FUNDAÇÃO GETULIO VARGAS
ESCOLA DE ECONOMIA DE SÃO PAULO

UIRÁ CAIADO DE CASTRO

DERIVAÇÃO DE MODELOS DE TRADING DE ALTA FREQUÊNCIA EM JUROS UTILIZANDO APRENDIZADO POR REFORÇO

SÃO PAULO

2017

UIRÁ CAIADO DE CASTRO

DERIVAÇÃO DE MODELOS DE TRADING DE ALTA FREQUÊNCIA EM JUROS UTILIZANDO APRENDIZADO POR REFORÇO

Dissertação apresentada ao Programa de Mestrado Profissional da Escola de Economia de São Paulo da Fundação Getúlio Vargas, como requisito para a obtenção do título de Mestre em Economia.

Área de concentração:
Finanças Quantitativas.

Orientador:
Prof. Dr. Afonso de Campos Pinto

SÃO PAULO
2017

Castro, Uirá Caiado de.
Derivação de modelos de trading de alta frequência em juros utilizando aprendizado por reforço / Uirá Caiado de Castro. - 2017.
66 f.

Orientador: Afonso de Campos Pinto
Dissertação (mestrado profissional) - Escola de Economia de São Paulo.

1. Aprendizado do computador. 2. Modelos econômicos. 3. Ações (Finanças). 4. Investimentos - Análise. I. Pinto, Afonso de Campos. II. Dissertação (mestrado profissional) - Escola de Economia de São Paulo. III. Título.

CDU 336.76

UIRÁ CAIADO DE CASTRO

DERIVAÇÃO DE MODELOS DE TRADING DE ALTA FREQUÊNCIA EM JUROS UTILIZANDO APRENDIZADO POR REFORÇO

Dissertação apresentada ao Programa de Mestrado Profissional da Escola de Economia de São Paulo da Fundação Getúlio Vargas, como requisito para a obtenção do título de Mestre em Economia.

Área de concentração:
Finanças Quantitativas.

Data da Aprovação: 24 / 08 / 2017

Banca Examinadora:

Prof. Dr. Afonso de Campos Pinto
(Orientador)
EESP - FGV

Prof. Dr. Alessandro Martim Marques
EESP - FGV

Prof. Dr. Renato Vicente
IME - USP

Aos meus pais e irmãos. Cada um de vocês fez parte do que sou hoje.

Agradecimentos

Agradeço à minha mãe, por sempre me incentivar a perseguir meus objetivos, sejam eles loucos ou não. Agradeço à minha querida Marianna, pela paciência e companheirismo ao trilhar junto comigo a realização de um mestrado. Agradeço também aos professores e aos colegas do MPEFQ-2015 e, por fim, ao Prof. Dr. Alessandro Martim Marques, pelas inúmeras revisões realizadas e confiança depositada.

"In order to succeed you must first survive"
(Warren Buffett)

RESUMO

O presente estudo propõe o uso de um modelo de aprendizagem por reforço para derivar uma estratégia de *trading* em taxa de juros diretamente de dados históricos de alta frequência do livro de ofertas. Nenhuma suposição sobre a dinâmica do mercado é feita, porém é necessário criar um simulador com o qual o agente de aprendizagem possa interagir para adquirir experiência. Diferentes variáveis relacionadas a microestrutura do mercado são testadas para compor o estado do ambiente. Funções baseadas em P&L e/ou na coerência do posicionamento das ofertas do agente são testadas para avaliar as ações tomadas. Os resultados deste trabalho sugerem algum sucesso na utilização das técnicas propostas quando aplicadas à atividade de *trading*. Porém, conclui-se que a obtenção de estratégias consistentemente lucrativas dependem muito das restrições colocadas na aprendizagem.

Palavras-chave: Aprendizado de máquina. Aprendizado por reforço. *Q-learning*. Curva de juros. Operação de alta frequência.

ABSTRACT

The present study proposes the use of a reinforcement learning model to develop an interest rate trading strategy directly from historical high-frequency order book data. No assumption about market dynamics is made, but it requires creating a simulator wherewith the learning agent can interact to gain experience. Different variables related to the microstructure of the market are tested to compose the state of the environment. Functions based on P&L and/or consistency in the order placement by the agent are tested to evaluate the actions taken. The results suggest some success in bringing the proposed techniques to trading. However, it is presumed that the achievement of consistently profitable strategies is highly dependent on the constraints placed on the learning task.

Keywords: Machine Learning. Reinforcement Learning. *Q-learning*. Yield Curve. High frequency trading.

Lista de ilustrações

Lista de tabelas

Lista de algoritmos

Sumário

1 Introdução

Atualmente, a área de inteligência artificial (IA) tem frequentemente dominado os noticiários e o imaginário popular. Muitos filmes e séries têm explorado cenários um tanto quanto tenebrosos dos possíveis impactos desta área sobre o cotidiano das pessoas.

Tais prognósticos negativos não estão apenas na ficção. Frey e Osborne (2017) estimam que 47% das vagas de emprego, apenas nos Estados Unidos, são passíveis de serem automatizadas futuramente por técnicas que estão sendo desenvolvidas neste campo. Mesmo hoje, os modelos de negócio de algumas das empresas mais valiosas do planeta talvez nem fossem possíveis se não fosse por técnicas de inteligências artificial[1].

Porém, não existe consenso quando se fala em IA. Por exemplo, contrastando com o que foi mencionado acima, em relatório publicado pela Organização para Cooperação e Desenvolvimento Econômico (OCDE), Arntz, Gregory e Zierahn (2016) argumentam que apenas 9% dos empregos estariam realmente em risco entre seus 21 membros.

Além disso, ainda que haja sinais claros do impacto deste campo em diferentes mercados, ou da capacidade para tal, em algumas indústrias, como a de investimentos, isso não está tão evidente. Hall e Kumar (2017) reportaram, por exemplo, que o índice Eurekahedge AI Hedge Fund, que segue 12 fundos que usam IA como parte de suas principais estratégias, apresentaram retorno de cerca de 9% ao ano entre 2011 e 2015. Estes ganhos foram superiores a média de outros *hedge funds*, porém não superou o retorno do S&P 500 no mesmo período.

Apesar de a performance apresentada por estes fundos não ser muito expressiva e a taxa de falhas de estratégias desenvolvidas com IA ser da ordem de 90%[2], grandes *players* do mercado, como o Two Sigma Investments, o Citadel e o Man Group PLC apostam no potencial deste campo. Este último, por exemplo, através do seu braço *quant* Man AHL, precisou de cerca de três anos de trabalho para conseguir desenvolver estratégias baseadas em IA nas quais tivesse confiança suficiente para alocar o capital de seus clientes.

O comprometimento de recursos destes fundos em formas alternativas de investimento não é por acaso. Segundo levantamento efetuado pela KPMG International (2016), 94% dos gestores de *hedge funds* acreditam que inovação é um diferencial competitivo. Neste contexto, pode-se enxergar a inteligência artificial como uma fonte desta inovação, argumento corroborado por 58% dos entrevistados, que acreditam que técnicas de IA terão impacto entre médio e alto na maneira como os fundos de investimento operarão no futuro.

Assim, considerando a importância dada para inovação pelo setor e do potencial

[1] *Facebook's AI chief: "Facebook today could not exist without AI"*. Disponível em <https://goo.gl/ai3clx>

[2] Estatística extraída de Hall e Kumar (2017).

do uso de técnicas de inteligência artificial no contexto de investimentos, esta dissertação desenvolverá uma estratégia adaptativa de *trading* de alta frequência para operar contratos futuros de taxa de juros utilizando uma técnica relacionada à IA chamada aprendizado por reforço. Como será melhor explorado nos próximos capítulos, modelos desta classe têm como objetivo encontrar uma estratégia ótima para um problema de decisões em sequência por tentativa e erro, interagindo diretamente com um ambiente.

A princípio, esta abordagem não exige nenhuma suposição sobre a dinâmica do mercado, como formação de preço ou chegada de novas ofertas. Porém, exige que exista um ambiente com o qual o agente possa interagir para adquirir experiência. Assim, um simulador de livro de ofertas será criado. Este simulador utilizará dados históricos de alta frequência para reproduzir múltiplos mercados de forma ordenada.

O problema de *trading* será caracterizado como um problema de aprendizado por reforço e o agente criado utilizará um modelo chamado *Q-learning* para derivar uma estratégia de *trading*, adaptando-se a diferentes instrumentos e períodos. Diferentes representações de estado do ambiente e noções de recompensa, que indicam ao agente se ele está certo ou errado, serão testados. A performance deste agente, então, será medida em uma base de dados diferente da utilizada para treinamento e será comparada ao desempenho médio de um agente que tome decisões aleatoriamente. A diferença entre a performance entre ambos os agentes será testada estatisticamente, com um nível de significância de 10%. Segundo as pesquisas deste autor, ainda nenhum trabalho do gênero foi realizado no mercado brasileiro e espera-se, com esta dissertação, incentivar outros a explorar este campo.

Este trabalho está dividido em seis capítulos. O capítulo dois apresenta uma revisão bibliográfica, contextualizando o trabalho em relação aos estudos já realizados com outras técnicas de inteligência artificial e estudos relacionados especificamente à aprendizado por reforço. No terceiro capítulo é apresentada a teoria que fundamenta o *framework* explorado, bem como o modelo que será desenvolvido nesta dissertação. O capítulo quatro apresenta todas as etapas necessárias para o tratamento dos dados utilizados, para criação do simulador mencionado e implementação do modelo proposto. Também são apresentados o *benchmark* escolhido e o desempenho inicial do modelo criado. No capítulo cinco são descritas todas as etapas para especificação final do modelo e é feita uma comparação de seu desempenho contra o *benchmark* escolhido. Por fim, no capítulo seis, é apresentada a conclusão deste trabalho e possíveis extensões.

2 Revisão Bibliográfica

A área de inteligência artificial evoluiu sobremaneira nos últimos anos. Graças à avanços em alguns de seus subdomínios, como *deep learning* e aprendizado de máquina, atualmente, uma miríade de ferramentas que utilizam técnicas de IA fazem parte do cotidiano de muitas pessoas.

Exemplos de aplicações podem ser encontrados no assistente de voz da Apple, no *feed* de notícias de cada usuário no Facebook, nos caminhos sugeridos pelo Waze e nos carros autoguiados da Tesla. Ainda que muitas destas técnicas tenham se originado em aplicações com pouca ou nenhuma relação com finanças, muitas pesquisas têm sido feitas em como adaptar tais algoritmos ao mercado financeiro. Roncalli (2014) elenca alguns trabalhos acadêmicos que procuraram fazer esta adaptação, aplicando modelos de aprendizado de máquina, ou *machine learning* (ML), à *asset management*. O autor alega que o desempenho de alguns dos algoritmos explorados foram tão bons ou superiores às técnicas tradicionais de finanças quando aplicados à *bond scoring*, *hedge fund tracking*, entre outros.

Fletcher (2012) encontra evidências similares em uma pesquisa extensa que fez com diferentes algoritmos de ML para previsão de séries temporais financeiras. Mostrou algum sucesso na aplicação destes modelos e sugere que variáveis[1] baseadas na microestrutura do mercado são úteis para prever movimentos em curtos espaços de tempo.

Outra pesquisa interessante na área foi realizada por Li et al. (2014), onde utilizam um algoritmo não-paramétrico de classificação chamado *support vector machine* (SVM) para aprender a relacionar os movimentos futuros dos ativos com padrões do livro de ofertas[2] e notícias de mercado. Estes movimentos foram classificados como positivos, negativos ou neutros e utilizados para ajudar uma estratégia de *market making* a decidir onde colocar suas ofertas. Este algoritmo foi atualizado diariamente durante o período de validação, testando o novo modelo no dia subsequente aos utilizados para treinamento. Segundo os autores, este cuidado se justifica, pois a microestrutura do mercado muda ao longo dos dias e padrões encontrados são mais consistentes durante períodos curtos de tempo.

O algoritmo de SVM, assim como os modelos testados nas outras pesquisas citadas até agora, são exemplos de aprendizado supervisionado, uma das grandes áreas de *machine learning*. Como explicado por Hastie, Tibshirani e Friedman (2009), estas técnicas são chamadas de aprendizado supervisionado, ou *supervized learning*, porque, para cada

[1] Na literatura de ML, estas variáveis são geralmente chamadas de *features*.
[2] O livro de ofertas também será referido como *book* de ofertas, ou apenas *book*.

observação na base (*input*), há uma variável de resultado (*output*), que é usada para guiar o processo de aprendizagem. Uma outra área de ML que está em ativo desenvolvimento é o aprendizado por reforço, ou *reinforcement learning* (RL). Como exposto em Ferrucci et al. (2010) e Silver et al. (2016), tal técnica, associada ao *deep learning*, está por trás de feitos recentes na área e inteligência artificial, em que as máquinas superaram adversários humanos em jogos como o *Go* e o *Jeopardy*.

A principal diferença entre aprendizagem por reforço e a supervisionada é que pares do tipo *input*/*output* não estão disponíveis no primeiro caso. Em vez disso, o agente precisa coletar estas informações ativamente, interagindo repetidamente com o ambiente para observar as recompensas (ou punições) imediatas resultantes de cada ação executada. Não é dito ao agente qual ação geraria o maior resultado a longo prazo, ele precisa inferir esta informação por tentativa e erro. Logo, uma maneira de se definir *reinforcement learning* é como o estudo do planejamento e da aprendizagem em situações em que o aprendiz (ou agente) interage ativamente com o ambiente para atingir um objetivo específico. (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012; KAELBLING; LITTMAN; MOORE, 1996).

Dada a natureza iterativa dos algoritmos de RL, percebe-se que o problema da criação de estratégias de *trading* pode ser convenientemente modelado sob tal *framework*. Como colocado por Spooner (2016), ao invés de criar as regras de apregoação explicitamente, caracteriza-se o problema envolvido dentro de uma estrutura de RL e então se permite que o agente interaja com o ambiente criado para tentar aprender a agir de maneira ótima. Assim, considerando que S e A são, respectivamente, os conjuntos de todos os estados e ações possíveis, Mitchell (1997) explica que a tarefa deste agente será aprender uma política π que mapeie cada estado em S em uma ação em A ($\pi : S \rightarrow A$). A próxima ação a_t é selecionada com base unicamente no atual estado observado s_t, de maneira que $\pi(s_t) = a_t$. A política ótima, ou estratégia de controle, é aquela que produzirá a maior recompensa acumulada ao longo do tempo.

De acordo com Spooner (2016), apesar do campo de RL estar em ativo desenvolvimento, são poucas as pesquisas combinando esta área com *algo trading*, e menos ainda com *high frequency trading*. Um dos primeiros trabalhos nesta direção pode ser encontrado em Chan e Shelton (2001), onde os autores conseguiram resultados interessantes ao desenvolverem uma estratégia de *market making* usando RL. Os agentes criados aprenderam explicitamente a formar mercado interagindo diretamente com uma versão simplificada do livro de ofertas de um mercado artificial. Foram testadas diferentes representações de estados, funções de recompensas e métodos de aprendizagem por reforço, como o *SARSA* e o *actor-critic*. A tarefa dos agentes criados foi aprender a modular os *spreads* utilizados para posicionar suas ofertas, observando, por exemplo, o excesso de demanda ou oferta no mercado. As funções de recompensa testadas buscaram motivar os agentes à maximizar o resultado e/ou reduzir o *bid-ask spread*. Todos os valores foram discretizados antes de

serem passados para os algoritmos.

Kim, Shelton e Poggio (2002) continuaram o trabalho de Chan, onde Shelton também é coautor, derivando uma estratégia de *market making* lucrativa utilizando os mesmos algoritmos de aprendizado por reforço da pesquisa anterior. Porém, usam uma simulação de mercado mais complexa, onde modelam o *book* de ofertas a partir de dados reais, estimando as distribuições de probabilidade associadas ao tamanho das ordens, ao tempo de chegada de novas ofertas a mercado, ordens limitadas e outros parâmetros pertinentes ao livro de ofertas. O resultado desta pesquisa demonstra que o sucesso da estratégia de aprendizado não está relacionado aos detalhes de algum modelo de mercado específico, uma vez que o mesmo método conseguiu apresentar resultados satisfatórios em diferentes tipos de simulações.

Em trabalho recente, o já citado Spooner (2016) também obteve sucesso ao derivar uma estratégia de *market making*. Seu trabalho diverge da literatura anterior principalmente em dois pontos. Primeiro, em vez de discretizar as variáveis de estado, ele emprega um método de aprendizado supervisionado chamado *tile coding* para atualizar seus modelos de RL. Além disso, não usa um modelo artificial de mercado, mas reconstrói o *book* de ofertas diretamente, usando dados históricos de alta frequência. Com esta estrutura, analisa o impacto de diferentes representações de estado na política aprendida pelos agentes. Também investiga possíveis métodos para diminuir a variabilidade dos resultados dos agentes utilizando diferentes funções de *reward* que incluem algum tipo de medida de risco. Por fim, compara a performance dos modelos *Q-Learning*, *double Q-Learning* e *SARSA*.

