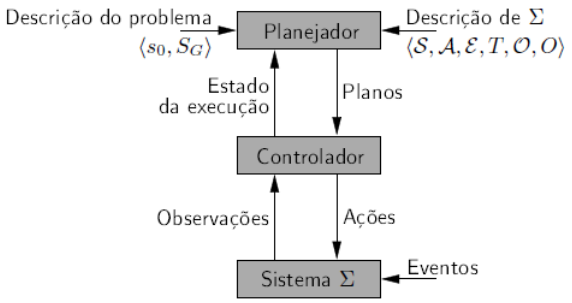

Figura 2.1: Modelo conceitual da interação entre o planejador, o controlador e o sistema (ambiente) que será controlado. A construção de um agente envolve os módulos planejador e controlador.

- $O : \mathcal{S} \rightarrow 2^{\mathcal{O}}$ é a função de observação para cada estado $s \in \mathcal{S}$. Se O é a função identidade, dizemos que o agente possui *observabilidade total do estado*, caso contrário, ele possui *observabilidade parcial*.

Note que a função de transição T define 2 tipos possíveis de planejamento: (i) planejamento determinístico ($T : \mathcal{S}\text{x}\mathcal{A}\text{x}E \rightarrow \mathcal{S}$) também chamado de *planejamento clássico*; (ii) planejamento não-determinístico ($T : \mathcal{S}\text{x}\mathcal{A}\text{x}E \rightarrow 2^{\mathcal{S}}$) em que a ação pode levar o agente para um dos estados de um subconjunto de S, $s_j \in S$, sem ser declarada uma preferência entre eles.

2.2 Planejamento probabilístico

No planejamento probabilístico o modelo de transição de estados é dado em termos de distribuição de probabilidades. Ou seja, para cada estado s do modelo e ação a existe uma distribuição de probabilidade sobre o subconjunto de estados alcançáveis S_j.

2.3 Processos markovianos de decisão

Um processo de decisão markoviano (Markov Decision Process - MDP) [Bellman, 1957] [Russell and Norvig, 2003] é um modelo matemático usado para modelar tomada de decisões sequenciais onde uma ação tem de ser escolhida pelo agente em cada estado. A execução de uma ação

levará a um estado e uma recompensa. O objetivo do agente é maximizar seu ganho com o passar do tempo.

Num MDP os efeitos das ações são probabilísticos e obedecem a propriedade de Markov: o efeito de uma ação depende apenas do estado atual do sistema e não de como o processo chegou em tal estado.

Um MDP pode ser definido formalmente como uma tupla $\mathcal{M} = (S, A, E, P, R)$ onde:

- S é um conjunto finito de estados. Se o processo tem conhecimento total do estado atual, então o MDP é chamado de *totalmente observável*, caso contrário, é *parcialmente observável*.
- A é um conjunto finito de ações que podem ser executadas pelo agente e permitem que o sistema mude de estado.
- E é o conjunto de eventos exógenos, ou seja, ações externas que não podem ser controladas pelo planejador,
- P define as probabilidades de transição sobre S, sendo $P(s'|s, a)$ a probabilidade condicional de ir para o estado $s' \in S$, dado que o agente está no estado $s \in S$, executa a ação $a \in A$ e ocorre o evento exógeno $e \in E$. Note que as transições são markovianas pois a probabilidade de alcançar s' a partir de s depende apenas de s, ou seja, independe dos estados anteriores,
- $R : S\text{x}A \rightarrow \mathbb{R}$ é a função de recompensa, sendo que R(i,a) representa a recompensa obtida pelo agente dado que ele está no estado $i \in S$ e executa a ação $a \in A$.

Definição 2.1. *O* horizonte h *de um MDP repesenta o número limite de passos (escolhas de ações) disponíveis para o agente. O horizonte pode ser* finito *quando o número de ações que o agente deve tomar é fixo,* infinito *quando a tomada de ações deve ser feita repetidamente e sem possibilidade de parar ou* indefinido *semelhante ao infinito porém, com possibilidade de parar em estados meta.*

Definição 2.2. *Uma* política *de um MDP é uma função* $\pi : S \rightarrow A$ *que mapeia para cada estado do sistema qual ação o agente deve escolher. Em geral, resolver um processo de decisão se resume a encontrar boas políticas que otimizam de alguma forma um critério de desempenho (atingimento de metas, recompensa recebida, etc).*

Uma política pode ser classificada como:

- *Total, se é definida para todos os estados do MDP;*
- *Parcial, se é definida apenas para alguns estados do MDP.*

Uma política também pode ser classificada como:

- Estacionária*, se a ação recomendada para um dado estado $s \in S$ independe do momento de decisão,*
- Não-estacionária*, se a ação recomendada pode variar durante o processo.*

Definição 2.3. *A função valor de uma política π para um MDP $\mathcal{M} = (S, A, P, R)$ é uma função $V^\pi : S \to \mathbb{R}$ em que $V^\pi(s)$ devolve o valor esperado da recompensa para esta política a partir do estado $s \in S$.*

Por exemplo, para horizonte infinito com desconto e política estacionária π,

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, a) V^\pi(s') \tag{2.1}$$

onde γ representa o fator de desconto.

Dados um estado $s \in S$, uma ação $a \in A$ e uma política π para um MDP, pode-se definir o valor da ação a no estado s considerando a recompensa imediata de a e a recompensa esperada após a. Essa função denotada por Q é definida como:

$$Q^\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V^\pi(s') \tag{2.2}$$

2.3.1 Política ótima

Para a função valor de uma política, o processo de decisão é determinar π^* tal que $V^{\pi^*}(s) \geq V^\pi(s)$ para todo π e para todo s. Chamamos π^* de *política ótima*.

A função valor ótima V^* é a função valor associada a qualquer política ótima. Sendo assim, para um agente que deseja maximizar a sua recompensa esperada ótima, satisfaz a seguinte

igualdade [Bellman, 1957]:

$$V^*(s) = \max_{a \in A}\{R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V^*(s')\} \tag{2.3}$$

A Equação 2.3 é chamada de *Equação de Bellman* [Puterman, 1994] e os principais algoritmos para encontrar uma política ótima se baseiam nesta equação.

2.3.2 Algoritmos

Existe uma grande quantidade de algoritmos para a solução de MDPs que procuram uma política ótima. Os mais comuns, descritos a seguir, trabalham diretamente com políticas ou com a função valor.

Iteração de Valor

O algoritmo de iteração de valor [Puterman, 1994] (Algoritmo 1) usa programação dinâmica para determinar o valor de $V^*(s)$ para cada estado s do MDP calculando de forma iterativa $V_t(s)$. $V_t(s)$ pode ser interpretado como o valor esperado da recompensa que o agente pode obter t estágios para frente a partir do estado s. O algoritmo inicia com valores V_0 aleatórios. Em cada iteração i calcula-se V_i baseado na função valor V_{i-1}. O algoritmo também faz uso da função $Q(s,a)$ que representa o valor da ação a aplicada no estado s no passo, ou estágio, t, dada por:

$$Q_t(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V_{t-1}(s') \tag{2.4}$$

E o valor do estado s no estágio de decisão t é igual ao valor da melhor ação:

$$V_t(s) = \max_{a \in A} Q_t(s,a) \tag{2.5}$$

O critério de parada do algoritmo para horizonte finito é que o número de iterações seja igual ao horizonte do problema. No caso de horizonte finito, o algoritmo gera uma política para cada estágio de decisão (política não-estacionária). A complexidade do algoritmo é $O(h|A||S|^2)$.

Algoritmo 1: IteracaoDeValor(M, ϵ)

Entrada: M: um MDP, ϵ: erro máximo permitido
Saída: π^*: uma política ótima

para cada $s \in S$ **hacer**
  $V_0(s) \leftarrow$ *Valor Aleatório*;
fin
$t \leftarrow 1$

para cada $s \in S$ **hacer**
  $Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V_{t-1}(s')$;
fin

repita
  para cada $s \in S$ **hacer**
    para cada $a \in A$ **hacer**
      $Q^t(s,a) \leftarrow R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V_{t-1}(s')$
    fin
    $V^t(s) \leftarrow \max_{a \in A} Q^t(s,a)$
    $\pi(s) \leftarrow arg \max_{a \in A} Q^t(s,a)$
  fin
  t ← t+1
até $\{V_t(s) - V_{t-1}(s)\} < \epsilon \; \forall s \in S$;
retorna π.

Para processos com horizonte infinito (ou indefinido) o algoritmo deve parar apenas após a convergência do valor de cada estado, ou seja, quando $\max s \in S\{V_t(s) - V_{t-1}(s)\} < \epsilon$ para algum ϵ suficientemente pequeno, neste caso, tem-se uma política estacionária.

Soluções eficientes

A complexidade do algoritmo de Iteração de Valor é polinomial no número de estados, no entanto, o número de estados cresce exponencialmente com o número de variáveis de estado. Além disso, o algoritmo de Iteração de Valor é *off-line*, ou seja, calcula a política completa (para todos os estados do sistema) antes do agente sair executando as ações. Isso faz com que os algoritmos apresentados sejam pouco úteis em sistemas grandes e que necessitem de tomadas de decisão em tempo real.

Existem diversos algoritmos para resolver MDPs de forma assíncrona e de maneira mais eficiente. Em geral estes algoritmos partem do mesmo princípio dos dois algoritmos apresentados porém resolvem o problema de forma relaxada. Isso faz com que o sistema perca a solução ótima mas permita encontrar uma política *anytime* com menor tempo de convergência. Exemplos de

algoritmos eficientes para resolver MDPs são:

- **Programação Dinâmica em Tempo Real (RTDP)** [Barto et al., 1993] Apresenta uma tática eficiente para atualização assíncrona dos valores dos estados. A ideia básica do algoritmo é simular inúmeras vezes a execução da política sorteando o próximo estado de acordo com sua probabilidade, e atualizar o valor dos estados ao mesmo tempo em que a política é atualizada. Esta maneira de escolher quais estados terão seu valor atualizado é muito poderosa porque os estados que influenciam mais no valor da esperança são atualizados mais vezes.

- **Programação Dinâmica em Tempo Real com Marcações (LRTDP)** [Bonet and Geffner, 2003]. No RTDP, escolher qual estado s terá seu valor atualizado de acordo com a probabilidade dele ser alcançado faz com que $V^{\pi}(s)$ se aproxime rápido de $V^{*}(s)$. No entanto, estados com pouca probabilidade de serem alcançados demoram para convergir. O LRTDP aumenta a velocidade de convergência do RTDP marcando como resolvidos estados cujos valores já convergiram.

Capítulo 3

Aprendizado por Reforço

Aprendizado por reforço é aplicado em problemas de tomada de decisão sequencial que podem ser modelados com MDPs.

No *planejamento probabilístico* (Capítulo 2) nem sempre é possível ter uma descrição completa do problema e construir uma política baseando-se no conhecimento prévio do ambiente. O ambiente pode ser parcialmente desconhecido, completamente desconhecido ou dinâmico. Neste caso, o agente precisa aprender a melhor política através de tentativa e erro usando suas interações com o ambiente [Kaelbling et al., 1996]. O nome que se dá a esse processo é *aprendizado por reforço* [Sutton and Barto, 1998].

3.1 Definição

Aprendizado por Reforço (AR) consiste em fazer um agente escolher suas ações se baseando na sua interação com o ambiente. Diferentemente do aprendizado supervisionado em que existe um professor que diz ao agente qual deveria ter sido a ação correta para cada estado, no aprendizado por reforço existe apenas uma medida que indica o quão boa, ou ruim, foi a ação escolhida mas não diz exatamente qual teria sido a melhor ação. Esse tipo de aprendizado foi inspirado no processo de aprendizado infantil. Uma criança costuma realizar ações aleatórias e aprende quais ações são boas ou ruins de acordo com o *feedback* de seus pais.

Formalmente, no problema de aprendizado por reforço temos um agente que atua em um ambiente. O agente percebe um conjunto discreto de estados S, pode realizar um conjunto

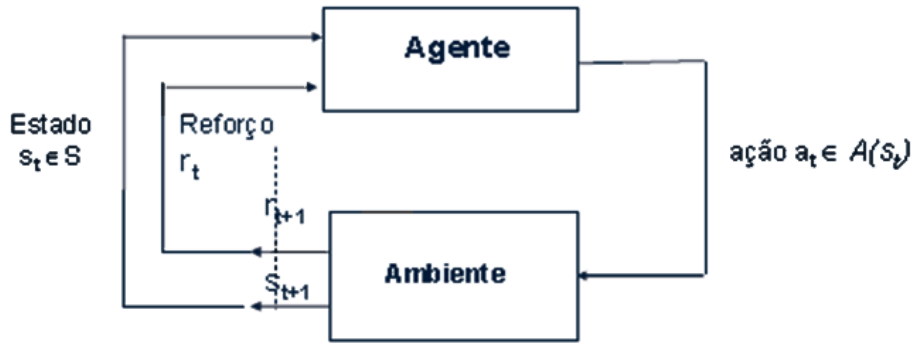

Figura 3.1: Modelo teórico de aprendizado por reforço apresentado por [Sutton and Barto, 1998].

discreto de ações A, e em cada instante de tempo t o agente conhece seu estado atual $s \in S$ e escolhe uma ação a ser executada que o leva para um outro estado s'. Para cada par estado/ação, (s, a), há um sinal de reforço $R(s, a)$ que é dado ao agente quando ele executa a ação a no estado s. O relacionamento agente/ambiente é ilustrado na Figura 3.1.

O problema consiste em escolher uma política de ações que maximize o total de recompensas recebidas pelo agente após n interações com o ambiente. Um agente pode seguir várias políticas de ações, mas o objetivo é calcular a política que maximize a soma das recompensas futuras, ou seja, o total de recompensas recebidas após a adoção dessa política. Essa política é denominada *política ótima*.

Como é possível perceber, a descrição de um problema de *aprendizado por reforço* é muito parecida com a descrição de um *processo markoviano de decisão (MDP)* descrito no Capítulo 2. A única diferença é que no *aprendizado por reforço* as transições entre os estados são desconhecidas, e a recompensa é chamada de *sinal de reforço*. Sendo assim, é possível resolver os problemas de AR com algumas adaptações nos algoritmos que resolvem MDPs.

3.2 Métodos de Resolução

Para resolver um problema de aprendizado por reforço é preciso atentar-se a algumas características do *aprendizado por reforço*:

- **Aprendizado por interação:** O agente age no ambiente e espera pelo valor do reforço que o ambiente devolve.
- **Retorno atrasado:** um valor de reforço alto recebido pelo agente ao escolher uma ação em um determinado estado não significa que a ação escolhida é recomendada. Uma ação é produto de uma decisão local no ambiente, mas o intuito do agente é alcançar objetivos globais. A qualidade das ações deve ser vista pelas soluções a longo prazo.
- **Orientado pelo objetivo:** no aprendizado por reforço não é necessário conhecer detalhes da modelagem do ambiente. Simplesmente existe um agente que age neste ambiente desconhecido tentando alcançar um objetivo.
- **Exploração versus Exploitação:** consiste na decisão entre aprender mais sobre as partes desconhecidas do ambiente (*exploração*) ou aperfeiçoar o conhecimento do ambiente já conhecido (*exploitação*). Há casos em que o agente precisa explorar o ambiente para encontrar recompensas globais altas ou evitar becos sem saída. Mas também há casos onde é primordial fazer boas escolhas locais. Para que um sistema seja realmente autônomo, a decisão entre a exploração ou exploitação deve ser tomada pelo agente. Uma boa estratégia é mesclar os dois modos.

3.3 Q-Learning

Q-Learning é um método proposto por [Watkins and Dayan, 1992] para resolver problemas de aprendizado por reforço. Este método estima de forma iterativa uma função $Q(s, a)$ que determina a esperança de recompensas futuras quando o agente executa a ação a no estado s e continua agindo de acordo com a política aprendida daí em diante. Como a soma das recompensas futuras pode ser infinita, é usado um fator de desconto γ entre $]0, 1[$. Quanto menor este fator mais valor será dado às recompensas próximas do estado atual. Dizemos que a função Q define a soma das recompensas futuras descontadas.

Uma vez estimada a função Q, a política a ser seguida pelo agente pode ser facilmente determinada. Para maximizar a recompensa esperada basta o agente escolher em cada estado s a ação a que tem maior valor de Q(s,a).

Entretanto, seguindo essa abordagem, o agente perde uma parte de sua capacidade de aprender. Como em cada um dos estados é sempre executada a ação de maior valor de Q, apenas essa ação terá seu valor de atualizado. Dessa forma torna-se necessário intercalar fases de exploração e exploitação durante o processo de aprendizado.

É importante fazer uma ponderação entre o valor das ações que foi aprendido em experiências anteriores com o valor que será aprendido em experiências futuras. Em ambientes não determinísticos isto evita que uma recompensa obtida ao acaso muito diferente da média real tenha um peso alto na atualização da política. Ao mesmo tempo evita que o resultado de experiências distantes interfira na política em ambientes dinâmicos. Para isso é usado um fator α que poder ser compreendido como um *fator de aprendizado* ou *fator de esquecimento*.

O Algoritmo 2 mostra um exemplo de implementação do algoritmo Q-Learning. Uma propriedade deste algoritmo é que a convergência dos valores de Q para Q^*(valor ótimo) é garantida desde que o par (estado, ação) sejam visitados muitas vezes [Mitchell, 1997].

Algoritmo 2: Q-Learning (M, s_0, α, γ)

Entrada: M: um modelo, s_0: um estado inicial,
α: fator de aprendizado, γ: fator de desconto

para cada $s \in S$ **hacer**
  para cada $a \in A$ **hacer**
    $Q_0(s, a) \leftarrow$ valor aleatório
  fin
fin

$t \leftarrow 0$;
$a_0 \leftarrow arg \max_{a' \in A} Q_0(s, a')$

enquanto *critério de parada não é satisfeito* **faça**
  $[s_{t+1}, r] \leftarrow executaAcao(s_t, a_t)$
  $a_{t+1} \leftarrow arg \max_{a' \in A} Q_t(s_{t+1}, a')$
  $V_t(s_{t+1}) \leftarrow \max_{a' \in A} Q_t(s_{t+1}, a')$
  $Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha[r + \gamma V_t(s_{t+1}) - Q_t(s_t, a_t)]$
  $t \leftarrow t + 1$
fim

Uma estratégia bastante utilizada em implementações do algoritmo *Q-Learning* é a exploração aleatória denominada $\epsilon - Greedy$ em que o agente executa a ação com maior valor de Q com probabilidade $1 - \epsilon$ e escolhe uma ação aleatória com probabilidade ϵ.

Capítulo 4

Aprendizado por Reforço em Banco de Dados

Algoritmos básicos de aprendizado por reforço, como Q-learning, geralmente precisam de muitas interações até a convergência para boas políticas, tornando sua aplicação para problemas reais impossíveis. Além disso, há situações críticas em que o agente não pode interagir com o ambiente durante o processo de aprendizado, e para isso precisa aprender baseando-se em um número fixo de amostras de interações do passado conhecidas *a priori*. Para estas situações são utilizadas adaptações das técnicas de aprendizado por reforço conhecidas como aprendizado por reforço em lote (Batch Reinforcement Learning - BRL) [Ernst et al., 2005].

4.1 Aprendizado por reforço em lote (Batch Reinforcement Learning - BRL)

Aprendizado por Reforço em Lote é uma sub-área da programação dinâmica, baseada no aprendizado por reforço, usada para descrever o processo de aprendizado por reforço em casos onde a quantidade total de experiências usadas no aprendizado, geralmente um conjunto (lote) de amostras de transições, é fixa e dada *a priori* [Ernst et al., 2005].

No caso mais geral de um problema de aprendizado por reforço em lote o planejador não pode fazer suposições sobre o procedimento de amostragem das transições. As transições podem ser obtidas usando estratégias arbitrárias e não precisa necessariamente conter amostras

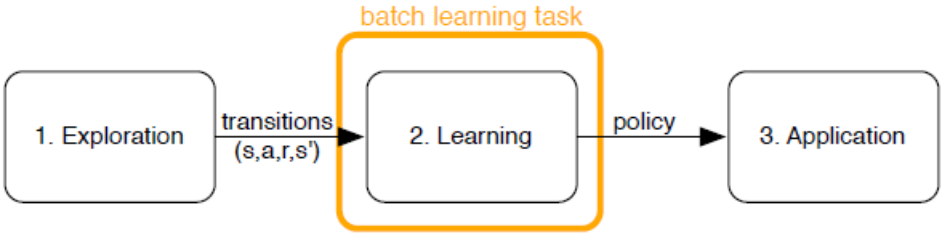

Figura 4.1: Problema geral do aprendizado por reforço em lote [Sascha Lange and Riedmiller, 2012]. As três fases distintas: 1) Coleta de transições com uma estratégia de amostragem arbitrária. 2) Aplicação de algoritmos de aprendizagem (em lote), a fim de aprender a melhor política possível a partir do conjunto de transições. 3: Aplicação da política aprendido. Nesta abordagem a exploração não é parte da tarefa de aprendizagem. Durante a fase de aplicação, que também não faz parte da tarefa de aprendizagem, as políticas não são melhoradas.

uniformes do espaço estado/ação. Elas não precisam ser coletadas ao longo de trajetórias ligadas. Utilizando apenas essas informações, o planejador tem que ser capaz de desenvolver a política que será usada pelo agente para interagir com o ambiente. Como o agente não interage com o ambiente, e o conjunto de transições fornecido é geralmente finito, não se pode esperar que o planejador consiga encontrar uma política ótima. O objetivo, portanto, muda de aprender uma política ótima para aprender a melhor política possível a partir dos dados fornecidos [Sascha Lange and Riedmiller, 2012].

4.2 Fundamentos dos Algoritmos de BRL

Métodos de aprendizado *on-line* independentes do modelo como o *Q-learning* são atraentes do ponto de vista conceitual e são bem sucedidos quando aplicados a problemas com espaço de estados pequeno e discreto, no entanto, ao aplicá-los em sistemas reais com o espaço de estados maior, e possivelmente contínuo, estes algoritmos se deparam com alguns fatores limitantes como por exemplo [Sascha Lange and Riedmiller, 2012]:

1. A sobrecarga do processo de exploração, tornando o aprendizado lento

2. Ineficiências devido à aproximação estocástica

3. Problemas de estabilidade ao utilizar a função de aproximação

Os algoritmos modernos de BRL abordam todas as três questões e encontram soluções específicas para cada um deles:

- A idéia de "Repetição das experiências" para resolver o problema de sobrecarga do processo de exploração.
- A idéia de "Ajuste" para tratar o problema de estabilidade. No aprendizado por reforço *on-line* a função valor é atualizada localmente para um estado particular depois de cada observação, deixando todos os outros estados intocados. Atualizações subsequentes podem usar este valor atualizado para sua própria atualização. **?**, entre outros, mostraram exemplos em que combinações particulares de *Q-Learning* possuem comportamento instável ou até mesmo conduz a certeza de divergência. A idéia é primeiro aplicar uma atualização de programação dinâmica para todos os membros de um conjunto dos chamados 'suportes' (pontos distribuídos por todo o espaço do estado), calcular novos valores para todos os suportes, e depois usar aprendizado supervisionado, substituindo assim as atualizações locais.
- Substituição de Aproximação Estocástica Ineficiente.

Uma característica do aprendizado por reforço em lote é que ele concede comportamento estável para o Q-learning, independente de uma modelagem especial ou função recompensa específica. Existem dois aspectos a discutir: a) estabilidade, no sentido de convergência garantida de uma solução, e b) a qualidade, no sentido da distância desta solução para a função de um valor óptimo verdadeiro.

4.3 Fitted Q Iteration

Proposto por Damien Ernst's, *Fitted Q Iteration* [Ernst et al., 2005] é possívelmente o algoritmo mais popular de Aprendizado por Reforço em Lote. Ele pode ser visto como o 'Q-Learning do BRL'. Dado um conjunto fixo $F = \{(s_t, a_t, r_{t+1}, s_{t+1}|t = 1, ..., p)\}$ de p transições (s, a, r, s') e um valor inicial $\overline{q}^0$ para Q (Ernst et al. [2005] usou $\overline{q}^0$ = 0), o algoritmo começa inicializando uma aproximação inicial $\hat{(}Q)^0$ da função Q $\hat{Q}^0$ com $\hat{(}Q)^0(s, a) = \overline{q}^0$ para todo $(s, a) \in SxA$. Então ele itera sobre os seguintes passos:

1. Inicia com o conjunto vazio p^{i+1} de padrões $(s, a, \overline{q}^{i+1}_{s,a})$. Então, para cada transição $(s, a, r, s') \in F$ calcula

$$\overline{q}^{i+1}_{s,a} = r + \gamma \max_{a' \in A} \hat{Q}^i(s', a') \tag{4.1}$$

   e adiciona $(s, a, \overline{q}^{i+1}_{s,a})$ a p^{i+1}.

2. Usa aprendizado supervisionado para extrair $\hat{Q}^{i+1}$, uma aproximação de Q^{i+1}

O Algoritmo 4 mostra como seria uma implementação do *Fitted Q Iteration*.