Du, Zhai e Lv (2016) desenvolvem uma estratégia de *trading* que, a cada passo, decide se compra ou vende um determinado ativo baseado apenas na direção anterior do mercado. Compararam os resultados de um algoritmo clássico de RL chamado *Q-learning* com outra técnica chamada *recurrent reinforcement learning* (RRL). Entre as funções de recompensa testadas, usaram o P&L acumulado da estratégia e o *Sharpe ratio*. Concluem que as simulações usando o RRL produziu resultados mais estáveis que aquelas que usaram *Q-learning*. Também sugerem que uma escolha adequada da função de *reward* foi vital para atingir a performance ótima em seus testes.

Kearns e Nevmyvaka (2013) investigam o uso de RL em duas situações diferentes. Na primeira, otimizam um algoritmo de execução e mostraram que o método superou estratégias comuns utilizadas na indústria. No segundo caso, eles derivam um modelo para prever movimentos no *mid-price* de curtíssimo prazo. Apesar de terem conseguido um sucesso moderado neste estudo, explicam que o resultado observado deixaria de existir se considerassem custos de execução. Concluem enfatizando a importância do processo de *feature design* e sugerem que nenhuma aprendizagem é possível se o modelo não acessar informações relevantes para resolver o problema em questão.

Lee et al. (2007) desenvolvem uma estratégia de *trading* com periodicidade diária

aplicando *Q-learning*. Dividem o problema proposto em várias etapas e usam diferentes agentes para resolver cada uma delas, empregando diferentes funções de recompensa e representação de estados em cada caso. Outro exemplo de *reinforcement learning* aplicado à modelos de *trading* com periodicidade diária pode ser encontrado em Cumming, Alrajeh e Dickens (2015), onde desenvolvem uma estratégia para operar câmbio no mercado de *Forex*.

O presente estudo propõe desenvolver um modelo de *trading* para operar contratos futuros de taxas de juros utilizando o *framework* de *reinforcement learning*. Assim como Spooner (2016), este trabalho também reconstrói o *book* de ofertas utilizando dados históricos, além de implementar o mesmo método de generalização para tratar variáveis contínuas na representação de estado. Inspirado na abordagem de Li et al. (2014), alguns aspectos da estratégia não serão aprendidos, mas dados pelo ambiente. Como enfatizado nos diferentes trabalhos pesquisados, também será analisado o impacto de diferentes representações de estado e funções de *reward* nas políticas que o agente é capaz de aprender. Esta pesquisa utilizará o algoritmo de *Q-learning* para suportar a aprendizagem.

3 Modelo Proposto

A seguir será apresentada a estrutura matemática e nomenclaturas utilizadas ao longo deste trabalho. A teoria que fundamenta o aprendizado por reforço será descrita brevemente, assim como os algoritmos que serão utilizados nesta dissertação. Por fim, será discutida a abordagem utilizada para o tratamento de espaço de estados com alta dimensionalidade.

3.1 Fundamentos de *reinforcement learning*

A ideia de aprender por iteração com o ambiente é provavelmente uma das primeiras que vem a tona quando se pensa na natureza da aprendizagem. Seja quando aprende-se a dirigir ou a manter uma conversa, o indivíduo está sempre atento em como o ambiente com o qual está interagindo responde ao que faz, procurando influenciar o que ocorre através de seu comportamento (SUTTON; BARTO, 2017).

Spooner (2016) explica que o aprendizado por reforço, uma área de *machine learning* que tem raízes na teoria de controle ótimo e na psicologia, é o estudo de como um tomador de decisão - chamado de *agente* - interage com o *ambiente* com o objetivo de maximizar alguma noção de *recompensa*. O ambiente, neste *framework*, compreende tudo que não seja o próprio agente. A tarefa de tal agente é se adaptar a este ambiente, dadas algumas experiências passadas, de maneira que suas iterações futuras maximizem o valor das recompensas possíveis ao longo de um período tempo chamado *episódio*.

Abaixo serão explorados alguns dos conceitos e elementos do *framework* de aprendizado por reforço. Em seguida, na seção 3.2, será mostrado com detalhes o algoritmo sobre o qual baseou-se este trabalho.

3.1.1 Elementos básicos do problema

Além do agente e do ambiente, Waskow (2010) explica que qualquer problema de RL é composto por pelo menos mais três componentes: a política π, a função de recompensa $r(s, a)$ e a função de valor $\nu(s)$ ou $q(s, a)$.

Sutton e Barto (2017) explicam que a política π define a maneira como o agente se comporta a qualquer passo de tempo t. Basicamente, ela faz o mapeamento dos estados do ambiente, percebidos pelo agente, para ações a serem tomadas nestas situações. A cada passo, o ambiente envia para o agente um único sinal escalar, chamado de recompensa. O único objetivo do agente, então, é maximizar o total de recompensas que ele recebe ao longo do tempo. A função de recompensa $r(s, a)$ define o que é um evento bom ou

ruim para o agente. Segundo Waskow (2010), é esta função que define o problema a ser resolvido.

Por fim, enquanto a recompensa indica o que é bom em um sentido imediato, a função de valor $\nu(s)$ ou $q(s, a)$ indica o que é bom a longo prazo. Sutton e Barto (2017) explicam que o valor de um estado é quanto o agente pode esperar acumular com o passar do tempo, começando daquele estado s. Enquanto um estado pode sempre resultar em uma recompensa negativa imediatamente, ainda pode ter um valor alto porque geralmente é seguido por outros estados que geram recompensas maiores.

Assim, a recompensa r é fundamental no problema de RL porque ela permite a estimação do valor dos estados. Porém, o agente não procura por ações que gerem recompensas imediatamente mais altas, mas sim por ações que levem para estados que tenham a maior valor de recompensa esperada ao longo do tempo . Este conceito, também chamado de retorno, será definido a seguir.

3.1.2 Recompensa e objetivo de aprendizado

Em problemas de RL, o objetivo do agente é maximizar o total de recompensas acumuladas ao longo de um episódio. Um episódio compreende um período de tempo composto por passos discretos de tempo $t = 0, 1, 2, 3, ..., T; T \leq \infty$. As recompensas são valores numéricos calculados por uma função $r(s, a)$, também chamada de função de recompensa ou função *reward*. Elas são resultantes da aplicação uma ação $a_t \in A(s_t)$ no estado $s_t \in S$, onde $A(s_t)$ é o conjunto de todas ações possíveis no estado s_t e S, o conjunto de todos os estados possíveis.

Sutton e Barto (2017) explicam que é imperativo que a função *reward* utilizada realmente indique o que se deseja que seja aprendido pelo agente. Segundo eles, esta função *não* é o lugar adequado para transmitir para o agente algum conhecimento prévio em como atingir aquele objetivo que se espera que ele atinja. Por exemplo, um agente jogador de xadrez deve ser recompensado apenas quando ganhar a partida, não por atingir sub-objetivos, como capturar as peças do oponente ou controlar o centro do tabuleiro. Se atingir estes tipos de sub-objetivos for recompensado, o agente pode encontrar maneiras de atingi-los e não atingir o objetivo principal. No caso do jogador de xadrez, o agente poderia decidir capturar as peças do oponente, mesmo que isso resultasse em perder o jogo. A função de recompensa é a maneira de comunicar ao agente o que se quer que ele atinja, não como ele deve atingir isso.

Neste ponto é importante diferenciar a noção de recompensa imediata e acumulada. Como colocado por Spooner (2016), a recompensa imediata, por ela mesma, não determina como o agente deverá agir. Em vez disso, ela é apenas um valor escalar que informa o agente quão bem ele está se comportando. Já a recompensa acumulada pode ser definida

como a soma da sequência destes valores escalares em uma quantidade finita de passos depois do tempo t : $r_{t+1}, r_{t+2}, r_{t+3} + ... + r_{t+T}$. Uma função G_t, dependente desta sequência de recompensas, pode ser definida como

$$G_t \coloneqq \sum_{k=0}^{T-1} r_{t+k+1} \ , \tag{3.1}$$

onde T denota o passo final ou estado terminal do episódio. Desta maneira, o objetivo do agente é maximizar o valor esperado $\mathbf{E}\left[G_t\right]$. Porém, como mencionado por Sutton e Barto (2017), em casos onde o episódio não tem um estado terminal ($T = \infty$), a equação (3.1) pode resultar em um valor acumulado infinito. Evita-se isso incluindo um fator de desconto na função, de maneira que o objetivo do agente passa a ser selecionar ações tal que a expectativa da soma das recompensas recebidas no futuro, descontadas por algum fator, sejam maximizadas. Desta forma, temos

$$G_t \coloneqq r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \ , \tag{3.2}$$

onde o parâmetro $\gamma \in [0, 1]$ é o fator de desconto. De acordo com Mitchell (1997), γ é uma constante que determina o valor relativo da recompensa imediata e adiada. No caso em que $\gamma = 0$, apenas as recompensas imediatas são consideradas. A medida que $\gamma \to 1$, se dá mais ênfase para recompensas futuras do que para as imediatas. A função da equação (3.2) também é chamada de função de retorno.

3.1.3 A hipótese de Markov

Sutton e Barto (2017) explicam que o estado s se refere a qualquer informação que estiver disponível para o agente no momento que consultar o ambiente. A representação de estados pode ser uma versão pré-processada das variáveis que o agente pode observar, muitas vezes referidas como sensações, ou ainda uma estrutura complexa formada pela sequência destas observações ao longo do tempo.

Idealmente, o estado deveria sumarizar as sensações passadas de forma compactada, sem perder nenhuma informação relevante. Dizemos que este sinal que retém todas as informações relevantes seguem a hipótese de Markov. Como definido por Shreve (2012), se para todo n entre 0 e $N-1$ e para toda função $f(x)$, existir outra função $g(x)$, dependente de f e n, de maneira que $\mathbf{E}_n\left[f(X_{n+1})\right] = g(X_n)$, dizemos que a sequência de variáveis aleatórias X_0, $X_1, ..., X_N$ é um processo de Markov. Assim, uma variável segue a hipótese de Markov se toda informação necessária para resolver $\mathbf{E}_n\left[f(X_{n+1})\right]$ está contida em X_n.

Considerando que o estado s satisfaz tal hipótese, podemos expressar a dinâmica de como o ambiente responderia no tempo $t+1$ à ação a, tomada por um agente no tempo t, como

$$p(s',\, r\,|s, a) = Pr\{S_{t+1} = s'\,, R_{t+1} = r\,|S_t = s\,, A_t = a\}\ , \tag{3.3}$$

para todo r, s', s e a. Essa dinâmica de passo único, sob a hipótese de ambiente markoviano, possibilita prever o próximo estado e recompensa esperada observando apenas o estado e ação atuais. Sutton e Barto (2017) também argumentam que a política ótima encontrada por escolher-se ações em função de estados markovianos é tão boa quanto a melhor política encontrada por escolher-se ações em função de todo o histórico de observações.

Os autores seguem argumentando que, mesmo quando a representação de estado é não-markoviana, ainda é apropriado considerá-la como sendo uma aproximação de estados markovianos. Entretanto, para isso ser efetivo, deve-se procurar cuidadosamente por estados informativos o suficiente para servir como base para prever recompensas futuras e, consequentemente, para selecionar as melhores ações.

3.1.4 Processo de decisão de Markov

Muitos dos algoritmos da moderna teoria de aprendizado por reforço são fundamentados no pressuposto de que o problema de controle em questão satisfaz a hipótese de Markov e que, portanto, a iteração entre agente e ambiente pode ser modelado como um processo de decisão de Markov (SPOONER, 2016).

Termo cunhado por Bellman (1954), o *decision Markov process* (MDP) é um processo estocástico em que a distribuição de probabilidades condicionais para os estados futuros depende apenas do estado atual. Se o conjunto de estados e o conjunto de ações forem finitos, o MDP também será finito. Sutton e Barto (2017) pontuam que a maior parte da teoria atual de RL é restrita a estes MDPs finitos, porém, as ideias e os métodos relacionados à eles podem ser aplicados de forma mais geral.

Segundo Waskow (2010), a dinâmica de um MDP finito é definida por um processo estocástico de um passo, em que o último estado observado e ação tomada são suficientes para determinar as probabilidades de cada estado possível e recompensa resultante destes. Esta dinâmica foi apresentada na equação (3.3), que é repetida abaixo,

$$p(s',\, r\,|s,a) = Pr\{S_{t+1} = s'\,, R_{t+1} = r\,|S_t = s\,, A_t = a\}\ .$$

A partir desta dinâmica do ambiente, Sutton e Barto (2017) argumentam que pode-se extrair diversas informações do ambiente caso ele seja um MDP, como por exemplo, a probabilidade de transição do estado s para o s', tomando a ação a ($s \xrightarrow{a} s'$), é dada por

$$p(s'\,|s,a) = Pr\{S_{t+1} = s'\,|S_t = s\,, A_t = a\} = \sum_{r\in\mathbf{R}} p(s',\, r\,|s,a). \tag{3.4}$$

Analogamente a recompensa imediata esperada para um determinado par estado-ação decorrente das possíveis transições de estado

$$r(s,\, a) = \mathbf{E}\left[R_{t+1}|\, S_t = s,\, A_t = a\right] = \sum_{r\in\mathbf{R}} r \sum_{s'\in\mathbf{S}} p(s',\, r\,|s,a)\ . \tag{3.5}$$

Spooner (2016) argumenta que estes MDPs são adequados para modelar problemas episódicos, onde existe um estado inicial e um estado terminal. Nestes episódios, o sistema evolui a partir do estado inicial s_0 de acordo com as ações tomadas pelo agente. O estado terminal, que corresponde à algum subconjunto do conjunto de estados possíveis, é o ponto onde o processo termina.

Segundo Mitchell (1997), a tarefa do agente é aprender uma política π, definida como o mapeamento $\pi : S \to A$, que determine qual ação tomar em cada estado do ambiente. Assim, para cada estado no MDP, começando em s_0, o agente escolhe a próxima ação $a_0 = \pi(s_0)$ que fará com que o ambiente faça uma transição para um novo estado s' com probabilidade $p(s' \,|s, a)$, gerando uma recompensa numérica $r_0 = r(s_0,\, a_0)$. Spooner (2016) segue explicando que a sequência de iterações $(s_0,\, a_0,\, r_0)$, $(s_1,\, a_1,\, r_1), ...,$ $(s_T,\, a_T,\, r_T)$ continua até o estado terminal T ser atingido.

Waskow (2010) explica que a solução do MDP consiste em encontrar a política ótima π^* que resultará na sequência de ações que maximiza a recompensa esperada para o sistema. Como Mohri, Rostamizadeh e Talwalkar (2012) expõe, existem alguns métodos para estimar a política ótima, como iteração da política e programação linear. No presente trabalho, será explorado um dos métodos que envolve a iteração da função de valor.

3.2 Função de valor

Segundo Sutton e Barto (2017) , a grande maioria dos algoritmos de aprendizado por reforço envolve o uso de algum método para estimar funções que respondam quão bom é para o agente estar em um determinado estado - as chamadas funções de valor. Em problemas de RL, a noção de *"quão bom"* é definida por quanto o agente pode esperar receber de recompensas no futuro, estando naquele estado e tomando uma determinada ação. Abaixo, é definido o que é a função ótima de valor e é apresentado o algoritmo que será utilizado para estimá-la.

3.2.1 Função ótima de valor

O objetivo de um agente em problemas de RL é achar uma política ótima que maximize o retorno obtido, começando em um estado inicial s qualquer. Este retorno corresponde à agregação das recompensas recebidas ao longo de uma trajetória, começando de s. Seguindo explicações Sutton e Barto (2017), usando a definição de retorno na equação (3.2), podemos definir o retorno descontando esperado $\nu_\pi(s)$ para um horizonte infinito de tempo seguindo uma política π dada por

$$\nu_\pi(s_t) \coloneqq \mathbf{E}_\pi \left[G_t \,|\, S_t = s \right] = \mathbf{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|\, S_t = s \right], \tag{3.6}$$

onde t é um passo de tempo qualquer e $\mathbf{E}_\pi[.]$ é o valor esperado de uma variável aleatória, dado que o agente segue uma política π. Chamamos ν_π de função estado-valor por seguir a política π. Logo, a política ótima π^*, que maximizará $\nu_\pi(s_t)$ para todos os estados s, pode ser escrita como

$$\pi^* = \arg\max_\pi \nu_\pi(s) \ , \ \forall s \in S \ ; \tag{3.7}$$

assim o retorno esperado por seguir uma política ótima é, portanto, ν_{π^*}, que será referido, por brevidade, apenas como ν_*.

Mitchell (1997) explica que aprender a política $\pi^* : S \to A$ diretamente é uma tarefa complicada. Para ser possível o mapeamento das ações a partir de estados seria necessário que os dados disponíveis para treinamento estivessem na forma $\langle s, a \rangle$, o que não é o caso. O agente só tem disponível a sequência de recompensas imediatas produzidas pela função $r(s_t, a_t)$ para $t = 1, 2, 3, ..., T$. Como se deseja maximizar o retorno esperado $\nu_\pi(s_t)$ para todos os estados s, o agente sempre deve preferir s_1 à s_2 quando $\nu_\pi(s_1) > \nu_\pi(s_2)$.

Considerando que o agente deve escolher entre ações, e não estados, e que não é possível antecipar com precisão as recompensas e estados resultantes de todas transições geradas pelas diferentes combinações de estado-ação existentes, também se deve aprender ν_π indiretamente.

Busoniu et al. (2010) afirmam que uma forma conveniente de caracterizar as políticas é usando suas funções de valor, que são divididas em dois tipos: as já mencionadas funções de estado-valor (equação (3.6)), que são frequentemente referidas como *V-functions* na literatura, e as funções de ação-valor, também chamadas de *Q-functions*.

A *Q-function* $Q^\pi : S \times A \to \mathbb{R}$ de uma política π indica o retorno esperado quando o agente começa no estado s, aplica uma ação a e depois continua escolhendo novas ações seguindo uma política π, de modo que

$$q_\pi(s, a) = r(s, a) + \gamma \nu_\pi(\delta(s, a)) \ . \tag{3.8}$$

Aqui $\nu_\pi(\delta(s, a))$ é o retorno esperado do próximo estado $\delta(s, a)$, resultante da aplicação da ação a ao estado s.

Seguindo derivação de Busoniu et al. (2010), a equação (3.8) pode ser obtida primeiro escrevendo $Q^\pi(s, a)$ explicitamente como o retorno esperado por seguir uma política π. Este retorno é condicionado ao agente começar do estado s e tomar a ação a:

$$q_\pi(s, a) \coloneqq \mathbf{E}_\pi[G_t \mid S_t = s \, , A_t = a] = \mathbf{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid S_t = s \, , A_t = a\right] = \sum_{k=0}^{\infty} \gamma^k r(s_k, a_k) \ , \tag{3.9}$$

onde $(s_0, a_0) = (s, a)$, $s_{k+1} = \delta(s_k, a_k)$ para $k \geq 0$, e $a_k = \pi(s_k)$ para $k \geq 1$. Usamos também a definição da equação (3.5) na derivação acima. Por fim, separamos o primeiro

termo da soma de forma a obtermos uma equação com dois componentes

$$\begin{aligned} q_\pi(s,a) &= r(s,a) + \sum_{k=1}^{\infty} \gamma^k r(s_k, a_k) \\ &= r(s,a) + \gamma \sum_{k=1}^{\infty} \gamma^{k-1} r(s_k, \pi(s_k)) \\ &= r(s,a) + \gamma \nu_\pi(\delta(s,a)) \ . \end{aligned} \tag{3.10}$$

O primeiro componente é a recompensa imediata $r(s,a)$ resultante de tomar a ação a, estando no estado s. O segundo é o retorno esperado descontado por continuar escolhendo as próximas ações de acordo com a política π, partindo do próximo estado $\delta(s,a)$. Utilizamos acima a definição (3.6) de forma a obter (3.8).

Considerando que obtemos ν_* quando seguimos a política ótima, podemos escrever que $q_*(s,a) = r(s,a) + \gamma\nu_*(\delta(s,a))$. Esta relação implica que o agente pode obter a política ótima aprendendo ν_*, desde que tenha conhecimento perfeito sobre a função de recompensa imediata $r(s,a)$ e da transição de estado $\delta(s,a)$. Para contornar esta limitação, podemos escrever que a ação ótima no estado s é a ação $a = \pi(s)$ que maximiza a função (3.8) e satisfaz

$$\pi^*(s) = \arg\max_{a'} q_*(s,a') \quad , \ \forall s \in S \ . \tag{3.11}$$

Assim, considerando as definições (3.11) e (3.7), Busoniu et al. (2010) sugere que a *V-function* ótima que pode ser obtida por qualquer política é a mesma que que é calculada pela *Q-function* ótima, de maneira que

$$\nu^*(s) = \max_{\pi} \nu_\pi(s) = \max_{a'} q_*(s,a') \ ; \tag{3.12}$$

substituindo a definição acima em (3.8), obtemos a equação de otimalidade de Bellman, caracterizada por q_*. Ela define que o valor ótimo da ação a tomada no estado s é equivalente à soma das recompensas imediatas e o valor ótimo descontado obtido pela ação ótima no próximo estado:

$$q_*(s,a) \coloneqq r(s,a) + \gamma \max_{a'} q_*(\delta(s,a), a') \ . \tag{3.13}$$

Devido a presença do operador max, Waskow (2010) explica que a equação de otimalidade de Bellman resulta em um sistema não-linear de N equações para N estados.

As *Q-functions*, então, são caracterizadas recursivamente pela equação de Bellman, que tem importância central para os algoritmos de RL de iteração de valor e iteração de política. A equação de Bellman para q_π, por exemplo, define que o valor de tomar a ação a no estado s, seguindo a política π, equivale a soma das recompensas imediatas e o valor descontado obtido por π no próximo estado. Ela pode ser derivada a partir do segundo

passo na equação (3.10) da seguinte maneira:

$$\begin{aligned} q_\pi(s,a) &= r(s,a) + \gamma \sum_{k=1}^{\infty} \gamma^{k-1} r(s_k, \pi(s_k)) \\ &= r(s,a) + \gamma \left[r(\delta(s,a), \pi(\delta(s,a))) + \sum_{k=2}^{\infty} \gamma^{k-2} r(s_k, \pi(s_k)) \right] \\ &= r(s,a) + \gamma q_\pi(\delta(s,a), \pi(\delta(s,a))) \ , \end{aligned}$$

onde $(s_0, a_0) = (s, a)$, $s_{k+1} = \delta(s_k, a_k)$ para $k \geq 0$, e $a_k = \pi(s_k)$ para $k \geq 1$. Utilizamos também a definição (3.8) no último passo.

Como explicado por Mitchell (1997), a política ótima pode ser obtida mesmo se o agente usar apenas a ação atual a e estado s para escolher a ação que maximizará a função de ação-valor $q_\pi(s,\, a)$. Desta maneira, a função (3.13) implica que o agente pode selecionar a ação ótima mesmo quando não conhece as funções $r(s,a)$ e $\delta(s,a)$.

Spooner (2016) explica que, na maioria dos casos práticos de aplicação de RL, o conhecimento sobre o ambiente que o agente opera e os recursos computacionais necessários para resolver a equação de Bellman (3.8), limitam a aplicação direta do sistema. Cita a solução de jogos de xadrez como exemplo, que tem aproximadamente 10^{46} estados diferentes.

Assim, é necessário recorrer a métodos de aproximação, como os algoritmos que usam o método de diferença temporal, do inglês *temporal difference* (TD). A seguir, é explorado uma das mais importantes formulações ligadas ao *TD learning*.

3.2.2 *Q-Learning*

A natureza recursiva da função (3.13) implica que o agente não conhece a verdadeira função $q_*(s,a)$, podendo apenas estimá-la. Esta estimação, que representa a hipótese do agente sobre a verdadeira função $q_*(s,a)$, será referenciada como $\hat{Q}(s,a)$. Um dos primeiros avanços na literatura de aprendizado por reforço foi o desenvolvimento por Watkins (1989) de um método *TD* de controle *off-policy* conhecido como *Q-learning*. Nesta especificação, a *Q-function* estimada é atualizada de acordo com a seguinte regra:

$$\hat{Q}_{t+1}(s_t, a_t) \leftarrow \hat{Q}_t(s_t, a_t) + \alpha_t \left[r_{t+1} + \gamma \max_{a'} \hat{Q}_t(s_{t+1}, a') - \hat{Q}_t(s_t, a_t) \right] \ , \tag{3.14}$$

na qual $\alpha \in (0, 1]$ é a chamada taxa de aprendizagem e $a' \in A$ é a ação que maximiza o valor da hipótese atual $\hat{Q}$. O termo entre colchetes é o erro na estimação cometido no tempo t e é conhecido como erro *TD*.

A função (3.14) representa a hipótese sobre a função $q_*(s,a)$ como uma grande tabela que mapeia cada tupla $\langle s\, ,\, a \rangle$ para um valor $\hat{Q}$. Esta tabela, ilustrada pela tabela 1, pode ser inicializada com números aleatórios, como explicado por Mitchell (1997), ou com zeros. Os valores em negrito correspondem ao valor $\hat{Q}$ ótimo de cada estado.

As ações relacionadas a cada um destes valores correspondem a ação a' que maximiza $\max_{a'} \hat{Q}_t(s_{t+1}, a')$. Esta tabela permite que o agente, conhecendo o estado atual s_t, escolha a ação que poderá gerar o maior valor de recompensas acumuladas no futuro.

	Ação 1	Ação 2	Ação 3
Estado 1	**10**	-2	9
Estado 2	7	1	**15**
Estado 3	-7	**3**	2

Tabela 1 – Exemplo de tabela $\hat{Q}$

O *Q-learning*, assim como outros métodos *TD*, abordam o problema de estimação de $\hat{Q}$ atualizando a hipótese sobre ela a cada iteração que o agente tem com o ambiente. Depois de interagir em t, aguarda $t+1$ para então usar a recompensa observada r_{t+1}, e o valor da recompensa estimada pela maximização de $\hat{Q}(s_{t+1}, a)$, para atualizar a hipótese sobre o valor que seria obtido em t pela aplicação da função q_*.

Posto de outra forma, a função de ação-valor aprendida, $\hat{Q}$, aproxima diretamente q_*, a função ótima de ação-valor, independente da política que está sendo seguida. Por isso o uso do termo *off-policy*. O algoritmo 1 apresenta o modelo Q-*learning* utilizando exploração ϵ-*greedy*, que será explicada mais abaixo.

Algoritmo 1 Q-*learning* com exploração ϵ-*greedy*

Entrada: fator de desconto γ,
dinâmica de exploração $\{\epsilon_t\}_{t=0}^{\infty}$,
dinâmica da taxa de aprendizado $\{\alpha_t\}_{t=0}^{\infty}$

Observe o estado inicial $s_t = s_0$
para cada passo $t = 0, 1, 2, ...$ **faça**
  $a_t \leftarrow \begin{cases} a \in \arg\max_{a'} \hat{Q}(s_t, a') & \text{com probabilidade } 1 - \epsilon_t \text{ (\textit{exploite})} \\ \text{selecione } a \sim U; a \in A_t & \text{com probabilidade } \epsilon_t \text{ (\textit{explore})} \end{cases}$
  Aplique a_t e observe o próximo estado s_{t+1} e recompensa r_{t+1}
  se não existir a entrada $\langle s_t, a_t \rangle$ na tabela **então**
    $\hat{Q}_t(s_t, a_t) \leftarrow 0$
  $\hat{Q}_{t+1}(s_t, a_t) \leftarrow \hat{Q}_t(s_t, a_t) + \alpha_t \left[r_{t+1} + \gamma \max_{a'} \hat{Q}_t(s_{t+1}, a') - \hat{Q}_t(s_t, a_t) \right]$
  $s_t \leftarrow s_{t+1}$

Para Sutton e Barto (2017), uma das vantagens do *Q-Learning* e dos outros métodos *TD* é que esses não exigem um modelo para o MDP, diferente de outras classes de algoritmo de RL. Não precisamos assumir premissas sobre a distribuição de probabilidade das recompensas ou da transição de estados, por exemplo. Além disso, os algoritmos de *TD learning* convergem para a política ótima de maneira incremental, reduzindo seu custo computacional.

Watkins e Dayan (1992) mostram que $\hat{Q}$ converge para q_* assintoticamente a medida que o número de transições $t \to \infty$, quando o espaço de estados e ações for discreto

e finito. Além disso, alegam ser necessário que: a soma $\sum_{t=0}^{\infty} \alpha_t^2$ produza um valor finito, enquanto $\sum_{t=0}^{\infty} \alpha_t$ resulte em infinito; todos os pares estado-ação sejam (assintoticamente) visitados infinitamente.

Como explicado por Busoniu et al. (2010), a primeira condição é fácil de satisfazer. Uma escolha padrão seria definir $\alpha_t = \frac{1}{t}$, por exemplo. Na prática, a taxa de aprendizagem α pode exigir alguma otimização, uma vez que influencia diretamente na quantidade de transições necessárias para que o *Q-learning* obtenha uma solução satisfatória. A escolha dessa dinâmica depende muito do problema de aprendizagem em questão.

A segunda condição pode ser satisfeita se, entre outras coisas, o agente tiver uma probabilidade diferente de zero de selecionar qualquer ação a em todo estado s encontrado. Como explicado por Mitchell (1997), o que pode acontecer caso não for adotado esta abordagem probabilística, é que o agente pode se comprometer com ações que apresentaram valores positivos $\hat{Q}$ cedo na simulação, falhando em explorar outras ações que poderiam ter gerado valores mais altos (*exploration*).

Porém, o agente também precisa aproveitar o que já aprendeu (*exploitation*) para atingir um boa performance. Para isso, escolhe as ações que geraram maiores valores de $\hat{Q}$ segundo a experiência acumulada, chamadas de *greedy actions* na literatura. Esta dinâmica entre explorar novas ações e aproveitar o que já foi aprendido é chamado de *exploration-exploitation trade-off*.

Uma maneira clássica de balancear este *trade-off*, chamada de exploração *ϵ-greedy*, foi inclusa no passo 3 do algoritmo 1, onde $\epsilon_t \in (0, 1)$ é a probabilidade de exploração no passo t, que tende a zero a medida que $t \to \infty$. Busoniu et al. (2010) explicam que é necessário que taxa de exploração diminua com o tempo para que a política $\hat{Q}$ usada pelo agente vá se aproximando da política ótima q_*. É importante relembrar que o algoritmo descrito é adequado para MDPs com espaço de ação e estados discreto. Abaixo serão explorados os conceitos necessário para utilizar o *Q-learning* quando o espaço de estados é contínuo.

3.2.3 *Q-Learning* com generalização de função

Em muitas aplicações práticas de RL, a representação do espaço de estados no qual estamos interessados pode ser tão grande que torna improvável a solução do problema usando um método tabular, como a forma original do *Q-learning*. Por exemplo, isto acontece quando são utilizadas variáveis contínuas na representação. Nestes casos, Sutton e Barto (2017) argumentam que não pode-se esperar que o agente preencha cada combinação $\langle s, a \rangle$ possível da tabela de maneira satisfatória, principalmente se considerar as limitações computacionais e de quantidade de dados necessários para a estimação dos valores.

Cumming, Alrajeh e Dickens (2015) usam um algoritmo de aprendizado não

supervisionado chamado *k-means* para agrupar os estados encontrados, que possuem valores contínuos, usando estes *clusters* como seu novo espaço (discreto) S. Usam, assim, a versão tabular do *Q-learning*. Já Spooner (2016) trata este problema usando um método linear de generalização paramétrico para estimar os valores $\hat{Q}$ para estados ainda não visitados, chamado *tile coding*. Este método, que é uma instância de aprendizado supervisionado, faz parte de uma abordagem chamada *aproximação de função* na literatura de aprendizado por reforço, e seu uso exige que os métodos clássicos de RL sejam adaptados para garantir convergência.

Spooner (2016) explica que os métodos lineares de aproximação paramétrica assumem que $\hat{Q}$ pode ser expressado como uma combinação linear de n funções independentes $\phi_1, ..., \phi_n : S \times A \to \mathbb{R}$ (também chamadas de *basis functions* (BFs)), ponderadas por um vetor $\theta \in \mathbb{R}^n$ de parâmetros, de maneira que $\hat{Q}(s,a) = [F(\theta)](s,a)$. Busoniu et al. (2010) define a função de valor aproximada $[F(\theta)](s,a)$ como

$$[F(\theta)](s,a) = \sum_{i=1}^{n} \phi_i(s,a)\theta_i = \phi^\intercal(s,a)\theta \ , \tag{3.15}$$

onde $\phi(s,a) = [\phi_1(s,a), ..., \phi_n(s,a)]^\intercal$ é um vetor coluna de BFs estado-ação e o número de parâmetros n é estabelecido arbitrariamente. Cada $\phi_i(s,a)$ é uma função de ativação binária com valor 1 para os parâmetros ativados e 0, caso contrário.

Busoniu et al. (2010) propõe usar gradiente descendente para integrar o método linear de aproximação paramétrica ao *Q-learning*, assumindo que a aproximação da função $\hat{Q}(s_t, a_t) = [F(\theta_t)](s_t, a_t)$ seja diferenciável em θ. Vamos assumir que, depois de tomar a ação a_t no estado s_t, o algoritmo recebe o valor ótimo de Q no atual par $\langle s, a \rangle$, $q_*(s_t, a_t)$, além do próximo estado s_{t+1} e recompensa r_{t+1}. Sob estas circunstâncias, o algoritmo de gradiente descendente, que minimiza o erro quadrático entre o valor ótimo e o valor atual de Q, pode ser escrito como

$$\begin{aligned}
\theta_{t+1} &= \theta_t - \frac{1}{2}\alpha_t \frac{\partial}{\partial \theta_t}\left[q_*(s_t,a_t) - \hat{Q}_t(s_t,a_t)\right]^2 \\
&= \theta_t + \alpha_t \left[q_*(s_t,a_t) - \hat{Q}_t(s_t,a_t)\right] \frac{\partial}{\partial \theta_t}\hat{Q}_t(s_t,a_t) \\
&= \theta_t + \alpha_t \left[r_{t+1} + \gamma \max_{a'} \hat{Q}_t(s_{t+1},a') - \hat{Q}_t(s_t,a_t)\right] \frac{\partial}{\partial \theta_t}\hat{Q}_t(s_t,a_t) \\
&= \theta_t + \alpha_t \left[r_{t+1} + \gamma \max_{a'} \left(\phi^\intercal(s_{t+1},a')\theta_t\right) - \phi^\intercal(s_t,a_t)\theta_t\right] \phi(s_t,a_t) \ .
\end{aligned} \tag{3.16}$$

Usamos a função (3.13) para substituir q_* e a função (3.15) para substituir $\hat{Q}(s_t, a_t)$. Foi demonstrado por Melo, Meyn e Ribeiro (2008) que o *Q-learning*, usando aproximação linear, converge para a função de valor ótima. Abaixo apresentamos a versão do algoritmo 1 baseada em gradiente do *Q-learning* com aproximação linear e exploração ϵ-*greedy*.