Algoritmo 3: BatchReinforcementLearning(m) adaptado de [Kalyanakrishnan and Stone, 2007]

```
Saída: Q
Q ← Q_0;
enquanto Q não convergiu faça
    D ← {}
    i ← 0
    experiencias ← 0
    enquanto experiencias < m faça
        i ← i + 1
        s_i ← estadoDoAmbiente()
        experiencias ← experiencias + 1
        a_i ← selecionaAcao(Q, s_i)
        executaAcao(a_i)
        r_i ← recompensaDoAmbiente()
        s_{i+1} ← estadoDoAmbiente()
        d_i ← (s_i, a_i, r_i, s_{i+1})
        D ← D ∪ {d_i}
    fim
    Q ← treinaLote(D)
fim
```

Algoritmo 4: fittedQIteration(D) adaptado de [Kalyanakrishnan and Stone, 2007]

Saída: Q

$i \leftarrow 0$;

enquanto $i < k$ **faça**

  para cada $t \in [1, ...|D|]$ **hacer**

    $T_i \leftarrow r_i + \gamma \max_a Q_{i-1}(s_{i+1}, a)$

  fin

  $Q_i \leftarrow Q_0$

  para cada *iteracao = 1 até k* **hacer**

    para cada $t \in [1, ...|D|]$ **hacer**

      $Q_i(s_t, a_t) \leftarrow Q_i(s_t, a_t) + \alpha_{supervisionado}(T_t - Q_i(s_t, a_t))$

    fin

  fin

fim

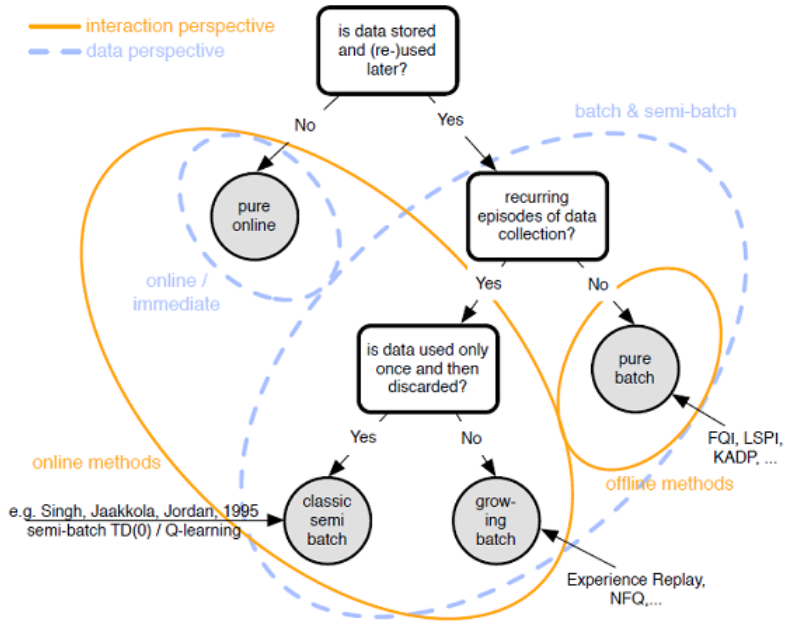

Figura 4.2: Classificação dos algoritmos de Aprendizado por reforço batch vs. não-batch de acordo com a perspectiva de uso dos dados e interação ([Sascha Lange and Riedmiller, 2012]).

Capítulo 5

Estudo de caso: O Problema do Vendedor de Impressoras

A oferta de soluções ou produtos sofisticados dependem de um processo de venda complexo em que uma série de ações sequenciais (que chamaremos de *ações de venda*) devem ser executadas pelos vendedores com o objetivo de influenciar o cliente à compra. Como o resultado das ações pode ser não-determinístico, a escolha da melhor sequência de ações de venda pode ser vista como um problema de planejamento.

5.1 O Processo de Venda

Um processo de venda pode variar de acordo com o vendedor, com o produto a ser comercializado e do cliente, e em geral um processo de venda passa por algumas etapas básicas descritas a seguir:

- **Prospecção**: O objetivo dessa etapa é buscar clientes em potencial (*prospects*), isto é, clientes que possuam algumas características que os apontem como possíveis consumidores do produto que está sendo comercializado.
- **Qualificação do cliente**: A etapa de prospecção tende a ser pouco criteriosa. Ter uma lista de clientes com características de potenciais compradores ainda é muito pouco. Por exemplo, se o produto a ser comercializado for *impressoras de outdoors*, é natural procurar na etapa de prospecção por empresas de *propaganda e marketing*. Porém nem toda empresa

de *propaganda e marketing* trabalha com esse tipo de mídia. A etapa de qualificação tem como objetivo obter mais informações do cliente e descobrir se ele tem necessidade de adquirir os produtos comercializados.

- **Definição das necessidades do cliente**: Nessa etapa o vendedor deve procurar entender com detalhes as necessidades do cliente para oferecer o produto mais adequado garantindo assim a satisfação do cliente após a venda.

- **Elaboração de propostas**: É uma das etapas mais críticas no processo de venda. Uma proposta precisa resolver a necessidade do cliente de forma satisfatória e precisa ter um custo compatível com sua situação. Nem sempre a necessidade do cliente será satisfeita pelo produto mais barato, e é comum ter soluções parecidas de produtos concorrentes a preços inferiores. A proposta precisa defender que aquilo que está sendo oferecido é o produto ideal e melhor que o produto concorrente.

- **Negociação**: Depois de convencer o cliente de que o produto oferecido é o que melhor atende as suas necessidades, o processo entra na etapa de negociações. Esta etapa fica muito centralizada na questão financeira. O vendedor nessa etapa terá que oferecer descontos ou condições favoráveis ao comprador, mas também precisa garantir a rentabilidade de seus produtos.

- **Fechamento do negócio**: Essa é a etapa em que o cliente decide pela compra do produto e é consequência das ações tomadas nas etapas anteriores. Embora pareça uma etapa simples onde a venda está totalmente concluída, na verdade há muitas razões para o cliente desistir ou adiar a compra nessa etapa. O cliente pode desconfiar de que as condições apresentadas não são as melhores possíveis, pode desconfiar das afirmações do vendedor, pode analisar a possibilidade de encontrar condições melhores nos concorrentes ou simplesmente pode ter entraves financeiros que inviabilizam a compra. É normal nessa etapa retroceder para outras etapas do processo ou até mesmo perder definitivamente a venda.

- **Pós-venda**: Nessa etapa a venda foi concluída com sucesso, porém é importante que o vendedor mantenha contato com o cliente para verificar se ele está satisfeito e se surgiram novas necessidades.

Figura 5.1: Ciclo de Vendas: etapas básicas de um processo de venda.

A Figura 5.1 mostra as dependências entre as etapas básicas de um processo de venda. É importante notar que em qualquer uma dessas etapas o processo pode ser interrompido com a desistência de compra do cliente. Cada uma dessas etapas envolve uma ou mais *ações de venda*. As caixas escuras indicam estágios do processo de venda em que o vendedor deve tomar decisões críticas que serão tratadas nesse trabalho.

5.2 Impressoras Plotters

Plotters de Impressão são impressoras destinadas a impressões de grandes dimensões e são muito usadas em empresas de arquitetura para imprimir plantas e projetos de engenharia. Também são muito usadas pelo mercado de comunicação visual na impressão de pôsters, banners e painéis, e na indústria têxtil para impressão em tecidos.

Existem modelos específicos de impressoras plotters para cada necessidade. De modo geral os modelos se distinguem pelo tipo de aplicação (impressão Digital, geoprocessamento, ...), tipo de mídia usada na impressão (lona, tecido, papel fotográfico, ...), largura de impressão, tecnologia de impressão (tinta a base de água, tinta a base de látex, ...) e quantidades de cores.

Cada tipo de impressão exige características específicas da impressora plotter. Por exemplo, um engenheiro ao imprimir uma planta não precisa de fidelidade nas cores, mas precisa de precisão nos traços. Impressão de fotografia já exige fidelidade nas cores. Impressão de adesivos para carros exigem mídias específicas e tinta que não dissolve em água. Cada característica influencia diretamente o custo do equipamento, o tamanho, as condições necessárias para instalação e os conhecimentos técnicos exigidos para o manuseio.

De uma forma geral, podemos dividir os tipos de impressoras plotters em 3 grandes grupos de acordo com o mercado que ela atende:

- **TÉCNICO**: plotters para arquitetos e engenheiros,
- **CRIATIVO**: plotters para fotógrafos e designers,
- **PSP**: plotters para prestadores de serviço de impressão,

Com tudo isso é possível deduzir que o processo de venda de uma impressora plotter é bastante complexo, possui um ciclo de vendas característico e tempo de negociação estendido.

5.3 Sistema de Acompanhamento de Negociações (Forecast)

O Big Impression é um programa especializado no mercado de impressão em grandes formatos, que atende o público SMB (pequenas e médias empresas) e profissionais liberais. Surgiu em 2003 após a percepção da necessidade de mercado de ter especialistas no segmento de impressão em Grandes Formatos. Hoje, o Programa Big Impression é um canal mais especializado, representando o maior número de vendas de impressoras Plotters HP no território brasileiro através de revendedores espalhados por todo território nacional.

Com o objetivo de controlar as negociações e profissionalizar o relacionamento comercial entre as revendas parceiras e os clientes, foi criado um sistema de controle de negócios denominado *Forecast*. O Forecast se tornou uma importante ferramenta de apoio ao processo de vendas de impressoras plotters pois permitiu ter controle e previsão da vendas, porém ela não dá suporte à tomada de desisão do vendedor.

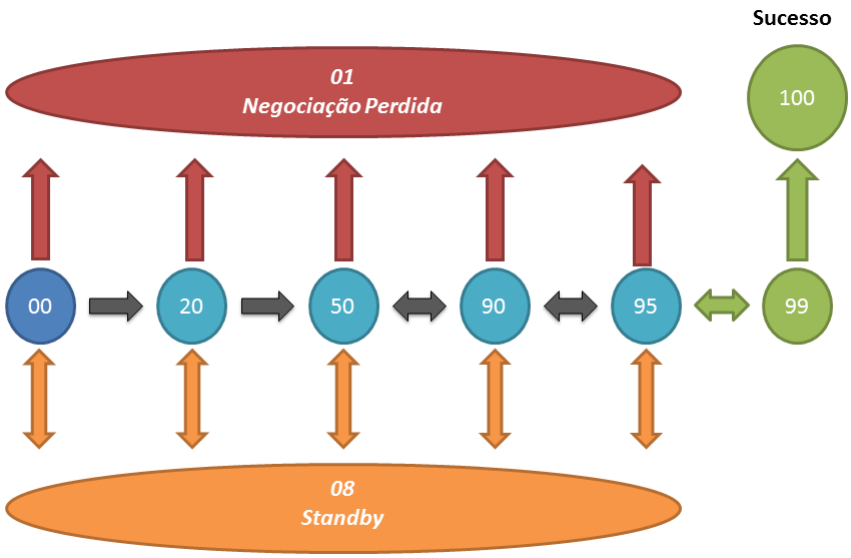

Figura 5.2: Ciclo de Vendas do Programa Big Impression. Na etapa 08 (Standby) o vendedor aguarda por um tempo antes de retomar o processo de negociação.

5.3.1 Estágios do processo de negociação

O processo do sistema é feito em etapas denominadas *estágios de negociação*. Em cada estágio a revenda deve executar algumas ações e reportar ao sistema o resultado da execução das ações avançando, retrocedendo, adiando ou terminando o processo de negociação. A aplicabilidade das ações no sistema em geral dependem de pré-requisitos e o resultado das ações em geral dependem de fatores externos ao sistema e não são determinísticos. A Figura 5.2 mostra o ciclo de vendas implementado pelo sistema Forecast.

Estágio 00 - Primeiro contato / Pré Qualificação

O estágio *00*, como o próprio nome sugere, é o estágio inicial do processo de venda. Neste estágio a revenda precisa realizar o contato inicial com o cliente para conhecer suas necessidades e definir quais produtos serão oferecidos ao cliente. Convidar o cliente para uma demonstração do produto pode ser importante se o cliente não conhece o produto oferecido ou tiver dúvidas se o produto atenderá suas necessidades.

Estágio 20 - Conhecer melhor o Cliente

Neste estágio espera-se que o vendedor mantenha contato e conheça melhor o cliente, suas necessidades mais detalhadamente, suas instalações e quem serão os responsáveis por aprovar a compra do produto (por exemplo o diretor financeiro ou o dono da empresa).

Estágio 50 - Proposta Comercial

Após conhecer o cliente, suas necessidades e suas condições, chega o momento de elaborar uma *proposta comercial*. É natural que o cliente esteja negociando com outros fornecedores e por isso é essencial tentar conhecer quais são os concorrentes no processo, quais produtos estão sendo oferecidos e com quais condições. Fazer demonstrações para mostrar a superioridade do produto oferecido em relação ao concorrente pode ser necessário caso o produto oferecido tenha um preço superior.

Oferecer descontos, condições de pagamento e serviços agregados podem ser necessários nessa fase se por algum motivo o cliente perder o interesse pela compra por causa de preço. Para oferecer descontos o vendedor precisa solicitar autorização via o sistema Forecast, justificar o pedido de desconto e aguardar o deferimento da solicitação realizado pelo gerente do programa

Estágio 90 - Aceite da proposta comercial

O objetivo deste estágio é aguardar o aceite do cliente em relação a proposta elaborada no estágio anterior. É comum neste estágio o cliente recusar a proposta oferecida e o vendedor deve decidir entre voltar para o estágio anterior para elaborar uma nova proposta ou desistir da venda.