Assim como Spooner (2016), esta dissertação utilizará um método de aproximação linear chamado *tile coding*. Waskow (2010) explica que este método representa as variáveis

Algoritmo 2 Q-*learning* com parametrização linear e exploração ϵ-*greedy*

Entrada: fator de desconto γ,
BFs $\phi_1, ..., \phi_n : S \times A \to \mathbb{R}$,
dinâmica de exploração $\{\epsilon_t\}_{t=0}^{\infty}$,
dinâmica da taxa de aprendizado $\{\alpha_t\}_{t=0}^{\infty}$

1: Inicialize vetor de pesos $\{\theta_i\}_{i=0}^{M} \leftarrow 0$
2: Observe o estado inicial $s_t = s_0$
3: **para** cada passo $t = 0, 1, 2, ...$ **faça**
4: $a_t \leftarrow \begin{cases} a \in \arg\max_{a'} \hat{Q}(s_t, a') & \text{com probabilidade } 1 - \epsilon_t \ (\textit{exploite}) \\ \text{selecione } a \sim U; a \in A_t & \text{com probabilidade } \epsilon_t \ (\textit{explore}) \end{cases}$
5: Aplique a_t e observe o próximo estado s_{t+1} e recompensa r_{t+1}
6: $\theta_{t+1} \leftarrow \theta_t + \alpha_t \left[r_{t+1} + \gamma \max_{a'} \left(\phi^{\intercal}(s_{t+1}, a')\theta_t\right) - \phi^{\intercal}(s_t, a_t)\theta_t \right] \phi(s_t, a_t)$
7: $s_t \leftarrow s_{t+1}$

do espaço de estados, chamadas de *features* na literatura de *machine learning*, em partições denominadas *tilings*. Cada partição é dividida em subpartições, chamadas *tiles*. A granularidade de cada *tiling* é controlada pela largura destes *tiles*, sendo que esta largura cobre um intervalo finito dos valores possíveis de cada *feature*. Sherstov e Stone (2005) sugerem o exemplo apresentado na figura 1. Nele, o espaço de estados consiste em uma única variável contínua x, t indica o número de *tilings* e w, a largura de cada *tile*. A razão w/t é chamada de resolução da estrutura de *tilings*.

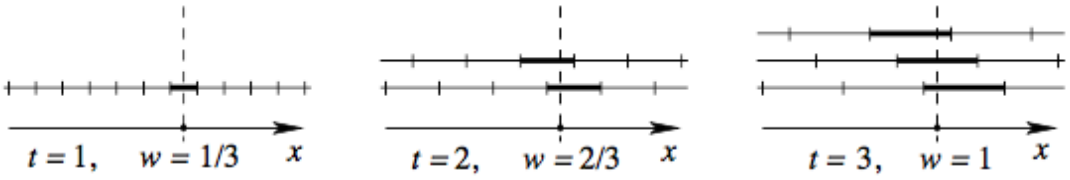

Figura 1 – Espaço de Estados divididos em um, dois e três *tilings* com mesma resolução $r = 1/3$ (SHERSTOV; STONE, 2005).

As áreas hachuradas na figura 1 indicam os *tiles* acionados pela variável x, sendo que a quantidade dos *tiles* ativados aumenta em função do número de *tilings* do modelo. O primeiro exemplo corresponde à discretização direta da variável x, que Sherstov e Stone (2005) alegam não generalizar bem nas bordas dos *tiles*. Generalizar, neste contexto, está relacionado a capacidade do modelo em classificar variáveis ainda não observadas como pertencendo a um tile ou outro. Assim, ainda que os três exemplos apresentados possuam resolução equivalente, a quantidade maior de *tilings* permite melhor generalização.

O $\hat{Q}$ *value* aproximado de cada tupla $\langle s, a \rangle$ é calculado pela da soma do valor de cada *tile* ativado, similar à função (3.15), com a diferença de que chamamos o vetor de parâmetros θ de vetor de pesos e o vetor ϕ de vetor de ativação. Da mesma forma, quando os valores dos *tilings* são atualizados, atualiza-se apenas os valores dos *tiles* ativados, resultando em um ganho computacional considerável.

Nesta dissertação, será utilizada uma implementação *open source* deste método em *Python*, fornecida por Sutton[1]. Sutton e Barto (2017) explicam que, para o método *tile coding* funcionar como esperado, é necessário conhecer a amplitude das *features* antes de inicializar a função e utilizar uma quantidade de *tilings* maior ou igual a quatro vezes o número de variáveis contínuas na representação de estado, de preferência uma potência de 2, sendo o expoente um número ímpar.

[1] Veja documentação em <http://incompleteideas.net/sutton/tiles/tiles3.html>

4 Metodologia

Abaixo serão apresentadas todas as etapas necessárias para a implementação do modelo proposto no capítulo anterior. Inicialmente, serão descritos o ambiente com o qual o agente irá interagir e os passos para sua construção. Depois, o problema de criação de um modelo de *trading* é abordado sob o *framework* de aprendizagem por reforço, especificando diferentes funções de recompensa, representações de estado e espaço de ações. Por último, é detalhado como o agente será avaliado e o *benchmark* utilizado, além do processo de otimização dos parâmetros do modelo.

4.1 Simulação de Mercado

O modelo proposto no capítulo 3 será aplicado ao mercado futuro de DI de um dia, negociado na BM&FBovespa. Lueska (2016) explica que o DI Futuro, um dos instrumentos mais líquidos do mercado brasileiro, é um derivativo cujo alvo de negociação é a taxa média dos depósitos interbancários entre a data de negociação e a data expiração deste contrato, também chamada de taxa *forward* média.

Como este trabalho tem como objetivo desenvolver uma estratégia de *trading* em juros futuro, utilizando o *framework* de aprendizado por reforço, o livro de ofertas será modelado como um processo de decisão de Markov. Este MDP será utilizado para desenvolver um agente que não terá nenhum conhecimento prévio sobre o instrumento operado, precisando aprender como atuar interagindo sucessivamente com o mercado, por tentativa e erro. Para tanto, um simulador de *order book* será construído, em *Python*, com o qual o agente poderá interagir para adquirir a experiência necessária. Diferentemente da maioria dos trabalhos mencionados no capítulo 2, nos quais o mercado é construído com dados artificialmente gerados, no trabalho da presente dissertação, o *book* será reconstruído utilizando dados históricos de alta frequência.

Abaixo será explicado brevemente como funciona o simulador construído e suas limitações, além de alguns dos conceitos de microestrutura para entender seu funcionamento. Depois, os dados utilizados são apresentados e os passos necessários para prepará-los para o simulador são discutidos. Em seguida, as premissas adotadas nos testes são detalhadas.

4.1.1 Reconstrução do *Book* de Ofertas

Nevmyvaka, Feng e Kearns (2006) alegam que, atualmente, a maioria dos mercado financeiros ao redor do mundo utilizam sistemas com *books* de ofertas limite (*limit order books* (LOB)) para viabilizar a negociação de diversos instrumentos. Nestes sistemas,

seguindo notação de Gould et al. (2013), em um determinado tempo t_x, um comprador (vendedor) pode submeter uma ordem $x = \langle p_x, \omega_x, t_x \rangle$ (também chamada de oferta), onde se compromete a comprar (vender) até ω_x unidades de um determinado instrumento com preço não superior (inferior) a p_x.

Quando uma oferta de compra (venda) x é submetida ao mercado, o *order matching engine* do LOB checa se é possível combinar x com alguma oferta de venda (compra) previamente submetida. Se for possível, um negócio (*trade*) é realizado imediatamente, resultando em ofertas executadas e/ou parcialmente executadas. Caso contrário (ou se for parcialmente executada), a oferta x fica ativa no *book*, e continua assim até ser combinada com alguma outra oferta de venda entrante (compra) ou ser cancelada. Estes cancelamentos ocorrem quando o dono da oferta não estiver mais disposto a realizar um negócio no preço apregoado ou por alguma regra da bolsa para cancelamento de ordens ativas. Por exemplo, no segmento BM&F, todas as ofertas ativas são canceladas no final do pregão (sessão). Neste caso, chamamos o cancelamento de oferta expirada. Uma oferta limite que é submetida de maneira que já resulte em um negócio, é chamada de ordem a mercado.

São as ofertas ativas que constituem um LOB. Assim, Gould et al. (2013) define que um LOB $\mathcal{L}(t)$ é o conjunto de todas as ordens ativas no mercado no tempo t e pode ser particionado em dois subconjuntos, um das ofertas ativas de compra (*bid side*) $\mathcal{B}(t)$ e outro, das ofertas ativas de venda (*ask side*) $\mathcal{A}(t)$. Cada um destes subconjuntos é organizado por ordem de prioridade preço-hora. Ou seja, quando chega uma oferta venda (compra) a mercado, a prioridade na execução para as ordens ativas de compra (venda) é dada primeiro para os preços p mais altos (baixos) e depois, considerando ofertas em um mesmo nível preço, para aquelas com a hora t de registro no sistema mais cedo.

O melhor preço de compra $p^b(t)$ no tempo t é definido como sendo o preço de compra mais perto do *ask side*. Podemos definir o melhor preço de venda $p^a(t)$ de maneira similar, sendo que $p^b(t) < p^a(t), \forall\, t$. Assim, temos que,

$$\begin{aligned} p^b(t) &\coloneqq \max_{o \in \mathcal{L}_B(t)} p_o \ , \\ p^a(t) &\coloneqq \min_{o \in \mathcal{L}_A(t)} p_o \ . \end{aligned} \tag{4.1}$$

Definimos também que o *bid-ask spread* no tempo t é dado por $\Delta(t) \coloneqq p^a(t) - p^b(t)$ e que o *mid-price* é definido como $p^m(t) \coloneqq \left[p^a(t) + p^b(t)\right] / 2$. A profundidade na compra (venda) disponível, ou volume, em um dado preço p e tempo t, pode ser escrito como

$$\begin{aligned} n^b(p,t) &\coloneqq \sum_{o \in \mathcal{B}(t) | p_o = p} \omega_o \ , \\ n^a(p,t) &\coloneqq \sum_{o \in \mathcal{A}(t) | p_o = p} \omega_o \ . \end{aligned} \tag{4.2}$$

Utilizando as equações (4.2), podemos agrupar os subconjuntos das ordens ativas $\mathcal{B}(t)$ e $\mathcal{A}(t)$ em novos subconjuntos de pares preço-volume $\langle p, n^b(p,t) \rangle$ e $\langle p, n^a(p,t) \rangle$,

ordenados por prioridade de preço. Estes novos livros são ditos agrupados (por preço). No mercado brasileiro, geralmente se visualiza $\mathcal{B}(t)$ e $\mathcal{A}(t)$ desta maneira, agrupados e organizados por sua ordem de preferência, lado a lado, sendo que o conjunto de ofertas ativas de compra é posta do lado esquerdo e o de venda, do lado direto. Na tabela 2 exibimos um exemplo de tal representação, mostrando apenas os 5 melhores preços para de cada lado. A coluna *qBid* se refere aos valores $n^b(p,t)$, sendo que p é cada preço exibido na coluna *Bid*. O mesmo raciocínio é válido para as colunas *qAsk* e *Ask*.

	qBid	Bid	Ask	qAsk
0	2.170	10.08	10.09	255
1	4.385	10.07	10.10	2.120
2	2.775	10.06	10.11	2.020
3	2.320	10.05	10.12	1.220
4	945	10.04	10.13	3.085

Tabela 2 – LOB agrupado. DI1F21, 2017-02-24 12:04:53.934

Dada estrutura do *book* de ofertas $\mathcal{L}(t)$ apresentada acima, a ordem de compra $x = \langle p_x, \omega_x, t_x \rangle$, que chega imediatamente depois do tempo t, obedece as seguintes regras:

- Se torna ativa, sendo inserida no nível de preço p_x, caso $p_x \leq p^b(t)$ (ou $p_x \geq p^a(t)$, para venda). Ela não altera p^b ou p^a. Se $p_x \in \mathcal{B}(t)$, então x tem a menor prioridade no nível de preço $n^b(p_x, t_x)$, já que $t_x > t$. O mesmo vale para venda.

- Se torna ativa, ocupando o novo nível de preço p_x, caso $p^b(t) < p_x < p^a(t)$. Isso aumenta o melhor preço $p^b(t)$, caso for uma compra, ou diminui melhor preço $p^a(t)$, caso x for uma venda. Sua ordem de prioridade será a maior para aquele nível de preço $n^b(p_x, t_x)$ e permanecerá assim enquanto a quantidade w_x for menor ou igual a inserida inicialmente. O mesmo vale para venda.

- Se $p_x \geq p^a(t)$ (ou $p_x \leq p^b(t)$ para venda), então x é uma ordem a mercado que é imediatamente combinada com uma ou mais ordens ativas em $\mathcal{A}(t)$ (ou $\mathcal{B}(t)$, caso venda). Quando isso ocorre, a ordem entrante x é combinada com a ordem ativa $y = \langle p_y, \omega_y, t_y \rangle$ com maior prioridade do lado oposto, onde $t_y < t_x$. Se $\omega_x > \omega_y$, então a quantidade residual de x é combinada com a próxima oferta ativa do lado oposto com maior prioridade. Este processo continua até x ser completamente executada ou consumir todas as ofertas ativas até o limite p_x, quando o resíduo de x se torna ativo em $\mathcal{B}(t)$. Todos os negócios realizados para preencher x são fechados no preço da ordem ativa consumida (independentemente do valor p_x). Os novos preços $p^b(t)$ e $p^a(t)$ são atualizados de acordo com as equações 4.1. O mesmo raciocínio é válido para venda.

Uma parcela da presente dissertação ocupou-se em implementar uma biblioteca para reproduzir a evolução de um LOB durante todo um pregão, obedecendo as características definidas acima. O agente será capaz de interagir com o *order book*, competindo por preferência ao inserir novas ofertas, além de poder cancelar as que submeteu previamente e poder executar *trades* e ter suas ofertas executadas. Também foi criado um mecanismo de *order matching* para administrar toda esta dinâmica, mesclando as ordens do agente com o fluxo original de ofertas. Assim, é definida uma prioridade adequada para cada ordem de acordo com o horário de chegada e seu preço limite. Para qualquer oferta que tenha sua quantidade aumentada, ou seu preço é alterado, é estabelecida uma nova ordem prioridade para ela.

Dentro do *framework* criado, a classe *limit order book* representa os conjuntos $\mathcal{B}(t)$ e $\mathcal{A}(t)$, já agrupados de acordo com as equações (4.2), como duas árvores binárias, que é um tipo de estrutura de dados conhecida pela eficiência na ordenação de seus nós. Os nós destas árvores estão ordenados por nível de preço, como na tabela 2. Em cada nível de preço, foi criada uma nova árvore binária composta pelas ordens submetidas pelos agentes, que estão ordenadas por hora de chegada. Utilizou-se a implementação da árvore binomial disponível na biblioteca *Bintrees*[1], que é implementada em C e portada para *Python*.

Spooner (2016) sugere que simulações de *book* de ofertas baseadas em modelos, como feito por Chan e Shelton (2001) ou Kim, Shelton e Poggio (2002), por exemplo, podem ter resultados pouco realistas, mesmo quando calibrados com dados históricos. Já reproduzindo o *book* com dados de alta frequência, conseguimos replicar o comportamento dos diferentes *players* do mercado com precisão satisfatória.

Uma limitação importante desta abordagem é que as ações do agente não interferem na dinâmica do ambiente. Por exemplo, caso o agente inserisse uma oferta com volume relevante para aquele ativo, provavelmente observaria uma mudança no comportamento do mercado em resposta à sua oferta. Este comportamento não será refletido no simulador aqui implementado. Para tentar contornar esta limitação, os agentes criados poderão apregoar apenas com o menor lote possível para o ativo. Considerando a liquidez dos instrumentos que serão testados, ofertas deste tamanho provavelmente não causariam nenhuma fricção no mercado no mundo real.

4.1.2 Dados e pré-processamento

Foram utilizados dados *tick-by-tick* neste trabalho, os quais se referem a todos os eventos que aconteceram em todos os *books* de ofertas de todos os instrumentos negociados no segmento de BMF em um determinado pregão, disponíveis no FTP[2] da bolsa. Foram utilizados dados do dia primeiro até o dia 24 de fevereiro de 2017, totalizando 18 sessões

[1] Documentação disponível em <https://pypi.python.org/pypi/bintrees/2.0.2>
[2] Dados e *layouts* disponíveis em <ftp://ftp.bmf.com.br/MarketData/>

diferentes e cerca de 47 GB de arquivo texto no total. Estes eventos são compostos por ofertas limite, ofertas a mercado, ordens *stop*, cancelamentos, ofertas executadas, parcialmente executadas e ofertas expiradas.