Estágio 95 - Pedido

Neste estágio o cliente optou pela compra. Se necessário o vendedor pode oferecer ao cliente o financiamento do produto. Neste caso cabe ao vendedor aguardar a aprovação do financiamento. Como algumas plotters possuem algumas pré-condições de instalação, também é importante que o vendedor cheque com o cliente se as pré-condições são atendidas. Por fim cabe ao vendedor realizar o processo de inserção do pedido.

Estágio 99 - Entrega

Neste estágio cabe ao vendedor apenas acompanhar o processo de faturamento de entrega. Não há nenhuma ação para ser realizada.

Estágio 100 - Negociação concluida com sucesso

Chegar neste estágio significa que a venda foi realizada com sucesso. Para o sistema o processo acaba por aqui, mas é muito importante que o vendedor mantenha contato constante com o cliente para detectar novas oportunidades.

Estágio 01 - Negócio perdido

Existem muitas razões para que o negócio seja perdido, por exemplo: o cliente pode não ser um cliente potencial para os produtos oferecidos, o cliente pode não ter condições para efetuar a compra ou ainda o negócio pode ter sido perdido para um concorrente. Para todos estes casos o vendedor precisa finalizar o processo.

Estágio 08 - Standby

Em alguns casos o vendedor detecta que o cliente não tem potencial para compra no momento mas terá em algum momento futuro. As vezes o próprio cliente sinaliza a possibilidade de compra futura. Para estes casos o vendedor pode deixar a negociação inativa por um período de tempo e reativá-la com todo o histórico no futuro, ou ainda ser avisado automaticamente depois de um termpo pré-determinado.

5.4 Sistema de Tomada de Decisão

Um problema enfrentado pelo programa Big Impression é a rotatividade de revendas e vendedores. Frequentemente revendas são credenciadas ou descredenciadas do programa. Dentro das revendas é ainda mais comum a rotatividades de vendedores. Quando ocorre o desligamento de uma boa revenda ou um bom vendedor, o conhecimento e a experiência são perdidos.

Apesar do *programa Big Impression* responder por mais de 50% das plotters HP vendidas no brasil, menos de 10% das negociações terminam em vendas. Se considerarmos que os clientes são legitimados no sistema Forecast após a etapa de *qualificação*, espera-se que o volume de vendas possa aumentar se ações corretas forem aplicadas para cada cliente. Dado o grande número de revendas e vendedores que utilizam o sistema Forecast, uma maneira de oferecer suporte à escolha correta de ações de venda é através de um sistema automático que aprende com os vendedores melhores e mais experientes, e compartilha este conhecimento entre os vendedores menos experientes.

5.4.1 Identificando as Variáveis de Estado

Para resolver um problema complexo como o de venda de plotters não é possível partir do pressuposto de que toda negociação é igual. Pelo contrário, há muitos fatores que podem interferir no processo. Alguns são óbvios, como por exemplo a situação fianceira do cliente. Porém, há muitos fatores desconhecidos que influenciam esse processo e que precisam de alguma forma ser descobertos e mapeados no modelo.

O sistema Forecast possui entre 20 e 30 variáveis mapeadas no processo de venda. Portanto é preciso identificar quais destas variáveis são relevantes para a tomada de desisão. Numa análise prévia da base de dados do sistema Forecast com o histórico de negociações, descobrimos algumas das variáveis que certamente influenciam no processo, e que serão utilizadas na descrição de estados (i.e., definem variáveis de estado). Entre elas:

- **Segmento**: como explicado anteriormente, as plotters HP são divididas em 3 grupos chamados de *segmentos* (técnico, criativo e PSP). A Figura **??** mostra como altera a probabilidade média de transição entre estados de acordo com o segmento do produto. Uma possível explicação é a diferença de preços, procura entre os segmentos, O segmento PSP possui as máquinas maiores e mais caras e o segmento CRIATIVO possui as máquinas menos procuradas.

- **Região**: na análise prévia da base de históricos percebemos que o processo varia muito dependendo do estado onde mora o cliente.

Capítulo 6

Modelando o Problema do Vendedor de Impressoras como um MDP

A seguir uma modelagem formal do *problema do vendedor de impressoras* como um MDP. A Tabela 6.1 lista as ações de venda consideradas no modelo. A Tabela 6.2 lista as variáveis consideradas relevantes para o processo de decisão e que foram utilizadas no modelo.

Ações de Venda	Descrição
oferecer.produto.criativo	Ação de oferecer ao cliente um produto do segmento *Criativo*.
oferecer.produto.tecnico	Ação de oferecer ao cliente um produto do segmento *Técnico*.
oferecer.produto.PSP	Ação de oferecer ao cliente um produto do segmento *PSP*.
agendar.demonstracao	Ação de convidar o cliente para uma demonstração do produto.
estagio.20.conhecer.cliente	Ação de alterar o processo para o estágio 20 em que o objetivo do vendedor é conhecer melhor o cliente.
estagio.50.elaborar.proposta	Ação de alterar o processo para o estágio 50 em que o objetivo do vendedor é elaborar uma proposta comercial para o cliente.
proposta.sem.descontro	Ação de oferecer ao cliente uma proposta sem qualquer desconto ou benefício.
oferecer.desconto	Ação de montar uma proposta comercial com desconto para apresentar ao cliente. O desconto pode ser reprovado pelo gerente do projeto.
oferecer.financiamento	Ação de oferecer opções de financiamentos para o cliente. Necessária apenas em casos em que o cliente não tem condições de pagar o produto a vista. O financiamento pode ser reprovado pela instituição financeira.
estagio.90.aceite.da.proposta	Ação de alterar o processo para o estágio 90 em que a proposta pode ser aceita ou recusada pelo cliente.
estagio.95.pedido	Ação de alterar o processo para o estágio 95 em que o vendedor pode criar o pedido.
estagio.99.entrega	Ação de alterar o processo para o estágio 99. Ao chegar nesse estágio considera-se que a venda foi efetuada com sucesso.
standby	Ação de inativar a negociação por um período de tempo.
cliente.nao.potencial	Ação de desistência do cliente ou do vendedor.

Tabela 6.1: Lista de ações consideradas no modelo do problema do vendedor de impressoras.

```
Dominio vendedor_impressoras {

    Tipos{
        enum_estagio: {@00, @01, @08, @20, @50, @90, @95, @99}
        enum_segmento: {@TECNICO, @CRIATIVO, @PSP, @NULL}
        enum_desconto: {@0, @10, @20, @30}
    };

    variaveis{
        CUSTO_DEMONSTRACAO : {real, default = -10 };
        CUSTO_LIGACAO : { real, default = -2 };
        CUSTO_DESCONTO_10 : { real, default = -10 };
        CUSTO_DESCONTO_20 : { real, default = -20 };
        CUSTO_DESCONTO_30 : { real, default = -30 };
        CUSTO_PEDIDO_FINANCIAMENTO : { real, default = -5 };
```

Variáveis de estado	Descrição
CUSTO_DEMONSTRACAO	Custo que será aplicado a ação de agendar uma demonstração
CUSTO_DESCONTO	Custo associado a ação de oferecer um desconto%
CUSTO_PEDIDO_FINANCIAMENTO	O pedido de financiamento inicia um processo de análise de crédito que possui um custo. Esta variável representa este custo.
CUSTO_FINANCIAMENTO	Custo caso o financiamento seja aprovado.
RECOMPENSA_VENDA	Recompensa pela venda do produto.
estagio	Variável que representa o estágio em que a negociação se encontra.
estagio_anterior	Uma negociação em standby (estágio 08) ao ser ativada volta automaticamente para o estágio de onde saiu.
segmento	Segmento do produto oferecido ao cliente (técnico, criativo ou PSP)
pj	Variável booleana. Indica se o cliente é uma empresa (1) ou pessoa física (0).
regiao	Região onde o cliente está localizado (NORTE, NORDESTE, CENTRO-OESTE, SUDESTE, SUL)
demonstracao	Variável booleana. Indica se o cliente foi chamado para uma demonstração
standby	Variável booleana. Indica se a negociação ficou em *standby*.
proposta	Variável booleana. Indica se foi apresentada uma proposta ao cliente
desconto	Variável booleana. Indica se foi oferecido desconto ao cliente.
desconto_aprovado	Variável booleana. Indica se o desconto foi aprovado pelo gerente do programa.
aceitou_proposta	Variável booleana. Indica se o cliente aceitou alguma proposta apresentada.
financiamento	Variável booleana. Indica se foi solicitado financiamento do produto.
financiamento_aprovado	Variável booleana. Indica se o financiamento foi aprovado.

Tabela 6.2: Lista de variáveis de estado consideradas relevantes e utilizadas no modelo do problema do vendedor de impressoras.

```
        CUSTO_FINANCIAMENTO : { real, default = -30 };
        RECOMPENSA_VENDA : { real, default = 100 };

        estagio: { enum_estagio, default = @00};
        estagio_anterior: { enum_estagio, default = @00};
        segmento: { enum_segmento, default = @NULL};
        agendamentos : { int, default = 0 };
        tempo_standby : { int, default = 0 };
        tempo_negociacao : { int, default = 0 };
        tempo_alteracao : { int, default = 0 };
        //ligações, emails, etc
        contatos : { int, default = 0 };
        // número de propostas
        propostas : { int, default = 0 };
        // se foi oferecido desconto
        desconto: { enum_desconto, default = @0};
        desconto_aprovado: { bool, default = true};
        aceitou_proposta : { bool, default = false };
        // se foi oferecido financiamento
        financiamento : { bool, default = false };
        financiamento_aprovado : { bool, default = true };
    };
    acoes{
        exclui_produtos:{
            pre: (estagio = @00 or estagio = @50)
            ef: segmento = @NULL
            r: 0
        }
        escolhe_produto(enum_segmento s){
            pre: (segmento = @NULL)
            ef: segmento = s
            r: 0
        }
```

```
agenda_demonstracao{
    pre: (segmento not @NULL)
    ef: agendamentos++
    r: CUSTO_DEMONSTRACAO
}
muda_estagio_p_20{
    pre: (estagio = @00 and segmento not @NULL)
    ef: {
            estagio = @20
            tempo_alteracao = 0
        }
    r: 0
}
liga_p_cliente{
    pre: (estagio not @01 and estagio not @100)
    ef: ligacoes++
    r: CUSTO_LIGACAO
}
muda_estagio_50{
    pre: ((estagio = @20 and ligacoes > 0) or estagio = @90)
    ef: {
            estagio = @50
            tempo_alteracao = 0
        }
    r: 0
}
faz_proposta {
    pre: (estagio = @50)
    ef: proposta++
    r: 0
}
oferece_desconto(enum_desconto d){
    pre: (estagio = @50)
```

```
        ef: {
            proposta++
            desconto = d
            desconto_aprovado = false
        }
        r: 0
    }
    consulta_desconto{
        pre: (desconto not @0)
        ef: {
            p = ?p0: desconto_aprovado = true
            p = 1 - ?p0: desconto_aprovado = false
        )
    }
    muda_estagio_90{
        pre: ((estagio = @50 and proposta > 0 and desconto_aprovado = true)
                or estagio = @95)
        ef: {
                estagio = @90
                tempo_alteracao = 0
            }
        r: 0
    }
    consulta_cliente{
        pre: (proposta > 0 and estagio = @90)
        ef: {
            p = ?p1: aceitou_proposta = true
            p = 1 - ?p1: aceitou_proposta = false
        }
    }
    muda_estagio_95{
        pre: (estagio = 90 and aceitou_proposta = true)
        ef: {
```

```
            estagio = @95
            tempo_alteracao = 0
        }
    r: {
        (desconto = @0) 0
        (desconto = @10) CUSTO_DESCONTO_10
        (desconto = @20) CUSTO_DESCONTO_20
        (desconto = @30) CUSTO_DESCONTO_30
       }
}
oferece_financiamento{
    pre: (estagio = @95)
    ef: {
            financiamento = true
            financiamento_aprovado = false
        }
    r: CUSTO_PEDIDO_FINANCIAMENTO
}
consulta_financiamento{
    pre: (financiamento = true and estagio = 95)
    ef: {
        p = ?p2: financiamento_aprovado = true
        p = 1 - ?p2: financiamento_aprovado = false
    }
)
}
muda_estagio_99{
    pre: (estagio = @95 and financiamento_aprovado = true)
    ef: {
            estagio = @99
            tempo_alteracao = 0
        }
    r: {
```

```
                (financiamento = true) CUSTO_FINANCIAMENTO
                (financiamento = false) 0
            }
        }
        consulta_entrega{
            pre: (estagio = @99)
            ef: {
                p = ?p4: {
                        estagio = @100
                        r = RECOMPENSA_VENDA
                     }
                p = 1 - ?p4: estagio = @01
            )
        }
        muda_estagio_standby(int t){
            pre: (estagio not @100 and estagio not @01)
            ef: {
                    estagio = @08
                    tempo_standby = t
                    estagio_anterior = estagio
                    tempo_alteracao = 0
                }
            r: 0
        }
    }
    eventos_exogenos{
        ((tempo_standby > 0)
            p = 1: tempo_standby--
        )
        ((tempo_standby = 0 and estagio = @08)
            p = 1: estagio = estagio_anterior
        )
        ((estagio not @01 and estagio not @100)
```

```
            p = 1: tempo_negociacao++;
            p = 1: tempo_alteracao++;
        )
        ((estagio not @01 and estagio not @100 and esatgio not @99 and tempo_altera
            p = 1: estagio = @01
        )
    }
}
```

Capítulo 7

Resultados Experimentais

7.1 Avaliação da política aprendida

No *Problema do Vendedor de Impressoras*, o objetivo é encontrar uma política ótima π^*, ou seja, uma função que mapeia a melhor ação de venda a ser executada em cada estado de forma a maximizar a esperança das recompensas acumuladas ao longo de um processo de venda, considerando que o resultado da execução das ações é estocástico conforme modelado no Capítulo 6.

Na Seção 7.1.1 discutimos uma maneira de validação das técnicas usadas para resolver o *Problema do Vendedor de Impressoras* através da adaptação da técnica de *Validação Cruzada* conhecida como *K-Fold* apresentada no Apêndice A.4.

7.1.1 Teste de convergência

Considere:

- $V(s)$ a esperança da recompensa acumulada que pode ser obtida a partir do estado s,
- $q(s, a), s \in S$ e $a \in A$ a esperança de recompensa acumulada que pode ser obtida a partir do estado s executando a ação a,
- $\hat{V}(s)$ o valor aprendido,
- $\hat{q}(s, a)$ o valor aprendido,

- $T = \{t_1, t_2, ..., t_n\}$ um conjunto de teste com n tulplas onde $t_i = \{s_i, a_i, s'_i, r_i\}$, $s_i \in S$, $a_i \in A$, $s'_i \in S$, $r_i \in R$ indica em que a ação a_i foi executada no estado s_i, gerando uma recompensa r_i e indo para o estado s'_i.

Considere as equações:

$$E(i) = \hat{q}(s_i, a_i) - (r + \alpha \hat{V}(s'_i)) \tag{7.1}$$

$$E(s, a) = \sum E(k), \text{em que } t_k = \{s, a, s'_i, r_i\} \tag{7.2}$$

A Equação 7.6 é uma estimativa do quanto $\hat{q}(s, a)$ convergiu em relação ao conjunto de testes T. É possível notar que se $\hat{q}(s, a) = q(s, a)$, então para uma amostra de testes T não viesada e suficientemente grande $E(s, a)$ converge para 0.

Sendo assim a equação 7.3 é uma boa estimativa do quanto a política aprendida convergiu para a política ótima.

$$E(T) = \frac{\sum_{s \in S, a \in A} N(s, a) E(s, a)}{n}, \text{onde } N(s, a) = \text{número de tuplas } \{s, a, s'_i, r_i\} \tag{7.3}$$

Com isso, uma forma de validar os algoritmos usados para resolver o *Problema do Vendedor de Impressoras* é adaptando a técnica *K-Fold* de *Validação Cruzada* do aprendizado supervisionado, apresentada no Apêndice A.4, usando a Equação 7.3 para validar o aprendizado em cada um dos K subconjuntos de teste.

7.1.2 Aprendizado por reforço em lote vesus Iteração de Valor

Conforme abordado no capítulo anterior, foram usadas duas técnicas para resolver o problema apresentado. *Iteração de Valor* que aprende as probabilidades de transição e então calcula a política ótima, e *Aprendizado por Reforço em Lote* que aprende diretamente a política ótima.

As Figuras 7.1 e 7.2 mostram o resultado da validação dos dois algoritmos usando a técnica *K-Fold* para k igual a cinco e para k igual a 10 respectivamente. Como é possível notar, nos dois testes o algoritmo de *Iteração de Valor* se mostrou mais acurado.

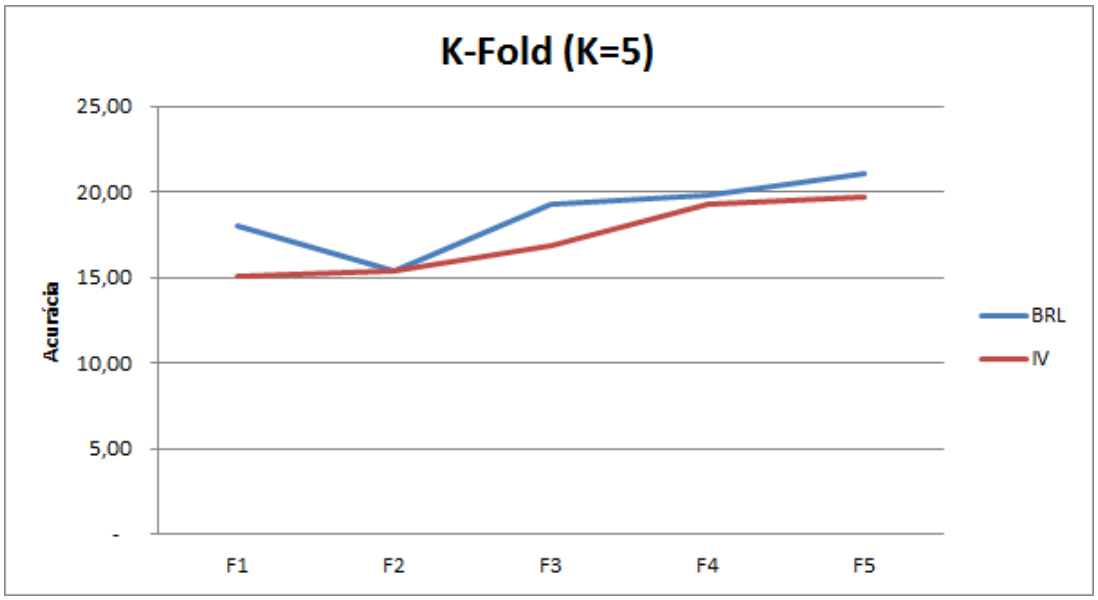

Figura 7.1: Avaliação dos algoritmos de Aprendizado BRL vs. IV usando Validação Cruzada adaptada com k=5

7.1.3 Política dos vendedores versus política aprendida

Uma segunda análise feita foi a comparação da política dos vendedores com a política dada pelos algoritmos.

Analisando cada transição separadamente:

- O Iteração de Valor sugere uma ação diferente da ação escolhida pelo vendedor em 45,73% dos casos;
- O Batch Reinforcement Learning sugere uma ação diferente da ação escolhida pelo vendedor em 47,28% dos casos;
- O Iteração de Valor sugere uma ação diferente da ação escolhida pelos melhores vendedores em 37,56% dos casos;

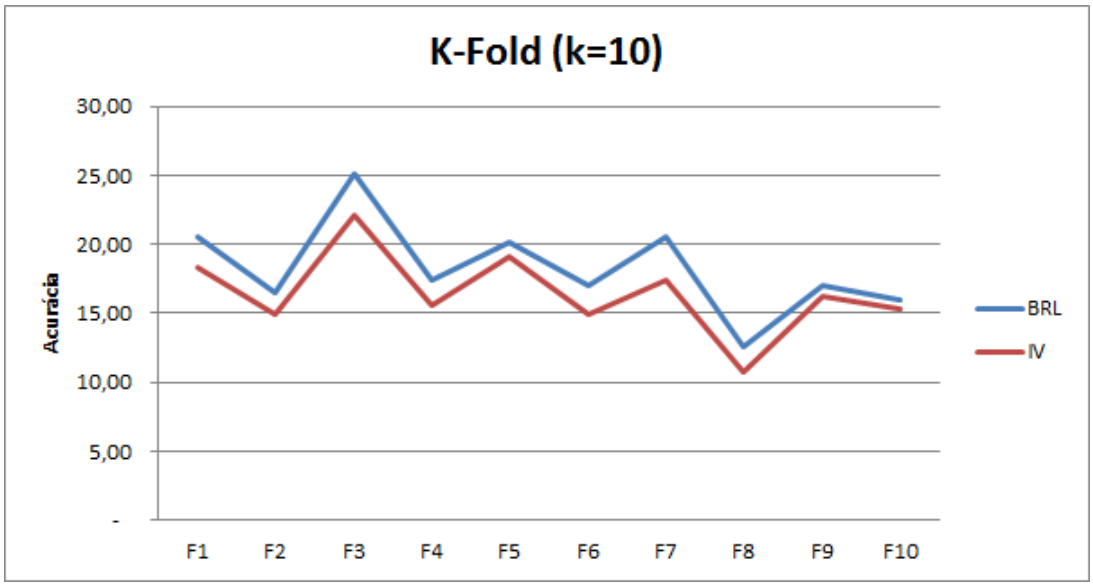

Figura 7.2: Avaliação dos algoritmos de Aprendizado BRL vs. IV usando Validação Cruzada adaptada com k=10

- O Batch Reinforcement Learning sugere uma ação diferente da ação escolhida pelos melhores vendedores em 38,17% dos casos;
- O Iteração de Valor sugere uma ação diferente da ação escolhida pelo vendedor em 26,25% dos casos em que a negociação terminou em sucesso;
- O Batch Reinforcement Learning sugere uma ação diferente da ação escolhida pelo vendedor em 28,57% dos casos em que a negociação terminou em sucesso;
- O Iteração de Valor sugere uma ação diferente da ação escolhida pelos melhores vendedores em 25,08% dos casos em que a negociação terminou em sucesso;
- O Batch Reinforcement Learning sugere uma ação diferente da ação escolhida pelos melhores vendedores em 28,01% dos casos em que a negociação terminou em sucesso.

Analisando o processo inteiro (É feito o cálculo da porcentagem de ações dentro de uma determinada negociação igual a ação dada pelo algoritmo. Então é feito a média aritmética entre todas as negociações) :

- O Iteração de Valor sugere uma ação diferente da ação escolhida pelo vendedor em 45,32%

dos casos;

- O Iteração de Valor sugere uma ação diferente da ação escolhida pelo vendedor em 39,19% dos casos em que a negociação foi feita pelos melhores vendedores;
- O Iteração de Valor sugere uma ação diferente da ação escolhida pelo vendedor em 24,72% dos casos em que a negociação terminou em sucesso;
- O Iteração de Valor sugere uma ação diferente da ação escolhida pelo vendedor em 23,64% dos casos em que a negociação foi feita pelos melhores vendedores e terminou em sucesso;
- O Batch Reinforcement Learning sugere uma ação diferente da ação escolhida pelo vendedor em 46,58% dos casos;
- O Batch Reinforcement Learning sugere uma ação diferente da ação escolhida pelo vendedor em 38,52% dos casos em que a negociação foi feita pelos melhores vendedores;
- O Batch Reinforcement Learning sugere uma ação diferente da ação escolhida pelo vendedor em 27,19% dos casos em que a negociação terminou em sucesso;
- O Batch Reinforcement Learning sugere uma ação diferente da ação escolhida pelo vendedor em 26,83% dos casos em que a negociação foi feita pelos melhores vendedores e terminou em sucesso.

Se retirarmos os casos claros de desistência do cliente, ou seja, negociações em que o cliente ficou 12 meses sem se decidir pela compra ou não:

- O Iteração de Valor sugere uma ação diferente da ação escolhida pelo vendedor em 44,64% dos casos;
- O Iteração de Valor sugere uma ação diferente da ação escolhida pelo vendedor em 38,55% dos casos em que a negociação foi feita pelos melhores vendedores;
- O Batch Reinforcement Learning sugere uma ação diferente da ação escolhida pelo vendedor em 46,04% dos casos;
- O Batch Reinforcement Learning sugere uma ação diferente da ação escolhida pelo vendedor em 37,95% dos casos em que a negociação foi feita pelos melhores vendedores.

Com isso é possível perceber que a política adotada pelos melhores vendedores é mais próxima da política devolvida pelos algoritmos e que a política adotada pelos vendedores nos casos de sucesso é ainda mais próxima da política dos algoritmos. Isso é mais um indicativo de que os algoritmos aprendem com os casos de sucesso e fracasso dos vendedores e encontram uma política melhor.

Considerando que a política aprendida é a ótima, esta análise também pode ser usada para indicar quais vendedores precisam de treinamento.

7.2 Adaptação às variações do ambiente

Até o momento estamos considerando que o mercado é estático, porém isso não é verdadeiro. O mercado é dinâmico e é provável que os vendedores tentam se adaptar a essas variações do mercado. As Figuras 7.3, 7.4, 7.5 e 7.6 mostra o comparativo entre a política aprendida pelos algoritmos com a política adotada pelos vendedores ao longo dos anos.