Os dados estão divididos em três arquivos. Um apenas com os eventos no lado da compra, outro com os eventos do lado da venda e um terceiro com todos os negócios realizados no dia. Este último arquivo não será utilizado, pois os negócios que aconteceram em cada pregão também aparecem nos arquivos de compra e venda, como ofertas executadas ou parcialmente executadas. Porém, o arquivo *trades* poderia ser utilizado, por exemplo, para identificar ofertas *cross market*, chamadas de diretos no Brasil, pois não aparecem nas outras bases. Este tipo de oferta não é relevante para o agente de aprendizagem, uma vez que são eventos que não passam pelo *book* de ofertas.

Os arquivos disponíveis, em estado bruto, não estão prontos para o simulador. Os dados não estão ordenados na maneira necessária e existem várias ofertas que não devem ser consideradas quando os LOBs são reconstruídos. Assim, é preciso pré-processar estes dados antes de utilizá-los. Eles estão organizados primeiro pelo nome dos instrumentos e depois pelo número de identificação de cada ordem. Na tabela 3 há uma comparação entre o dado original e o mesmo interpretado para uma linha do arquivo de eventos do lado da venda do DI1F18 (janeiro 2018).

	original	interpretado
Data da Sessão	2017-02-01	2017-02-01
Símbolo do Instrumento	DI1F18 [...]	DI1F18
Sentido da Oferta	2	Sell Order
Número de Sequência	000076421086361	000076421086361
Número Secundário	000000002101297	000000002101297
Cód do Evento da Oferta	002	Update
Hora Prioridade	09:27:13.530000	34033.53
Ind de Prioridade da Oferta	0018990608	18990608
Preço da Oferta	000000000010.910000	10.91
Qtd.Total da Oferta	0000000000000000025	25
Qtd.Executada da Oferta	0000000000000000000	0
Data da Oferta	2017-02-01	2017-02-01
Data de Entrada da Oferta	2017-02-01 09:27:13	2017-02-01 09:27:13
Estado da Oferta	5	Replaced
Indicador de agressão	0	Neutral
Corretora	00000127	127

Tabela 3 – Estrutura interpretada do arquivo original

Entre as transformações realizadas na tabela 3, alguns dados foram apenas convertidos de texto para valor, como no preço da oferta, outros foram substituídos para tornar a interpretação mais direta, como no estado da oferta, e, por fim, a hora de prioridade foi convertida para o número de segundos equivalente para facilitar a ordenação dos dados.

Como serão realizados testes com 3 vencimentos diferentes (o janeiro 2019, 2021 e 2023), os dados relacionados a cada instrumento foram separados em arquivos diferentes, diminuindo sensivelmente o volume de dados iterados em cada simulação. As linhas dentro destes arquivos foram reordenadas pelo horário do registro (Hora Prioridade) de cada evento no sistema da bolsa, que tem precisão de milissegundos, e depois pelo seu número de sequência, valor que é único e incremental para um dado instrumento e pregão.

Com estes arquivos reordenados, a dinâmica do livro de ofertas de cada instrumento foi recriada uma primeira vez. Os dados, neste ponto, ainda apresentam várias inconsistências. Observa-se que, em diferentes momentos, aparecem novas ofertas limite de compra $o = \langle p_o, w_o, t_o \rangle$ com o preço maior que o menor preço de venda ($p_o > p^a(t)$), evento este que não aconteceria sem resultar em um negócio, como definido na última seção.

Acompanhando estas ofertas, é possível verificar que muitas delas são modificadas durante o pregão, ainda para preços inconsistentes com o *book* e, posteriormente, canceladas, não sendo convertidas em *trades* em nenhum momento. Como ofertas do tipo *stop*[3] não estão identificadas nas bases de dados, ainda que sejam mencionadas no arquivo de *layout*, concluímos que estas ofertas, que cruzam o *book* sem resultar em negócio, provavelmente são ofertas *stop*. Então, para identificar tais ofertas, as seguintes checagens foram adotadas:

1. Quando acontece um negócio em um dos lados do *book*, e este *trade* é classificado no arquivo como *passive* (a oferta foi agredida por outro participante), identificamos todas as ofertas com melhor posição do que a agredida e marcamos seus códigos de identificação em uma lista. Naquele momento, sabemos qual a melhor oferta daquele lado do *book*. Logo, outras ofertas, com colocação supostamente melhor na fila, deveriam ter preferência na execução se realmente fossem ofertas limite.

2. Quando o negócio que aconteceu é identificado como *agressor* (o participante daquele lado do *book* agrediu o outro lado), não sabemos qual a melhor oferta, mas sabemos qual é o preço limite que as ofertas do lado agressor podem ter. Neste caso, identificamos todas as ofertas com preços melhores que este preço (maior se for compra e menor se for venda), e marcamos seus códigos de identificação em outra lista.

3. Quando estamos observando um LOB "consistente"(preço de compra menor que preço de venda) e aparece uma oferta nova que cruza os preços sem gerar negócio, marcamos seu código de identificação em uma terceira lista. Só voltamos a analisar se houve cruzamento novamente quando voltarmos a verificar que o *book* está consistente.

Depois desta iteração, o arquivo inicial é reprocessado, excluindo todas as mensagens com identificação presente em alguma das listas criadas. Elas não são excluídas apenas

[3] Ordem *stop*: oferta com preço limite definido, porém que só se torna ativa depois que algum negócio é realizado acima (quando é uma venda) ou abaixo (quando é uma compra) de um valor pré-estabelecido, chamado gatilho.

quando o evento é um *trade*. Este evento, que provavelmente se refere ao disparo de um *stop*, é pouco frequente. Devido à etapa 3, este processo precisa ser executado repetidamente para se obter um arquivo adequado. Um arquivo é chamado de "adequado" quando se efetua uma simulação, com um agente coletando os *books* a cada 3 milissegundos, e esses estão todos coerentes. Ou seja, seus preços não cruzam e o *bid-ask spread* ficou dentro de um *range* esperado.

Os dados finais, utilizados neste trabalho, foram obtidos depois de realizar o procedimento acima 3 ou 4 vezes para cada pregão e instrumento usado. Este processo excluiu cerca de 7% do total de eventos registrados nos arquivos originais, restando 9.442.335 linhas de dados para serem utilizadas, o equivalente à $2,3$ GB de arquivo texto. Para evitar horários que podem acontecer leilões, o agente implementado também só poderá operar entre às 9:20:00 até às 15:40:00 (o mercado regular de DI fecha às 16 horas).

4.1.3 Latência e custos

Segundo Xavier (2015), hoje em dia, ofertas executadas por estratégias de alta frequência representam quase metade de todo volume operado em mercados desenvolvidos. No Brasil, a representatividade ainda não é tão grande - cerca de 10% - mas é esperado que esta fatia aumente nos próximos anos. Neste contexto de alta frequência, O'Hara (2015) argumenta que um *player* informado é aquele que processa e reage a eventos de mercado antes de seus competidores. Ser desinformado é ser mais lento que os outros.

Considerando que não se pode ignorar a latência das ofertas do agente, pois isso daria uma vantagem indevida a ele, será adotado um intervalo de 10 milissegundos entre cada processamento de *book* pelo agente (quando poderá colocar, cancelar e modificar duas ofertas), e de 5 milissegundos para o agente reagir à uma execução. É interessante ressaltar que estes tempos são bastante conservadores, dado que estratégias de *algo trading* muitas vezes utilizam *co-location* (o sistema da estratégia é colocado fisicamente próximo do núcleo de negociação da bolsa para diminuir a latência). A latência para recolocar as ofertas e para reagir a uma execução pode ser acrescida ou diminuída em 2 e 1 milissegundos, respectivamente, com probabilidade de 40% para ocorrer a alteração ou não, e 50% para definir o sinal da alteração, quando ocorrer.

Adicionalmente, será considerado um custo de R$ 0,80 centavos por contrato operado, que é o custo médio (considerando emolumentos e corretagem) que um fundo de investimento pequeno teria para operar os instrumentos de interesse deste trabalho.

4.2 Modelo de *trading* descrito como um problema de RL

Abaixo será descrito como a criação de um modelo de *trading* foi adaptada ao *framework* de aprendizado por reforço. Neste contexto, não há uma estratégia pré-definida.

A forma de negociar aprendida pelo modelo apresentado no capítulo 3 constitui a estratégia pesquisada neste trabalho. Inicialmente são descritas quais ações podem ser tomadas pelo agente a cada passo de tempo. Depois, define-se três noções de recompensa diferentes para serem testadas e, por fim, são descritas as variáveis que o agente poderá observar em seu processo de aprendizagem.

4.2.1 Espaço de Ações

Nesta dissertação, adotou-se que cada episódio de simulação abrange todo um pregão. A cada passo de tempo t durante este episódio, o agente deverá escolher entre manter ou não ofertas nos melhores preços de mercado. Desta maneira, no passo de tempo t, o agente deve selecionar uma ação a_t de um conjunto de ações possíveis A, que consistem em quatro ações diferentes:

$$a_t \in (None\,, best\_bid\,, best\_ask\,, best\_both) \ ,$$

na qual $None$ indica que o agente não possui ofertas nos melhores preços, $best\_bid$ e $best\_ask$ indicam que o agente possui oferta apenas no preco $p^b(t)$ ou apenas no preço $p^a(t)$, respectivamente. $best\_both$ indica que possui ofertas de ambos os lados.

Inicialmente, esta dissertação adotou que quando o agente atingisse seu limite de posição Π_{max} dentro de um episódio, as ações disponíveis para serem selecionadas seriam alteradas de acordo. Por exemplo, quando o agente batesse seu limite de compra, poderia escolher apenas entre $a_t \in (None\,, best\_ask)$. Porém, durante as simulações realizadas, verificou-se que o agente ficava enviesado para o lado que o mercado se movia em dias muito agitados. A solução adotada foi permitir que o agente sempre escolha qualquer ação em A, porém, seu preço de apregoação quando enviar uma oferta limite $o = \langle p_o, w_o, t_o \rangle$ passou a ser dado por:

$$p_o = p^x(t) \pm \begin{cases} 0.01 & \text{se } |\Pi| \geq \Pi_{max} \text{ ou } n^x(p^x, t_o) \leq 10 \\ 0.0 & \text{caso contrário} \end{cases} \ ,$$

onde Π é o portfólio do agente, $x \in (a, b)$ é o lado de apregoação da oferta e o sinal da função acima depende deste lado, somando quando $x = a$, e subtraindo caso contrário. Este *spread* adicionado quando atinge seu limite de posição ou quando o volume no melhor preço do lado apregoado é baixo, na grande maioria das vezes é suficiente para evitar que a oferta deste agente seja executada.

O agente também poderá manter mais de uma oferta ativa em cada lado do LOB, até o máximo de seu limite de posição. Por exemplo, se estiver vendido em 5 contratos, e seu limite for de 15 contratos, poderá manter mais 2 ofertas de 5 contratos em $\mathcal{A}(t)$. Além disso, foi permitido que o agente agredisse mercado apenas no fechamento deste, para zerar posição, ou se atingisse algum limite pré-estabelecido de ganho ou de perda.

Ainda que tenham sido implementados limites fixos de 3 *basis points* para ambos os casos, observou-se que o agente geralmente zerou suas posições sem lançar mão deste artifício.

Outro desafio encontrado durante a implementação deste modelo foi entender o que o agente estava conseguindo aprender dependendo da frequência com que mudava de ideia. Em análise preliminar, quando se permitiu que o agente mudasse seu posicionamento em relação ao mercado a cada 10 milissegundos, o agente não conseguiu aprender nada de fato, mesmo aumentando a quantidade de iterações na mesma base. Aparentemente, o agente estava mudando de ideia muito antes de poder observar as consequências de sua decisão anterior.

Assim, ainda que o agente continue mudando suas ofertas de lugar a cada 10 milissegundos, adotou-se uma janela de 30 segundos onde o agente deve manter a mesma decisão tomada no início do intervalo. Assim, caso a ação tomada no início do intervalo seja *best_bid*, a cada 10 milissegundos o agente poderá reposicionar suas ofertas de modo que fique com ordem apenas no melhor preço de compra. Porém, independente do que acontecer no mercado, deverá manter este posicionamento (de manter oferta no melhor preço de compra) até o final dos 30 segundos ou até ser executado. Caso for executado, só poderá inserir uma oferta no lado executado depois de passados estes 30 segundos.

Restrições como estas aqui apresentadas, como o *stop* compulsório, o tempo para o agente mudar de ideia, o *spread* adicionado quando atinge limite de risco, foram escolhas feitas ao longo dos testes realizados. Como todas estas características funcionam como regras que o agente precisa seguir, é dito que elas fazem parte do ambiente. Outro componente de um problema de RL que faz parte do ambiente é a noção de *reward*, que será explorada em seguida.

4.2.2 Funções de recompensa testadas

No aprendizado por reforço, o objetivo central do agente é maximizar alguma noção de recompensa previamente estabelecida. Logo, a definição da função que representa tal noção é peça chave para aquilo que o agente será capaz de aprender.

Ao caracterizar um modelo de *trading* como um problema de RL, pode-se tentar perseguir múltiplos objetivos ao mesmo tempo. Naturalmente, é desejável que a estratégia seja lucrativa. Pode ser interessante também que este resultado seja obtido com pouca variabilidade, por exemplo. Contudo, como esta dissertação propõe a criação de um agente essencialmente passivo (não enviará ordens a mercado por conta própria), se apenas for utilizado o *Profit and Loss* (P&L) para recompensá-lo, o agente só receberá incentivos quando estiver posicionado. Logo, também é desejável que ele seja recompensado de alguma maneira por esta atuação passiva.

Na literatura, a prática padrão quando se têm múltiplos objetivos é usar uma

função de *reward* que seja uma soma ponderada dos termos que representam cada uma das fontes de recompensa. No entanto, muitas vezes é difícil equilibrar tais fatores de maneira que permitam alcançar um desempenho satisfatório em todas as recompensas. Sutton e Barto (2017) alertam, ainda, que é preciso certo cuidado ao definir funções desta forma, uma vez que o agente pode se concentrar em atingir os sub-objetivos estabelecidos e não o objetivo principal.

Assim, serão testadas três funções de *reward* distintas. A primeira motivará o agente simplesmente pelo ganho financeiro de suas operações. Tal ganho é definido como

$$PnL \coloneqq PU(\bar{b}, du) \cdot qtde_b - PU(\bar{a}, du) \cdot qtde_a - \Pi \cdot PU(p^m(t), du) - C \ ,$$

onde $\bar{b}$ é o preço médio das compras efetuadas pelo agente até o tempo t e $\bar{a}$ é o preço médio das vendas. *du* são os dias de saque bancário até o vencimento do contrato operado e $PU(p, du) \coloneqq 10^5/(1+p)^{(du/252)}$. Note que a marcação a mercado está sendo realizada pelo *mid-price* $p^m(t)$. $qtde_a$ e $qtde_b$ são os totais executados pelo agente no episódio, do lado da compra e da venda, respectivamente. Todos os valores relacionados com quantidade de contratos se referem à posição em taxa e por isso estão invertidos no cálculo de PnL. C é o custo em taxas e corretagem incorrido pelas operações do agente.

A primeira função de *reward* testada, portanto, é definida como a diferença do ganho do agente entre o tempo t e $t+1$, de modo que:

$$r_{t+1} = PnL_{t+1} - PnL_t \ . \tag{4.3}$$

A segunda função de recompensa testada premiará ou punirá o agente somente com base em quão bem ele posicionou suas ofertas no *book*. Isso será quantificado pela quantidade de contratos que entraram ou saíram das primeiras filas do LOB $\mathcal{L}(t)$ entre os momentos que o agente tomou alguma decisão.

Esta contabilização de lotes será baseada no *order flow imbalance* (OFI) do instrumento operado, variável esta que representa o *net* de quantidade que entrou e saiu das melhores filas do *bid* e do *ask*. Cont, Kukanov e Stoikov (2014) argumentam que os eventos do *book* - ordens a mercado, ordens limite e cancelamentos - impactam na dinâmica do preço do ativo e que este impacto pode ser modelado através desta variável. Assim, para cada evento e_t em $\mathcal{L}(t)$:

$$e_t \coloneqq \mathbf{I}_{p_t^b \geq p_{t-1}^b} n_t^b - \mathbf{I}_{p_t^b \leq p_{t-1}^b} n_{t-1}^b - \mathbf{I}_{p_t^a \leq p_{t-1}^a} n_t^a + \mathbf{I}_{p_t^a \geq p_{t-1}^a} n_{t-1}^a \ .$$

Por conveniência, definimos que $n_t^b = n^b(p^b(t), t)$ e $n_t^a = n^a(p^a(t), t)$ se referem a quantidade no melhor preço no $\mathcal{B}(t)$ e no $\mathcal{A}(t)$ no período atual t. O subscrito $t-1$ se refere ao estado anterior. $\mathbf{I}$ é uma função indicadora. Como os eventos que afetam o book acontecem em tempos τ_n aleatórios, definimos $N(t) = \max\{n \mid \tau_n \leq t\}$ o número de

eventos que ocorreram entre $[0,\, t]$. A variável OFI é definida sobre o intervalo $[t_{k-1},\, t_k]$ e é a soma de cada e_t que ocorreu neste período, de maneira que:

$$OFI_k = \sum_{n=N(t_{k-1})+1}^{N(t_k)} e_n \ .$$

Os autores alegam que esta variável consegue explicar mudanças no *mid-price* em curtos espaços de tempo utilizando uma relação linear simples. A intuição, segundo eles, é que é preciso volume para movimentar o preço.