Como é possível notar, embora o *Iteração de Valor* seja melhor, a técnica de *Aprendizado por Reforço em Lote* se adapta melhor as variações da política dos vendedores que muito provavelmente se deve a variações do mercado. Note que os melhores vendedores adotaram uma política muito mais próxima do *Aprendizado por Reforço em Lote* do que o *Iteração de Valor* em 2012 e 2013.

Um motivo para isso é que o Iteração de Valor aprende as probabilidades médias do período se adaptando à média do mercado no período, enquanto o *Aprendizado por Reforço em Lote*, que aprende em ordem cronológica dos acontecimentos, consegue se adaptar a essa variação.

7.3 Horizonte infinito versus Horizonte finito

Para as análises anteriores o *Problema do Vendedor de Impressoras* foi modelado, e resolvido, como um *MDP* de horizonte infinito. Conforme abordado no capítulo , num *MDP* de horizonte infinito o agente não possui limite de passos para interagir com o ambiente. Uma propriedade dessa modelagem é que o valor do estado $V(s)$ independe do número de passos que o agente gastou para chegar em s.

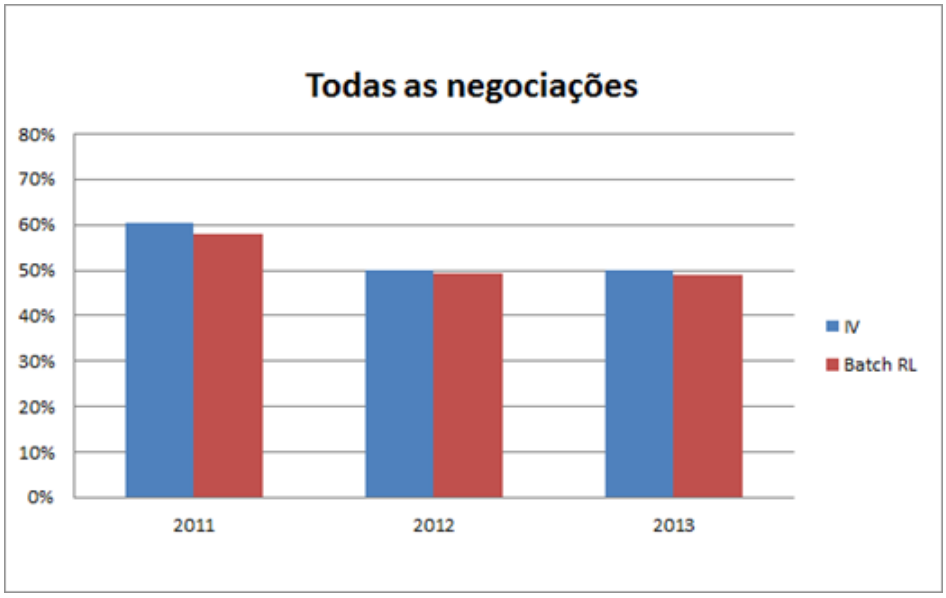

Figura 7.3: Adaptação à política dos vendedores ao longo dos anos.

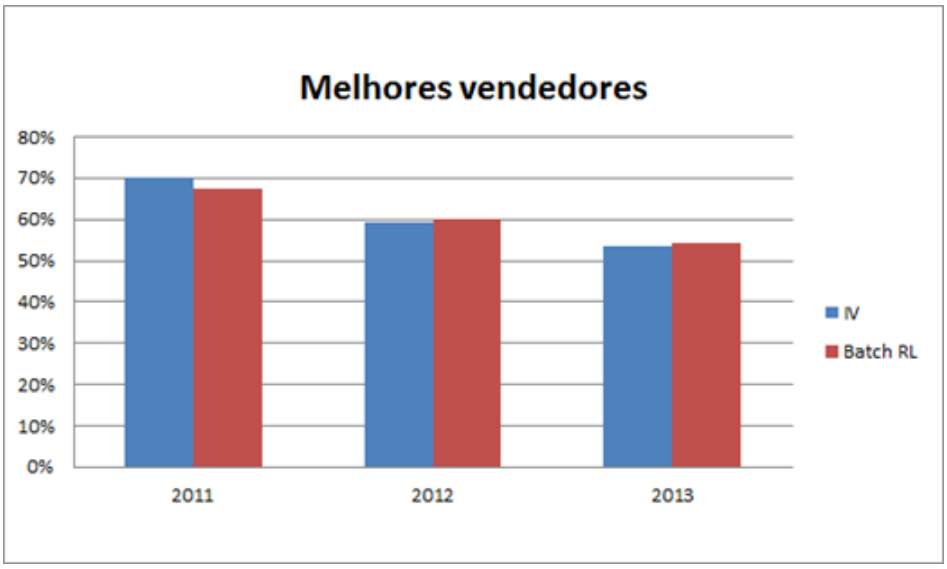

Figura 7.4: Adaptação à política dos vendedores ao longo dos anos.

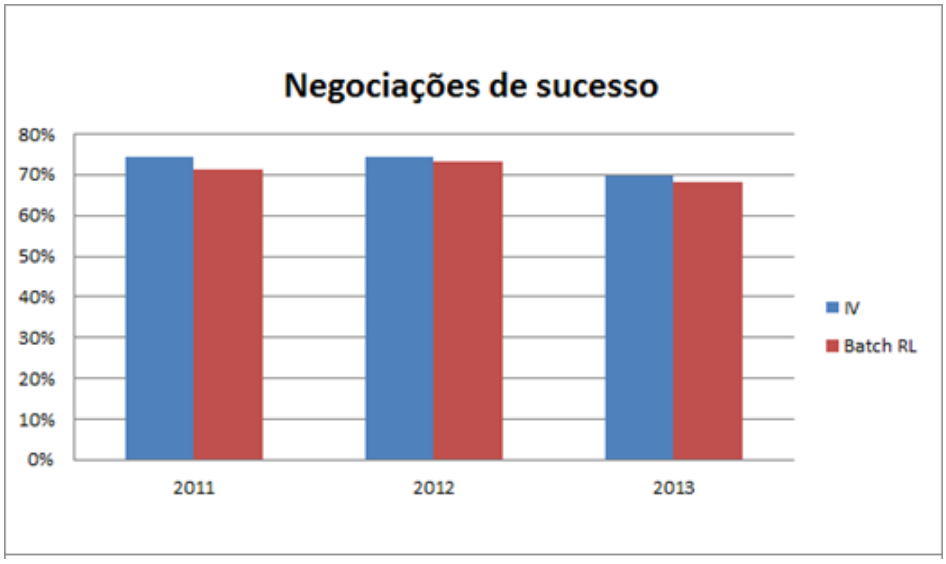

Figura 7.5: Adaptação à política dos vendedores ao longo dos anos.

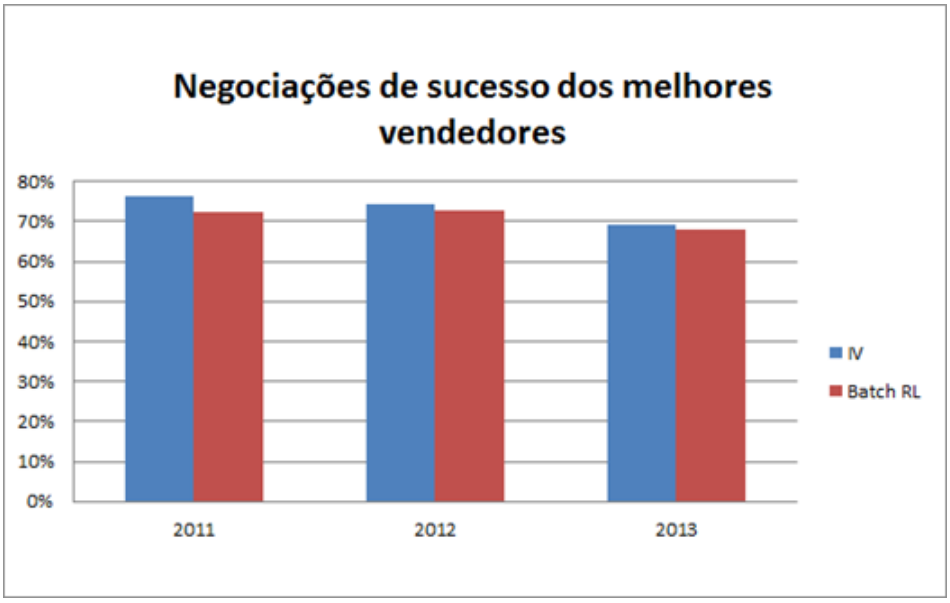

Figura 7.6: Adaptação à política dos vendedores ao longo dos anos.

Num processo de venda não é comum o vendedor manter a negociação com o mesmo cliente indefinidamente. Usando como exemplo o problema abordado, não é razoável sugerir ao vendedor que continue insistindo numa mesma negociação caso ele já tenha executado inúmeras ações e esta permanece estagnada no mesmo estado. Portanto, modelar o problema como um MDP de horizonte finito é mais fiel ao problema real. O problema da modelagem do problema como MDP de horizonte finito está no aumento do número de estados que passa agora a incorporar o tempo t em que o estado foi alcançado. Sendo assim a modelagem como MDP de horizonte finito só é viável computacionalmente se o horizonte for curto.

A Figura 7.7 compara a validação dos algoritmos que resolvem o problema como MDP de horizonte infinito com a validação dos algoritmos que modelam o problema como MDP de horizonte finito. Para isso foram feitas adaptações nas equações usadas na validação.

$$t_i = \{s_i, a_i, s'_i, r_i, h_i\} \tag{7.4}$$

$$E(i) = \hat{q}(s_i, a_i, h) - (r + \hat{V}(s'_i, h_i + 1)) \tag{7.5}$$

$$E(s, a, h) = \sum E(k), \text{em que } t_k = \{s, a, s'_i, r_i, h\} \tag{7.6}$$

Como é possível notar, o algoritmo de *Iteração de Valor* modelando o problema como um MDP de horizonte $h = 10$ se mostrou muito mais acurado. Já o BRL modelando o problema como um MDP de horizonte finito não se mostrou muito bom. Uma explicação para isso vem das propriedades de convergência do algoritmo abordadas na sessão 7.4. Com o estado expandido considerando o tempo t em que o estado foi alcançado, o número de amostras de cada par (estado, ação) diminui, diminuindo portanto a convergência do algoritmo para a política ótima.

7.4 Propriedades de convergência

Uma das propriedades do algoritmo que estima as probabilidades e executa o algoritmo de Iteração de Valor é que ele não depende do tamanho da amostra para convergir. Ele depende apenas do quão significativa é a amostra. No entanto, se a função de transição do ambiente não

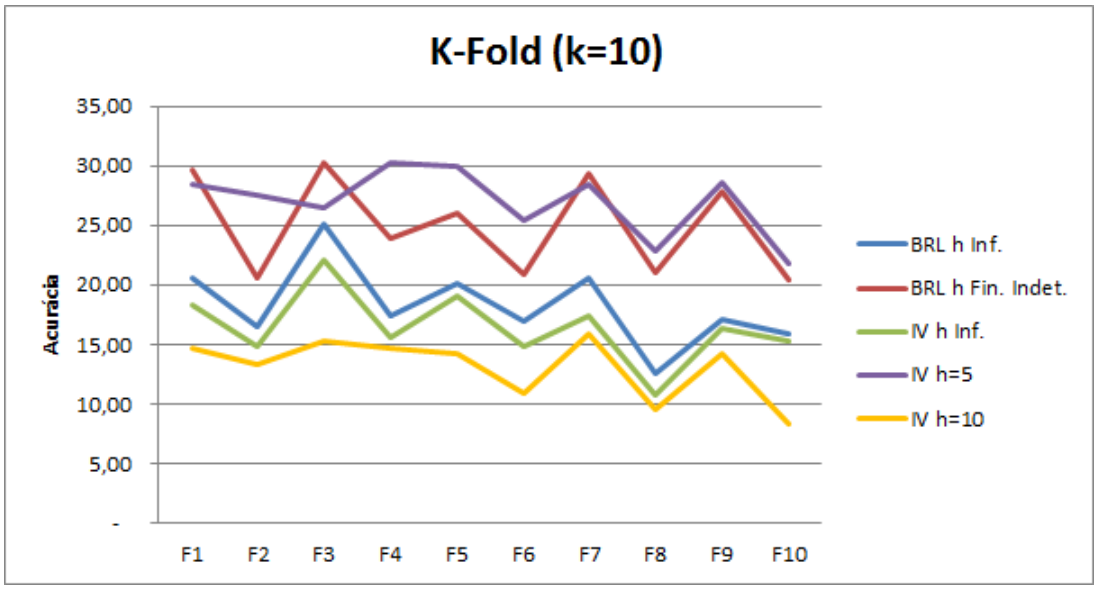

Figura 7.7: Avaliação dos algoritmos de Aprendizado BRL vs. IV modelando o problema como um MDP de horizonte finito ou infinito.

é estacionária, o algoritmo de iteração de valor pode convergir para uma política muito diferente da política ótima mais recente. Enquanto isso, o BRL aprendendo com as negociações na ordem cronológica em que elas ocorreram, converge para a política ótima mais recente. A Figura 7.8 mostra uma análise de convergência dos dois algoritmos aprendendo com as negociações dos últimos dois anos e meio.

Como é possível notar, para o problema abordado e período considerado, o IV com horizonte finito, $h = 10$, consegue manter um índice de convergência próximo do BRL aprendendo com apenas 25% dos dados.

7.5 Análise de convergência do aprendizado por reforço em lote com dados simulados

Conforme abordado no Capítulo 4, os algoritmos de aprendizado por reforço em Lote têm como características estabilidade e garantia de convergência para a solução ótima. A Figura 7.9 mostra uma análise da convergência do algoritmo de BRL usado em dados simulados. Para a análise foi implementado um simulador de negociações com base nas probabilidades de transição dadas

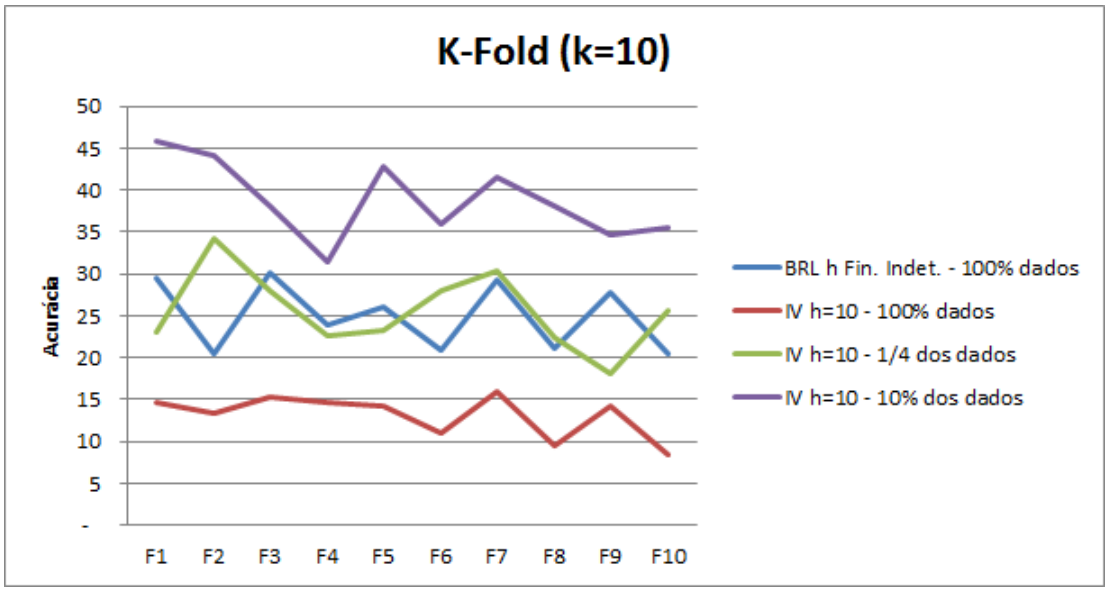

Figura 7.8: Teste de convergência dos algoritmos de Aprendizado BRL vs. IV modelando o problema como um MDP de horizonte finito.

pelo algoritmo de Iteração de Valor. Foram geradas amostras de tamanho n variando de 100 a 30.000 negociações completas. O algoritmo foi treinado em cada uma das amostras e teve sua convergência testada numa amostra distinta de mesmo tamanho.

7.6 Aprendizado por reforço em Lote com atualizações progressivas versus regressivas

Uma das vantagens do Aprendizado por Reforço em Lote nos casos em que as transições vieram de trajetórias completas é a possibilidade de acelerar a convergência para a política ótima aprendendo com cada processo em ordem regressiva, ou seja, na ordem contrária em que as transições aconteceram. Isso ocorre porque o valor de um estado depende dos valores dos possíveis estados futuros.

A Figura 7.10 mostra os resultados da análise de convergência do Aprendizado por Reforço em Lote aplicando atualizações em ordem progressiva e regressiva. Como é possível notar, as atualizações

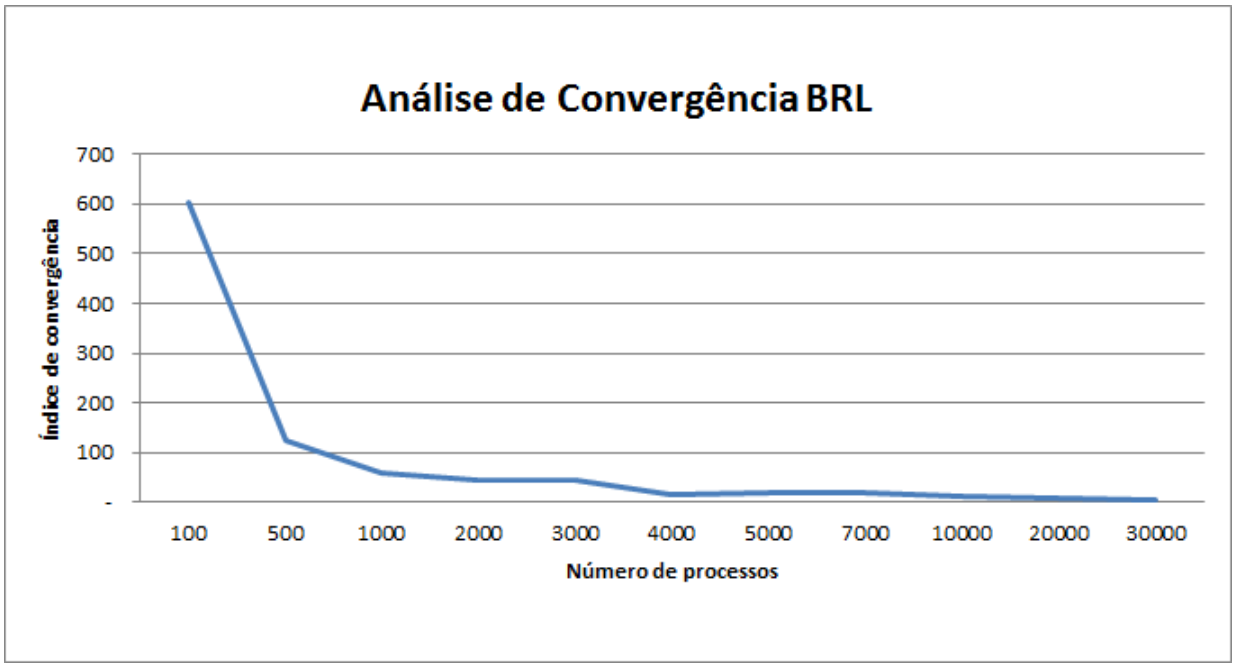

Figura 7.9: Teste de convergência do algoritmo de Aprendizado por Reforço em Lote usando dados simulados.

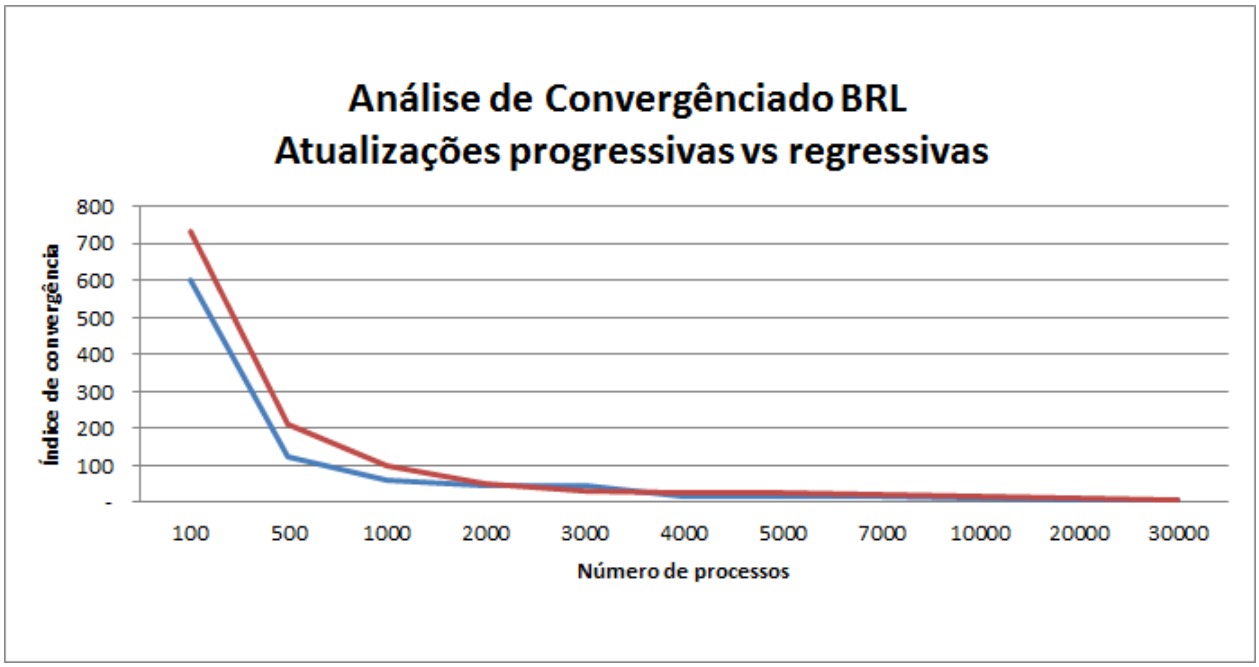

Figura 7.10: Teste de convergência do algoritmo de Aprendizado por Reforço em Lote usando dados simulados.

Capítulo 8

Conclusão

Como resultado desse trabalho podemos concluir que processos de venda podem ser modelados como Processos de Decisão Markovianos, em que a função de transição de estados é desconhecida. Com base em um histórico de negociações passadas é possível é possível encontrar boas políticas que convergem para a política ótima na medida em que novos dados são usados para o aprendizado

Identificados que os algoritmos usados aprendem com os melhores vendedores, e que é possível identificar quais vendedores precisam de treinamento, baseando-se na diferença entre a política adotada pelo vendedor e a política devolvida pelos algoritmos.

Identificamos que, embora o *Iteração de Valor* seja melhor, a técnica de *Aprendizado por Reforço em Lote* se adapta melhor as variações da política dos vendedores que muito provavelmente se deve a variações do mercado.

Num processo de venda não é comum o vendedor manter a negociação com o mesmo cliente indefinidamente. Usando como exemplo o problema abordado, não é razoável sugerir ao vendedor que continue insistindo numa mesma negociação caso ele já tenha executado inúmeras ações e esta permanece estagnada no mesmo estado. Portanto, modelar o problema como um MDP de horizonte finito é mais fiel ao problema real.

Identificamos que o algoritmo de *Iteração de Valor* modelando o problema como um MDP de horizonte $h = 10$ se mostrou muito mais acurado que o Aprendizado por reforço em lote. Uma explicação para isso vem das propriedades de convergência do algoritmo. Com o estado

expandido considerando o tempo t em que o estado foi alcançado, o número de amostras de cada par (estado, ação) diminui, diminuindo portanto a convergência do algoritmo para a política ótima.

Por fim indentificamos que é possível melhorar o Aprendizado por Reforço em Lote nos casos em que as transições vieram de tragetórias completas apenas aprendendo com cada processo na ordem contrária em que as transições aconteceram. Isso ocorre porque o valor de um estado depende dos valores dos possíveis estados futuros.

Apêndice A

Aprendizado Supervisionado

Aprendizado de máquina é uma sub-área da Inteligência Artificial que se dedica ao desenvolvimento de técnicas computacionais que permitam que uma máquina (agente) aprenda um novo comportamento com base em dados empíricos provenientes de sensores ou de banco de dados.

A.1 Introdução

Pesquisas na área de *aprendizado de máquina* lidam com a construção de agentes que possam *aprender* com suas experiências [Mitchell, 1997], ou seja, agentes que melhoram seu desempenho em uma deteminada tarefa através de uma dada *medida de desempenho*. Formalmente, um agente aprende a partir da experiência E, em relação a uma classe de tarefas T, com medida de desempenho P, se seu desempenho P em T melhora com E [Mitchell, 1997].

Usando como exemplo prático a tarefa de fazer *análise de crédito* teríamos:

- **Tarefa T:** classificar novos clientes como bons ou maus pagadores,
- **Medida de desempenho P:** porcentagem de clientes classificados corretamente,
- **Experiência de treinamento E:** uma base de dados em que clientes conhecidos são classificados como bons ou maus pagadores.

Figura A.1: Tipos de aprendizado.