Uma primeira etapa na construção da segunda função de recompensa a ser testada é aplicar uma transformação $f()$ à variável OFI_k de modo que

$$ofi = f(OFI_k) \in [0, 1] \, .$$

A transformação $f()$ depende da probabilidade empírica da variável, calibrada nos últimos três pregões de janeiro de 2017. O período k utilizado neste trabalho foi de 10 segundos. Esta transformação é utilizada para quantificar quão intenso foi o nível de atividade em $\mathcal{L}(t)$ durante o período k.

A segunda função recompensa será, então, dividida em quatro partes que premiam o agente com o valor ofi dependendo de sua última ação escolhida a_t e portfólio Π_t. A primeira parte desta função premia o agente por escolher um lado do livro de ofertas para apregoar. A segunda parte avalia os casos em que o agente possui ofertas no melhor preço de ambos os lados do *book*. A terceira e a quarta parte avaliam quando o agente **não** possui ofertas nos preços p^a e p^b e tratam os casos quando o agente tem ou não posição aberta, respectivamente.

Desta forma, definindo a primeira condição

$$c_1 \leftarrow a_t \in \{best\_bid,\, best\_ask\}$$

teremos a primeira parte da função de recompensa baseada em OFI dada por:

$$r_{t+1}^{(1)} = \mathbf{I}_{c_1} \times \begin{cases} ofi \times \ \ 1.05 & \text{se } a_t = best\_bid \\ ofi \times -1.05 & \text{se } a_t = best\_ask \end{cases} \ .$$

As constantes acima têm como objetivo dar um peso maior para quando o agente corretamente apregoar do lado que efetivamente estiver entrando mais ofertas. Em seguida, definindo

$$c_2 \leftarrow a_t = best\_both$$

teremos a segunda parte da função recompensa dada por:

$$r_{t+1}^{(2)} = \mathbf{I}_{c_2} \times \begin{cases} f_2\left(ofi\right)/2 & \text{se } |ofi| \leq 0.2 \\ f_2\left(ofi\right)/2 \times -1 & \text{se } 0.2 < |ofi| \leq 0.5 \\ \left(f_2\left(ofi\right)/2 + 0.3\right) \times -1 & \text{se } |ofi| > 0.5 \end{cases} \ ,$$

na qual

$$f_2\left(ofi\right) = \frac{|ofi| - \omega_1}{\omega_2 - \omega_1} \text{ com } \omega_1 < ofi \leq \omega_2 \ .$$

As variáveis ω_1 e ω_2 são os limites das condições. Por exemplo, na segunda linha, $\omega_1 = 0.2$ e $\omega_2 = 0.5$. Esta função foi definida para que a amplitude da recompensa seja igual em todos os casos.

O objetivo na primeira parte da função $r^{(2)}$ é que o agente ganhe uma recompensa por manter ofertas em $\mathcal{B}(t)$ e $\mathcal{A}(t)$ quando houver pouca atividade no *book*. Na segunda e na terceira linha, é dada uma punição por não escolher um lado específico do livro de ofertas para apregoar, sendo que, na terceira condição, a punição é mais severa. A função deste último caso é incentivar o agente, quando o *book* estiver muito ativo, a escolher um lado para apregoar ou não fazer nada.

Para a terceira parte da função de recompensa, definindo a condição

$$c_3 \leftarrow (\,(a_t = None) \ \wedge \ (\Pi = 0))$$

teremos o primeiro termo que trata situações em que o agente não possui ofertas nos melhores preços dado por:

$$r_{t+1}^{(3)} = \mathbf{I}_{c_3} \times \begin{cases} f_3\left(ofi\right) \times -1 & \text{se } |ofi| \leq 0.20 \\ f_3\left(ofi\right) & \text{se Se } |ofi| \geq 0.80 \\ 0 & \text{caso contrário} \end{cases} ,$$

na qual

$$f_3\left(ofi\right) = \frac{f_2\left(ofi\right)}{\exp(visitas_t/200)} \ .$$

Nesta última $visitas_t$ é a quantidade de vezes que o agente visitou aquela condição específica até aquele ponto do episódio. Esta transformação foi definida de modo que recompensas e penalidades provenientes destas condições percam cada vez mais relevância no processo de aprendizagem.

O objetivo da primeira condição acima é incentivar o agente a ter alguma oferta no *book* quando este estiver menos movimentado. No segundo caso, a intenção é exatamente a oposta. Em todos os outros casos, não há recompensas associadas.

Finalmente, definindo a condição

$$c_4 \leftarrow (\,(a_t = None) \ \wedge \ \neg(\Pi = 0)\,) \ ,$$

quando o agente estiver posicionado, tem-se que:

$$r_{t+1}^{(4)} = \mathbf{I}_{c_4} \times \begin{cases} f_3\left(ofi\right) & \text{se } ofi \geq 0.8 \text{ e } \Pi > 0 \\ f_3\left(ofi\right) & \text{se } ofi \leq -0.8 \text{ e } \Pi < 0 \\ 0 & \text{se } |ofi| < 0.1 \\ f_3\left(ofi\right) \times -1 & \text{caso contrário} \end{cases} .$$

Nos dois primeiros casos, o agente ganha um prêmio por não ter ofertas no livro quando há fluxo a favor de sua posição. O terceiro caso, não se pune o agente por não ter ofertas devido a baixa atividade no *book*.

A forma final da função de *reward* por OFI é dada por

$$r_{t+1} = r_{t+1}^{(1)} + r_{t+1}^{(2)} + r_{t+1}^{(3)} + r_{t+1}^{(4)} \ , \tag{4.4}$$

A função (4.4) também permitirá checar se os inúmeros sub-objetivos definidos prejudicarão o aprendizado do agente. Por fim, combinando as funções (4.3) e (4.4) e referindo-se a cada uma como r^{pnl} e r^{ofi}, respectivamente, define-se a terceira função de *reward* testada como:

$$r_{t+1} = \frac{r_{t+1}^{pnl}}{DV01 \times 5} + r_{t+1}^{ofi} \ . \tag{4.5}$$

Nesta função $DV01$, também conhecido como *dollar duration*, refere-se ao valor financeiro que é acrescido ou subtraído do PU de um contrato quando seu preço varia 1 *basis point*. Este valor foi multiplicado por 5 para corresponder ao $DV01$ do lote mínimo de operação dos contratos futuros de taxa de juros.

Considerando que o agente irá apregoar com o lote mínimo, e sua posição máxima será de 10 contratos, depois de escalar r_{t+1}^{pnl} pelo $DV01$ do contrato, a amplitude das recompensas geradas por (4.3) e (4.4) ficam semelhantes, não sendo preciso outros ajustes para juntar ambos sinais em uma única função.

Com o espaço de ações definido, além de uma função de *reward* para avaliar a qualidade das decisões do agente, agora falta definir o que o ele poderá observar para que o problema de aprendizagem fique completo, o que será feito a seguir.

4.2.3 Representação de Estados

Spooner (2016) explica que, a cada passo de tempo t, o agente poderá observar uma representação do ambiente, o estado s, que contém alguma combinação de variáveis que sejam informativas sobre o mercado atual. Como a escolha de ações do agente é baseada somente nos valores observados neste estado, a determinação destas variáveis impacta diretamente na habilidade do agente derivar uma estratégia lucrativa.

Dado que este trabalho é focado na criação de uma estratégia de alta frequência, é razoável que a escolha do espaço de estados seja restrita às variáveis que possam influenciar a dinâmica do LOB. O'Hara (2015) explica que modelos de *trading* desta categoria costumam ser estratégicos na maneira como posicionam suas ofertas, tentando sempre estar o mais próximo possível do topo da fila. Para tanto, tais estratégias precisam otimizar suas regras de apregoação em relação à microestrutura do mercado em que operam.

Neste contexto, há uma extensa literatura sobre diversas variáveis que podem ajudar a capturar diferentes aspectos da dinâmica do *book* de ofertas. Uma delas, que será

testada neste trabalho, será um medida de volatilidade de curtíssimo prazo chamada *High Low* (HL), sugerida por Hasbrouck e Saar (2013). Esta variável se refere a diferença entre o maior e o menor *mid-price* p^m observado dentro de um período, dividido pela média de ambos. Nesta dissertação será adotada uma janela de 10 minutos para esta medição.

Outra variável que será testada é o desbalanceamento entre o *book* da compra e o da venda, chamado de *queue imbalance*. Gould e Bonart (2016) alegam que há uma relação estatisticamente significante entre esta variável e o movimento subsequente de p^m. O indicador sugerido é definido como:

$$I(t) \coloneqq \frac{n_t^b - n_t^a}{n_t^b + n_t^a} \ ,$$

onde $I(t) \in [-1,\ 1]$ é o desbalanceamento das filas no tempo t e procura quantificar a força relativa da pressão compradora e vendedora em um $\mathcal{L}(t)$. Também discutem a perda de poder preditivo de $I(t)$ quando as filas do instrumento analisado são menores. Assim, além de ser testado $I(t)$ na representação de estados do agente, também será inclusa uma medida do volume n_t^b escalado pelo intervalo interquartil (IQR) de uma amostra extraída dos últimos dias de Janeiro de 2017, de modo que $BS_t = n_t^b/IQR$. Todas as variáveis testadas foram resumidas na tabela 4.

Nome	Tipo	Descrição
OFI_t	float	*order flow imbalance* (10 segundos)
I_t	float	*queue imbalance* do contrato
Π_t	integer	posição do agente no tempo t
BS_t	float	tamanho escalado da fila da Compra
HL_t	float	volatilidade de curto prazo (10 min)
Δ_t	integer	*bid-ask spread* no tempo t
$\bar{\delta_t^x}$	float	preços relativos médios (10 min)

Tabela 4 – Descrição das varáveis de estado testadas.

Não será inclusa uma medida do volume no melhor preço de venda n_t^a pois esta informação estará contida implicitamente no *queue imbalance*. Também serão usados o *bid-ask spread* e as médias dos preços relativos da compra e da venda para compor a representação de estado do agente.

Além das variáveis BS_t, I_t e Δ_t relacionadas ao contrato operado, os valores relacionados à um contrato com vencimento mais curto também serão checados. A utilização de variáveis relacionadas a outros contratos se justifica devido ao agente estar operando a curva de juros, onde os vértices são correlacionados.

Gould et al. (2013) argumentam que o *bid-ask spread* funciona como uma medida de liquidez do mercado. Por fim, os mesmos autores argumentam que os preços relativos, definidos como $\delta_t^x(p) = p_t^b - p$ quando compra, e $\delta_t^x(p) = p - p_t^a$ quando venda, podem ser

úteis para investigar regularidades em séries de preços. Ordens com maior preço relativo costumam ser submetidas quando a volatilidade aumenta, por exemplo.

4.3 Implementação do modelo proposto

Esta seção descreverá os passos necessários para treinamento e teste do modelo aqui descrito. Também será definido um *benchmark* para confrontar contra a performance do modelo criado e um *baseline*, que será utilizado na fase de calibração do modelo.

4.3.1 Treinamento e Teste

Idealmente, se o agente iterasse sobre a mesma base de dados repetidamente, deveria chegar algum momento em que não fosse capaz de melhorar o PnL no final do episódio. A política aprendida, que gerasse este resultado, seria a política ótima.

Um dos grandes temores durante a implementação do modelo sugerido nesta dissertação era se o agente realmente estava sendo capaz de aprender alguma coisa do ambiente. Para checar se este aprendizado está ocorrendo e ainda facilitar a análise dos resultados, as otimizações das características e parâmetros do modelo *Q-learning* serão feitas sobre uma mesma base de dados, em que o agente iterará 50 vezes seguidas para otimizar uma política. Espera-se, neste caso, que seus resultados melhorem constantemente.

Depois que a forma final do modelo estiver definida, este agente deverá otimizar uma política iterando por 10 pregões seguidos, repetindo 10 vezes cada, totalizando 100 episódios de treinamento. Por fim, a política aprendida neste processo será testada nos 4 pregões subsequentes aos dados utilizados para treinamento. Os procedimentos citados utilizarão os passos descritos no algoritmo 3. No processo de otimização, os parâmetros utilizados são $(N = 50, S = 1, Z = 0)$. Na derivação da estratégia de *trading*, $(N = 10, S = 10, Z = 3, Y = 4)$. Os passos $5 \sim 7$ foram implementados para se checar o desempenho da política aprendida na mesma base utilizada para sua criação.

Cada sessão de treinamento inclui dados da maior parte do pregão do dia especificado, começando às 9:20 e terminando às 15:40. Será permitido ao agente manter posição de no máximo 10 contratos (comprado ou vendido) durante cada episódio. No final da sessão, suas posições serão zeradas automaticamente a preço de mercado, começando o episódio seguinte com suas posições zeradas. Durante a sessão, as posições do agente também poderão ser zeradas automaticamente, caso atinjam um lucro ou prejuízo de 3 *basis points*.

Como explicado anteriormente, o agente poderá decidir, a cada 30 segundos, se mantém ofertas na melhor compra, na melhor venda, em ambos os lados ou em nenhum deles. Porém, poderá reposicionar estas ofertas a cada 10 milissegundos para ficar em

Algoritmo 3 Treino e teste do agente *Q-learning*

Entrada: Qtde de iterações sobre o mesmo episódio N,
Qtde de sessões S,
Índice da base inicial D,
Qtde de iterações *out-of-sample* Z,
Qtde de sessões *out-of-sample* Y

Inicialize agente com função $\hat{Q}$ de índice $D-1$, se existir
para cada iteração $n = 0, 1, 2, ..., N$ **faça**
  para cada sessão $s = 0, 1, 2, ..., S$ **faça**
    Atualize $\hat{Q}$, aplicando algoritmo 2 à base de índice $D+S$, e utilizando a probabilidade de exploração ϵ_t, de acordo com dinâmica adotada
    se $\epsilon_t < 0.05$ **então**
      configurar taxa de aprendizado $\alpha_t \leftarrow 0$
      configurar probabilidade de exploração $\epsilon_t \leftarrow 0$
    No final do episódio, zere qualquer a posição aberta
Salve nova função $\hat{Q}$, indexando por D
Inicialize agente com função $\hat{Q}$ de índice D,
com dinâmica de exploração $\{\epsilon_t\}_{t=0}^{\infty} \leftarrow 0$,
com dinâmica da taxa de aprendizado $\{\alpha_t\}_{t=0}^{\infty} \leftarrow 0$
para cada iteração $z = 0, 1, 2, ..., Z$ **faça**
  para cada sessão $y = 0, 1, 2, ..., Y$ **faça**
    Utilize $\hat{Q}$, aplicando algoritmo 2 à base de índice $D+S+y+1$
    No final do episódio, zere qualquer a posição aberta

conformidade com a decisão tomada no início dos 30 segundos. Se a melhor compra for 10.03 em um dado passo t, por exemplo, e o agente decidir manter ordem na melhor compra, ele irá inserir uma nova oferta a 10.03. Se, antes de $t+1$, o preço da melhor compra subir para 10.05, o agente irá inserir uma nova oferta a 10.05, mantendo sua oferta a 10.03 caso ainda tenha limite para tanto. Caso ele fique sozinho na fila a 10.05, ainda antes de $t+1$, ele cancelará esta oferta e inserirá uma nova oferta no melhor preço vigente.

Caso o mercado volte para 10.03, por exemplo, ele simplesmente manterá a oferta que tinha naquele preço anteriormente. Se esta oferta for executada, o ambiente informará ao agente que sua ordem foi preenchida para que ele atualize sua posição. Neste momento ele poderá tomar duas ações: ou executar um *hedge* depois de 5 milissegundos, caso esta opção esteja sendo testada, ou esperar até $t+1$ para tomar uma nova decisão.

4.3.2 Definição de *Benchmark*

Em 1988, o *Wall Street Journal* criou uma competição de dardos[4], onde os funcionários do jornal jogavam dardos em um alvo repleto de nomes de ações para selecionar seus portfólios, enquanto consultores profissionais de investimento montavam suas próprias

[4] Mais detalhes em <http://www.automaticfinances.com/monkey-stock-picking/>

carteiras. Depois de seis meses, compararam a performance dos dois métodos. Ajustando o resultado por nível de risco, eles observaram que os profissionais mal superaram a performance daqueles que escolheram as ações aleatoriamente.

Assim, na fase de teste do modelo, o *benchmark* escolhido para avaliar a performance do agente criado neste trabalho será o P&L de um agente aleatório, rodando sob as mesmas condições. Como descrito anteriormente, depois do modelo ser devidamente otimizado, o agente de aprendizagem será treinando utilizando os dados de 10 pregões diferentes e, então, a política criada será testada nos 4 pregões subsequentes à base de treinamento. Esta simulação de teste será rodada 5 vezes, totalizando 20 pontos de dados, e sua distribuição será comparada com a distribuição de resultados de um agente aleatório que realizará o mesmo procedimento.