Podemos dividir as técnicas de apendizado em 3 grupos mostrados na Figura A.1. Dizemos que o aprendizado é *supervisionado* quando é dado um conjunto de exemplos do comportamento desejado e *não supervisionado* quando é dado um conjunto de comportamentos quaisquer. Neste último caso, o aprendizado se dá através de uma medida que pode ser usada para identificar grupos de comportamentos, entre eles, o comportamento desejado. O *aprendizado por reforço* será detalhado no Capítulo 3.

A.2 Aprendizado supervisionado

No aprendizado supervisionado é fornecida uma referência do objetivo a ser alcançado, ou seja, o algoritmo de aprendizado recebe o valor de saída desejado para cada conjunto de dados de entrada apresentado. Neste caso, o processo de aprendizado significa aprender uma função a partir de exemplos de sua entrada e saída, e usar a função aprendida para outras entradas não previstas no treinamento.

Etapas do aprendizado supervisionado:

1. Coletar um grande conjunto de exemplos,
2. Dividir este conjunto em dois sub-conjuntos distintos: conjunto de treinamento e conjunto de teste,
3. Treinar o algoritmo de aprendizado utilizando o conjunto de treinamento,

4. Simular o algoritmo de aprendizado treinado no conjunto de testes e medir a porcentagem de exemplos corretamente classificados,

5. Repetir os passos de 1 a 5 para diferentes tamanhos de conjuntos de treinamento e diferentes conjuntos de treinamento

A.3 Aprendizado não-supervisionado

No aprendizado não-supervisionado é fornecido apenas o conjunto de dados de entrada sem a saída desejada. Em geral é utilizado como classificador e funciona encontrando aglomerados de conjuntos de dados semelhantes entre si (*clusters*). O processo não supervisionado é obviamente mais difícil que o supervisionado pois o sistema precisa definir sozinho o número de classes. Técnicas de aprendizado não supervisionado são muito utilizadas em *mineração de dados* e funcionam melhor quando as classes estão bem separadas no espaço de atributos.

As etapas do processo de aprendizagem não supervisionada envolvem:

1. **Seleção de atributos**: os atributos devem ser selecionados de forma que codifiquem a maior quantidade possível de informações evitando redundância entre eles,

2. **Medida de proximidade**: uma medida para quantificar a similaridade entre dois vetores de atributos,

3. **Critério de agrupamento**: pode ser compreendido como a sensibilidade que o algoritmo usado deve adotar,

4. **Algoritmo de clusterização**: após adotar uma medida de proximidade e um critério de agrupamento, o algoritmo de crusterização agrupa os dados em classes de objetos similares

5. **Validação dos resultados**: testes apropriados são usados para verificar o resultado.

6. **Interpretação dos resultados**

A.4 Validação dos sistemas de aprendizado

O principal objetivo de um sistema de Aprendizado é predizer com sucesso casos que ainda não foram vistos. Isto se torna bastante problemático quando um conjunto pequeno de dados de exemplo está disponível para treinamento. Nestes casos não se sabe dizer se tal número reduzido de dados para treinamento é suficiente para extrair o conhecimento necessário.

Existem inúmeros métodos utilizados para estimar o desempenho de um sistema de aprendizado. Alguns desses métodos podem ser tendenciosos ao apresentar estimativas muito otimistas ou muito pessimistas. Encontrar índices capazes de estimar de forma não tendenciosa o desempenho de um sistema de aprendizado é a base uma avaliação adequada dos sistemas de aprendizado.

A.5 Taxa de erro

Quando fornecemos uma entrada para um algoritmo de aprendizado de máquina este responde com modelo aprendido. O modelo aprendido pode ser o correto mas também pode conter erros. A taxa de erro é uma maneira bastante utilizada para medir a performance de algoritmos de aprendizado. Pode-se calcular a taxa de erro do algoritmo através da equação A.1.

$$E = \frac{\text{número de erros}}{\text{número de casos}} \tag{A.1}$$

Consideramos duas definições para a taxa de erro:

- **Taxa de erro amostral**: é a taxa de erro obtida utilizando somente os exemplos de treinamento, ou seja, o sistema aprende com um conjunto de exemplos e posteriormente é verificado quanto erro cometeu utilizando os mesmos exemplos com que foi treinado,
- **Taxa de erro real**: é a taxa de erro obtida sobre um conjunto muito grande de novos casos independentes dos casos usados para treinamento.

A taxa de erro real obviamente fornece uma excelente medida de acurácia de um algoritmo de aprendizado, no entanto, em casos reais conjuntos grandes de exemplo são raros e é muito comum a taxa de erro amostral ser muito inferior a taxa de erro real. Sendo assim, se a taxa de erro amostral fosse utilizada a estimativa de acurácia seria otimista e pouco confiável. Uma

alternativa é tentar se aproximar da taxa de erro real através de taxas de erros empíricas calculadas com pequenos conjuntos de exemplo.

Existem métodos estatísticos em que os algoritmos de aprendizado são executados com diferentes instâncias dos dados de amostra permitindo obter aproximações da taxa de erro real bem mais confiáveis que a taxa de erro amostral. Estes métodos serão apresentados na sessão A.6.

A.6 Estimativa da taxa de erro real

Como o cálculo da taxa de erro real é, na maioria das vezes, inviável e a taxa de erro amostral não é uma boa estimativa, uma alternativa é encontrar estimativas mais precisas para a taxa de erro real.

Um dos requisitos para estimar a taxa de erro real é manter a amostra em ordem aleatória, ou seja, a amostra não deve ser pré-selecionada evitando que qualquer cálculo seja feita sobre exemplo mais ou menos representativos.

A.6.1 Treinar e testar

O princípio dos métodos *Treinar e Testar* está em dividir a amostra em dois grupos mutuamente exclusivos e independentes. O primeiro, chamado de *conjunto de treinamento* é usado para treinar o sistema de aprendizado. O segundo, chamado de *conjunto de testes*, é usado apenas para medir a taxa de erro do sistema de aprendizado.

A.6.2 Reamostragem

Métodos de reamostragem são variações dos métodos *Treinar e Testar* nos quais são realizados inúmeros experimentos de Treinar e Testar com diferentes conjuntos de treinamento e teste. Em cada experimento é obtida uma taxa de erro, e a média dessas taxas de erro é a estimativa para a taxa de erro real. Para isso procura-se encontrar um estimador que não seja tendencioso, ou seja, que não estime a taxa de erro de forma otimista nem pessimista. Um estimador não tendencioso precisa convergir para o valor correto quando aplicado para uma amostra suficientemente grande. O estimador pode variar de amostra para amostra, mas a média sobre todas as amostras é correta.

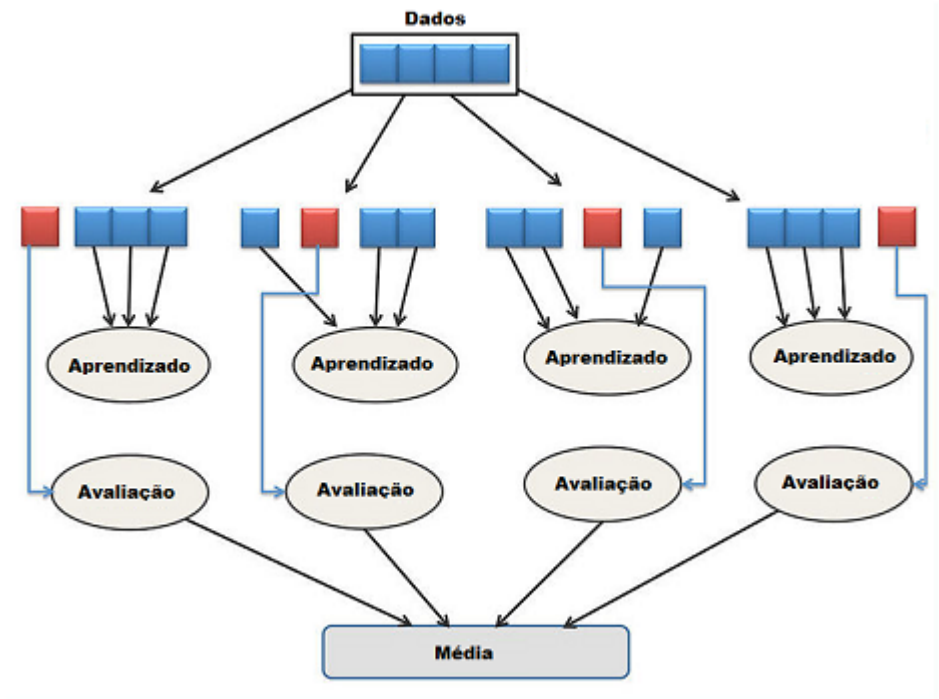

Figura A.2: Cross Validation. Método K-fold

A.6.3 Validação cruzada (Cross Validation)

Um dos métodos de estimativa de erro por reamostragem é o Validação Cruzada (Cross Validation). Também é conhecido como *K-fold Cross Validation* onde o *K* representa o número de partições da amostra geradas aleatoriamente.

Neste método a amostra é dividida em K partições mutuamente exclusivas e independentes. A cada iteração uma iteração diferente é usada para testar o sistema enquanto todas as K-1 partições são usadas para treinar o sistema. A taxa de erro é a média das taxas de erros calculadas nas K iterações.

Referências Bibliográficas

Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming, 1993. Citado na página 11.

R. E. Bellman. *Dynamic Programming*. Princeton University Press, USA, 1957. Citado 3 vezes nas páginas 1, 6 e 9.

B. Bonet and H. Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming, 2003. Citado na página 11.

Damien Ernst, Pierre Geurts, Louis Wehenkel, and L. Littman. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005. Citado 3 vezes nas páginas 2, 19 e 21.

M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers Inc., USA, 2004. Citado na página 5.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *CoRR*, cs.AI/9605103, 1996. URL `http://dblp.uni-trier.de/db/journals/corr/corr9605.html`. Citado 2 vezes nas páginas 1 e 13.

Shivaram Kalyanakrishnan and Peter Stone. Batch reinforcement learning in a complex domain. In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 650–657, New York, NY, USA, May 2007. ACM. ISBN 978-81-904262-7-5. Citado 2 vezes nas páginas 22 e 23.

T.M. Mitchell. *Machine Learning*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. ISBN 9780070428072. URL `http://books.google.com.br/books?id=nlmVQgAACAAJ`. Citado 2 vezes nas páginas 16 e 59.

M. L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994. Citado na página 9.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003. ISBN 0137903952. Citado na página 6.

Thomas Gabel Sascha Lange and Martin Riedmiller. Batch reinforcement learning. In M. Wiering and M. van Otterlo, editors, *Reinforcement Learning: State-Of-The-Art*, volume 12 of *Adaptation, Learning, and Optimization*, pages 45–73. Springer, Heidelberg, Germany, 2012. URL `http://www.springer.com/978-3-642-27644-6`. Citado 3 vezes nas páginas v, 20 e 23.

R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. Mit Press, 1998. ISBN 9780262193986. URL `http://books.google.com.br/books?id=CAFR6IBF4xYC`. Citado 4 vezes nas páginas v, 1, 13 e 14.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992. URL `http://jmvidal.cse.sc.edu/library/watkins92a.pdf`. Citado 2 vezes nas páginas 2 e 15.

Índice Remissivo

Aprendizado de Máquina, 59
- Aprendizado não-supervisionado, 61
- Aprendizado supervisionado, 60
- Definição, 59

Aprendizado por reforço, 13
- Definição, 13
- Métodos, 15
- Q-Learning, 15
  - $\epsilon - Greedy$, 16
  - Algoritmo, 16
  - Fator de aprendizado, 16

Aprendizado por reforço em banco de dados, 19
- Batch Reinforcement Learning, 19

Impressoras Plotters, 27
- Segmento criativo, 28
- Segmento PSP, 28
- Segmento técnico, 28

O Problema do Vendedor de Impressoras, 25
- Estágios do sistema, 29
  - Aceite da proposta comercial, 30
  - Conhecer melhor o Cliente, 30
  - Entrega, 31
  - Negócio perdido, 31
  - Pedido, 30
  - Pré Qualificação, 29
  - Proposta Comercial, 30
  - Secesso, 31
  - Standby, 31
- Sistema de tomada de decisão, 31
- Variáveis de estado, 32

O Processo de Venda, 25
- Ciclo de venda, 26
- Definição das necessidades do cliente, 26
- Elaboração de propostas, 26
- Fechamento do negócio, 26
- Negociação, 26
- Pós-venda, 26
- Prospecção, 25
- Qualificação do cliente, 26

Planejamento, 5
- Definição, 5
- MDP, 6
  - Equação de Bellman, 9
  - Horizonte, 7
  - Iteração de Valor, 9
  - Política, 7
  - Política Ótima, 8
  - Soluções Online, 10
- Planejamento Probabilistico, 6

Validação dos sistemas de aprendizado, 62
- Erro, 62
  - Erro amostral, 62
  - Erro real, 62
- Estimativa do erro real, 63
- Reamostragem, 63
- Treinar e testar, 63
- Validação cruzada, 64