O objetivo, por tanto, será superar a performance deste agente, que deve escolher suas ações aleatoriamente de um conjunto de ações A a cada passo de tempo t. Assim como o agente de aprendizagem, quando o agente aleatório também atingir seu limite $\Pi_{max} = 10$, o ambiente passará a acrescentar um *spread* nas ofertas enviadas por ele que tiverem potencial de aumentar sua posição atual.

Por fim, para comparar as distribuições de resultados de ambos agentes, será realizado um teste t de *Welch* unicaudal com variância desigual[5]. Neste teste, a hipótese nula é que o agente de aprendizado tem um P&L esperado semelhante ao do agente aleatório.

Como a implementação do teste t no *scipy*, a biblioteca de *Python* utilizada, assume que o teste é bicaudal, o teste unicaudal foi realizado dividindo-se por 2 o $p$valor e comparado o resultado a um valor crítico de 0.10, além de também exigir que o valor $t > 0$, que indicará se o valor esperado do agente de aprendizagem foi superior ao do agente aleatório.

4.3.3 Definição do *baseline*

Como explicado na subseção 4.3.1, a performance do agente criado será medida pelo *reward* médio por episódio, definido como $\left(\sum_{t=0}^{T} r_t\right)/T$, que foi capaz de gerar ao longo de uma série de simulações sobre uma mesma base de dados. A curva gerada por este agente será comparada com diferentes implementações deste mesmo agente, variando funções de *reward*, representações de estado e parâmetros. Para iniciar este processo de otimização, a figura 2 apresenta o primeiro *baseline* desta dissertação. Os dados foram obtidos utilizando a base do dia primeiro de fevereiro de 2017

Como pode ser observado, não fica claro se o resultado do agente melhorou ao longo do treinamento, além de que o modelo não conseguiu convergir para uma estratégia

[5] Mais detalhes em <https://goo.gl/GsP436>

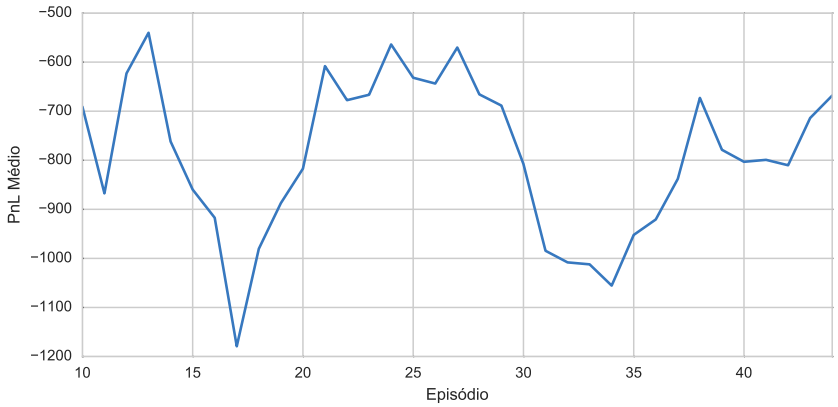

Figura 2 – média móvel de 10 episódios do *baseline* no dia primeiro de fevereiro.

ganhadora em 50 episódios.

O P&L médio (eixo y) se refere a média móvel de 10 períodos do resultado financeiro obtido pelo agente no final de cada episódio (eixo x). Esta suavização foi adotada em todos os gráficos plotados nesta dissertação para enfatizar a tendência dos dados visualizados. Ainda que o P&L médio não se refira ao *reward* médio do período, será mostrado na seção 5.1 o porquê do primeiro *baseline* ser a curva de P&L, e não a de recompensa. Este *baseline* será atualizado a cada decisão tomada sobre a forma final modelo, que será feita na próxima seção, e utilizado para uma próxima comparação.

A representação de estado utilizada pelo *baseline* acima inclui todas as variáveis mencionadas na tabela 4, além de usar a função de recompensa (4.3), taxa de aprendizado $\alpha = 0.5$, fator de desconto $\gamma = 0.5$ e decaimento linear da taxa de exploração. Como pontuado antes, espera-se que a otimização sobre a mesma base de dados faça o agente melhorar seu resultado naturalmente.

Um dos grandes desafios deste trabalho foi colocar o modelo de *trading* como um problema de *reinforcement learning*. Além das dificuldades com a construção do simulador, as decisões sobre quais ações o agente poderia tomar, sobre o tipo de informação que poderia extrair do ambiente e sobre como sua performance seria julgada, não foram triviais. Na próxima seção, será descrito todo o processo para definição do modelo final utilizado.

5 Resultados

O modelo proposto neste trabalho foi utilizado para derivar uma estratégia de *trading* de alta frequência aplicada ao mercado brasileiro de taxa de juros. Na seção 5.1, serão apresentados todos os testes realizados para escolher entre as diferentes opções de funções de *reward* e representações de estado, além de demonstrar o processo de otimização de parâmetros. A seção 5.2 apresenta os resultados da estratégia quando aplicada a 8 pregões do mês de fevereiro de 2017, nos vencimentos DI1F19, DI1F21 e DI1F23.

5.1 Especificação do modelo

Nesta seção são apresentados os testes realizados para obtenção do modelo final testado neste trabalho. Inicialmente, a subseção 5.1.1 apresenta como se decidiu pela função de recompensa. Depois, na subseção 5.1.2 é realizada uma investigação sobre o impacto de diferentes subconjunto das *features* testadas no aprendizado do agente. Por fim, na subseção 5.1.3, os parâmetros do modelo de *Q-learning* são otimizados.

5.1.1 Escolha da função de *Reward*

Foram testadas três funções de recompensa diferentes, como apresentado na subseção 4.2.2. A figura 3 a seguir apresenta a evolução do P&L do agente ao longo de 45 episódios de treinamento, utilizando cada uma das três funções definidas, com a política sendo otimizada sobre uma mesma base de dados. O P&L foi utilizado para comparação entre as diferentes funções por cada uma utilizar noções de *reward* distintas.

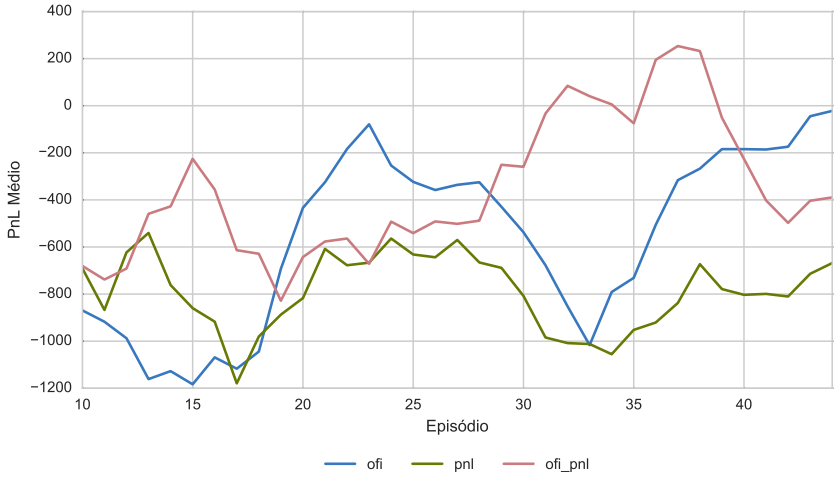

Figura 3 – Média móvel de 10 períodos do P&L do final do episódio, usando diferentes funções de *reward*.

A primeira função testada, utilizada no atual *baseline*, foi P&L do agente. O objetivo desta função foi influenciar explicitamente o agente a procurar pela política que gerasse maior retorno (ou menor prejuízo). Este foi o pior resultado entre as três funções testadas. A segunda função testada, baseada na variável OFI, teve como objetivo enviesar o agente a se focar na qualidade da apregoação, em vez de terminar o episódio positivo. Ainda que tenha resultado no P&L final mais alto, também foi a curva mais volátil dos testes.

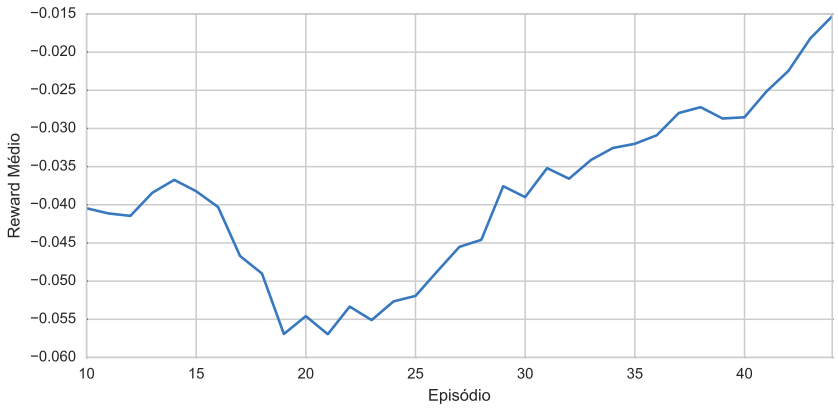

Figura 4 – Média móvel de 10 períodos da média de recompensa no episódio, utilizando função de *reward* ofi_pnl.

Por fim, a terceira função, ofi_pnl, procurou juntar ambas noções de recompensa em um único sinal e foi esta que apresentou resultados mais consistentes ao longo do treinamento. Como explicado no capítulo 3, o que se procura otimizar em um problema de RL é a recompensa acumulada no episódio. A figura 4 exibe a curva de recompensa média da função ofi_pnl. Ainda que não tenha obtido o resultado final mais alto, esta função que será utilizada como base de comparação para as próximas otimizações.

Um teste útil nesta etapa da otimização foi forçar o agente a sempre tomar a mesma ação durante toda a simulação. Este procedimento ajudou a entender o comportamento e limitações das diferentes funções de *reward*, bem como a encontrar falhas em suas implementações.

5.1.2 Escolha da representação de estado

Como explicado na subseção 3.1.3, é importante que a representação de estado utilizada seja informativa para que a solução do problema de RL em questão seja viável. Foi explicado que, entre outras coisas, esta representação deve ser útil para prever recompensas futuras.

Na figura 5 está exposta a matriz de correlação entre todas as variáveis testadas nesta dissertação. Para ajudar a entender a relação destas variáveis com o movimento futuro do ativo, foram incluídas a variação do *mid price* p^m e do OFI entre t e $t+1$. As

variáveis com sobrescrito l ou c, como em I_t^l e I_t^c, se referem à variável relacionada ao vencimento mais longo e ao vencimento mais curto, respectivamente.

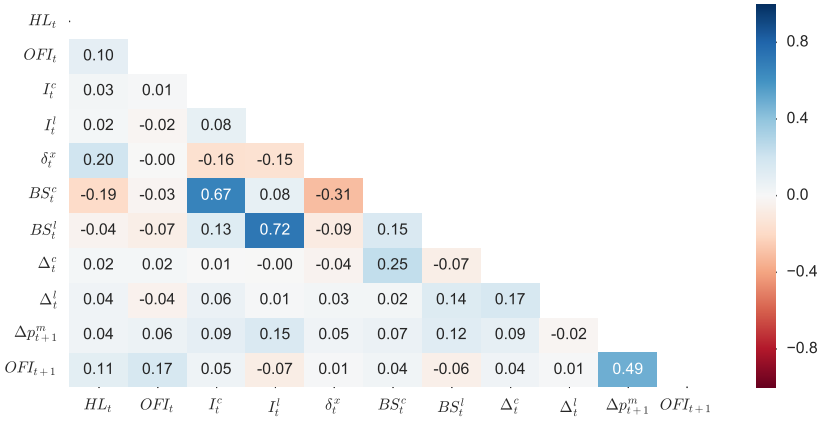

Figura 5 – Matriz de correlação das variáveis testadas dentro da representação de estado.

Nesta figura é possível observar, por exemplo, a alta correlação entre a variável *bid size* BS_t e o *queue imbalance* I_t, tanto do contrato longo como do curto. As variáveis p_{t+1}^m e OFI_{t+1} também apresentaram uma correlação mais alta. Porém quando se compara OFI_t com p_{t+1}^m, esta relação não se repete. Na figura 6 foram testadas as observações derivadas da figura 5. O *baseline* inicial de cada teste abaixo foi destacado com um asterisco na legenda, e todas as curvas de *reward* plotadas a seguir serão construídas como médias móveis de 10 períodos.

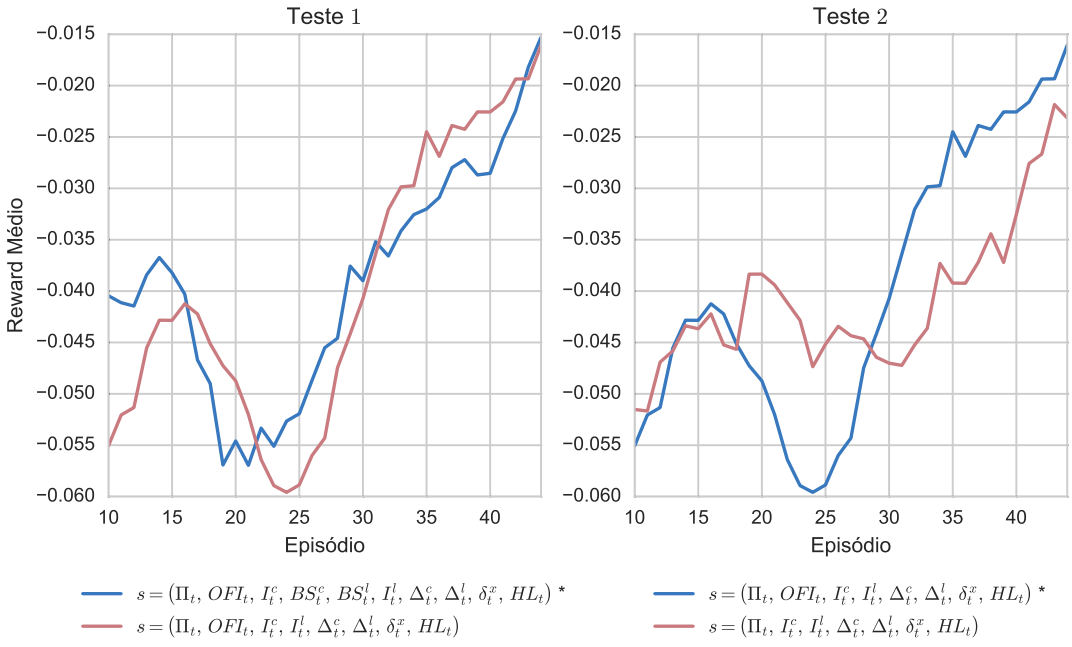

Figura 6 – Comparação e atualização do *baseline* utilizando diferentes representações de estado (1).

Primeiro, no teste 1, se excluiu as variáveis BS_t^c e BS_t^l e se observou pouco impacto na curva de recompensa, estabelecendo-se como novo *baseline* a representação sem estas variáveis. Podemos justificar esta escolha pelo princípio da *navalha de Occam*[1]. Já no teste 2, foi utilizada a representação do *baseline* atual sem o OFI_t. A curva de *reward* com a representação inicial, apesar de ter apresentado uma degradação inicial, superou o estado sem o OFI em mais da metade da simulação e, portanto, a referência foi mantida.

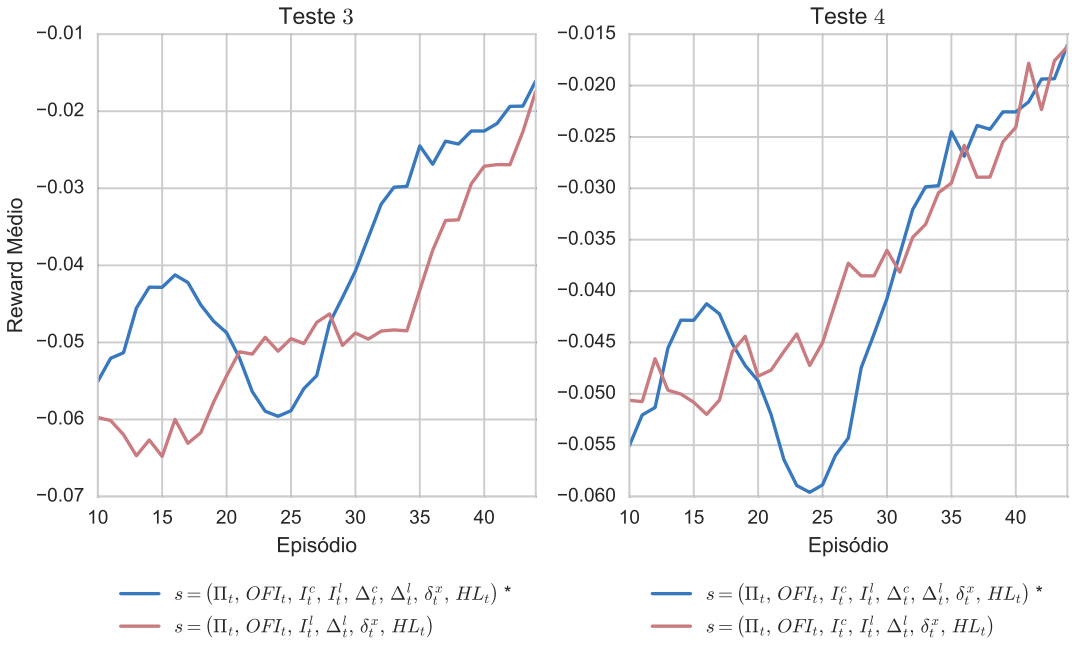

Figura 7 – Comparação e atualização do *baseline* utilizando diferentes representações de estado (2).

No teste 3 (figura 7), excluiu-se as variáveis relacionadas com o contrato curto. A variabilidade da curva diminui, porém ela fica abaixo da curva do *baseline* atual quase todo o treinamento, igualando seu resultado no final. Optou-se por manter a referência atual, neste caso. No teste 4, excluiu-se apenas o *bid-ask spread* do contrato curto Δ_t^c. O resultado desta representação de estado foi visivelmente mais estável que a anterior, sendo selecionada como novo *baseline*.

Na figura 8 foram testadas representações de estado sem o *bid-ask spread* do contrato longo (teste 5) e depois sem o *queue imbalance* do mesmo vencimento (teste 6). No primeiro caso, a remoção do Δ_t^l aumentou a variabilidade do *reward*, então a referência foi mantida. No segundo caso, a representação mais simples superou a referência atual apenas nas últimas 5 simulações e, assim, sua superioridade não é conclusiva. Além disso, operar qualquer instrumento olhando unicamente para seu *spread* parece inadequado e, portanto, optou-se por manter o *baseline* atual.

[1] *Occam's razor: Among competing hypotheses, the one with the fewest assumptions should be selected.*

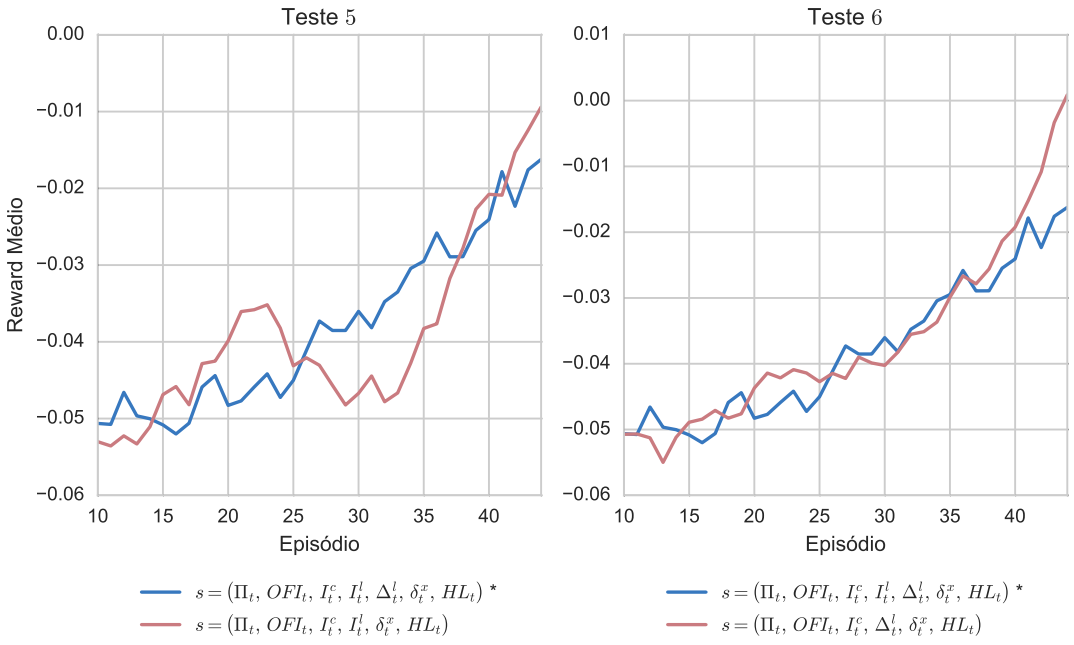

Figura 8 – Comparação e atualização do *baseline* utilizando diferentes representações de estado (3).

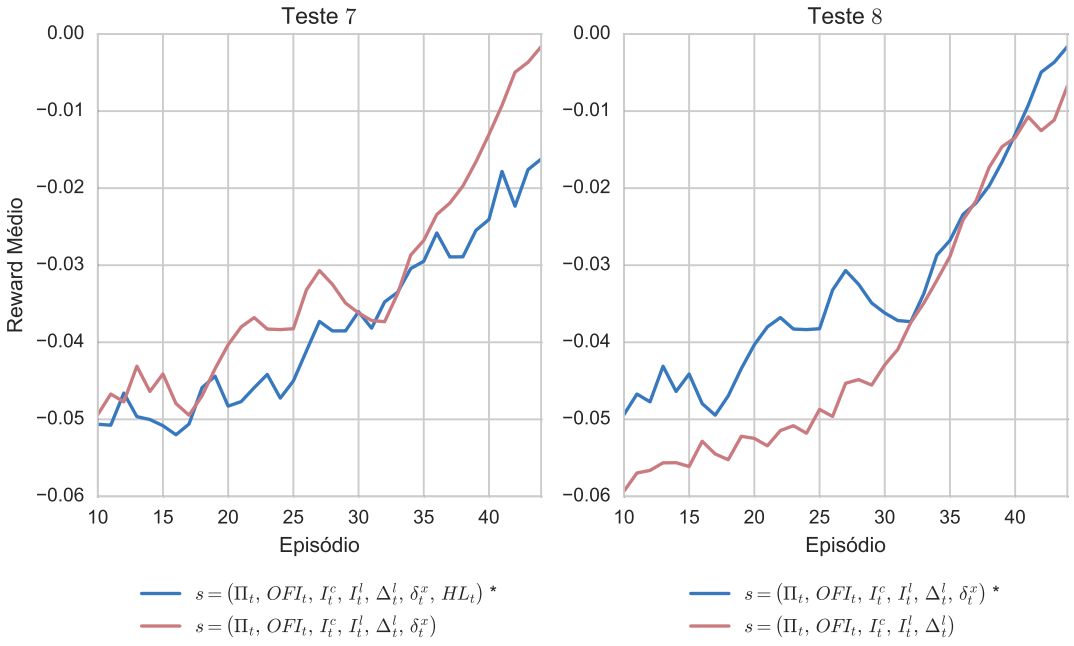

Figura 9 – Comparação e atualização do *baseline* utilizando diferentes representações de estado (4).

Por fim, na figura 9, excluindo o *High Low* da representação de estado de referência (teste 7), obteve-se um resultado superior. No teste 8, também se excluiu o preço relativo médio, mas o teste foi um pouco inferior a representação de estado apenas sem o HL_t. Assim, a representação de estado que será utilizada nas próximas seções ficou sendo $s = \langle \Pi_t, OFI_t, I_t^c, I_t^l, \Delta_t^l, \delta_t^x \rangle$.

5.1.3 Otimização dos parâmetros do modelo

Há três parâmetros diferentes para serem otimizados no modelo: a taxa de aprendizado α, o fator de desconto γ e o decaimento da taxa de exploração ϵ. Todos estes parâmetros influenciam diretamente na capacidade de aprendizagem do *Q-learning* e podem determinar se o modelo é capaz de obter ou não uma solução satisfatória.

Na figura 10, pode-se observar a otimização dos dois primeiros parâmetros. O α foi testado com 5 configurações diferentes, sendo que $\alpha = 0.5$ faz parte do *baseline* do modelo. Como pode-se observar, tirando a curva do $\alpha = 0.7$, que superou a referência algumas vezes, nenhuma outra configuração ficou melhor que a inicial. Porém, mesmo o $\alpha = 0.7$ degradou no final da simulação, sendo mantido o *baseline atual*.

O segundo teste foi realizado com diferentes valores de γ. Como explicado na subseção 3.1.2, este parâmetro é uma constante que determina o valor relativo da recompensa imediata e adiada. Quanto mais próximo de zero, maior a ênfase em recompensas imediatas. Todos os testes foram realizados com $\alpha = 0.5$ e variando o γ, sendo que $\gamma = 0.5$ faz parte do *baseline*. Neste caso, nenhuma configuração superou a referência atual.

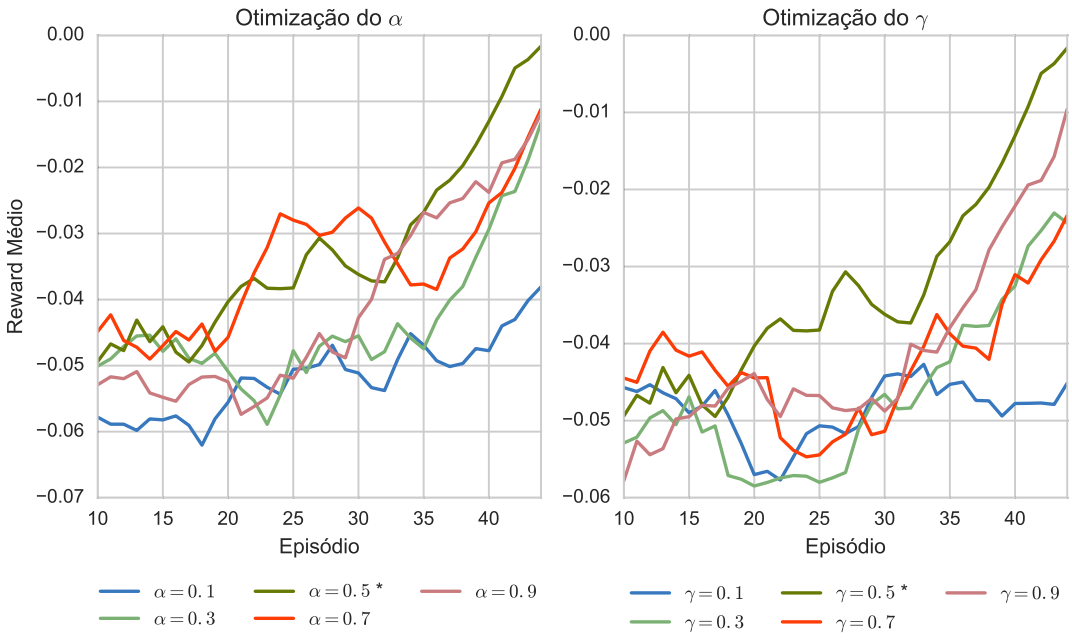

Figura 10 – Otimização da taxa de aprendizado α e do fator de desconto γ.

Na figura 11 está o resultado de diferentes dinâmicas de decaimento da taxa de exploração. Foram testadas três opções diferentes, sendo que a forma de cada dinâmica foi exibida no segundo gráfico. Em todas as funções, a e b são constantes, n é o passo da iteração atual e as funções são construídas de maneira que seus valores convirjam para zero ao final da simulação. O decaimento trigonométrico $\epsilon = \cos(an)$, em que o agente fica mais tempo explorando diferentes ações durante a simulação, apresentou o pior resultado.

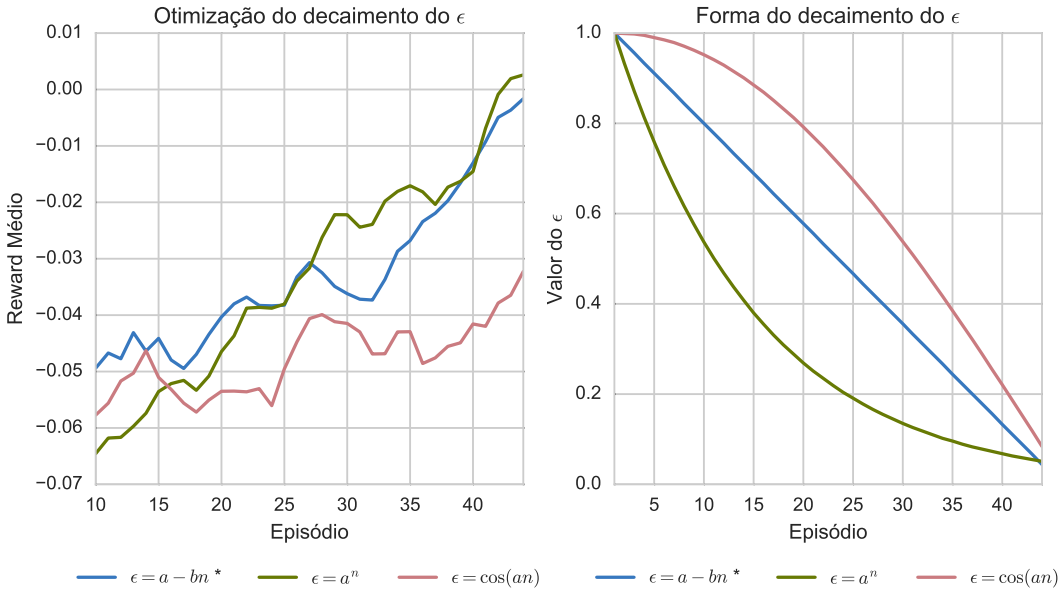

Figura 11 – Comparação da função de decaimento da taxa de exploração ϵ.

O decaimento exponencial $\epsilon = a^n$, em que o agente se compromete mais rápido com o que for aprendendo, apesentou um resultado um pouco melhor do que o *baseline* em uma parte da simulação, convergindo para quase o mesmo valor no final. Assim, optou-se por preservar a dinâmica linear $\epsilon = a - bn$, adotada na referência atual, para os próximos testes. Finalmente, a figura 12 apresenta a comparação, tanto em *reward* médio, como em P&L, entre modelo inicial desta subseção e o final, obtido depois de todas as otimizações.

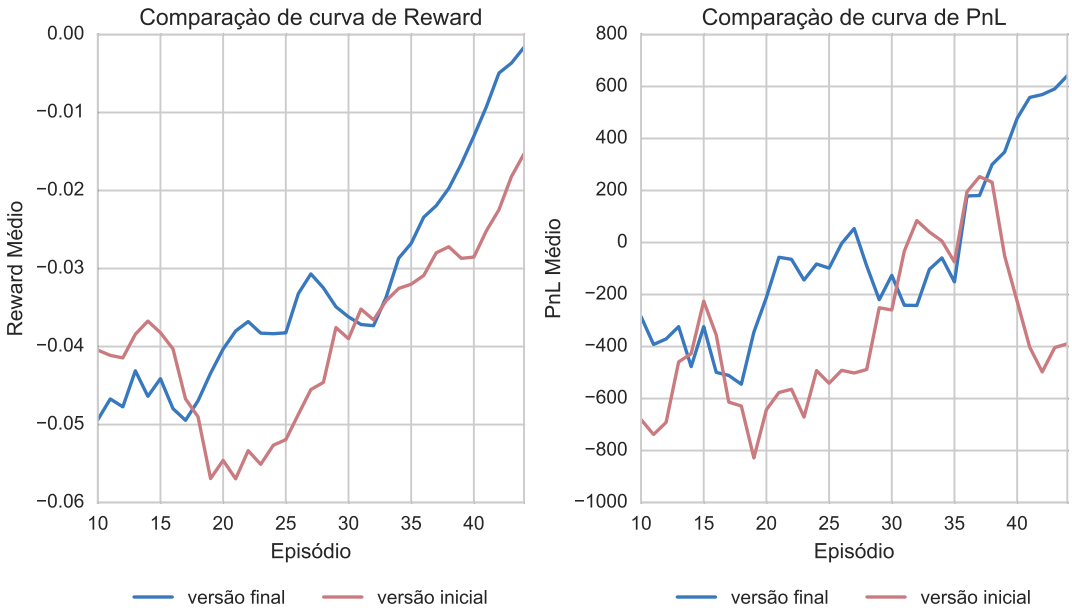

Figura 12 – Comparação da curva de recompensa e P&L da forma inicial e final do modelo.

A versão final do modelo usa, então, uma função de recompensa baseada em OFI e P&L, representação de estado $s = \langle \Pi_t, OFI_t, I_t^c, I_t^l, \Delta_t^l, \delta_t^x \rangle$, $\alpha = 0.5$, $\gamma = 0.5$ e decaimento linear da taxa de exploração. Como pode-se observar no primeiro gráfico acima, sobre o *reward*, o modelo final superou o inicial substancialmente. Esta superioridade, ainda que não na mesma proporção, também pode ser observada no gráfico de P&L.

Talvez a parte mais árdua desta dissertação tenha sido todo este processo de especificação do modelo descrito até aqui. Não apenas pelas inúmeras possibilidades que foram testadas, mas também pela dificuldade de validar os resultados de simulações de alta frequência. Pequenos erros na especificação de alguma função, ou no comportamento do simulador, podem ser difíceis de identificar e muitas vezes ficam evidentes apenas depois de muitas iterações. Esta dissertação, por exemplo, tomou cerca de 180 horas de processamento entre testes com alguma imprecisão, diferentes experimentos, além das otimizações[2]. É interessante frisar então, que na construção de estudos como os realizados aqui, os códigos e as funções que forem sendo criadas devem ser constantemente validadas para diminuir este risco de retrabalho.

5.2 Modelo de *trading* de alta frequência

Neste trabalho foi proposto a utilização do *framework* de RL para a criação de uma estratégia de *trading* de alta frequência em DI futuro. Como argumentado por O'Hara (2015), *high frequency trading* (HFT) é uma termo pouco preciso que define um conjunto diverso de estratégias e comportamentos.

A SEC, a CVM americana, usa uma série de possíveis atributos para definir o que caracterizam operações de alta frequência. Citam que HFT envolve o uso de algoritmos sofisticados e/ou de baixa latência para rotear e executar ordens. Além disso, pode envolver ainda o uso de *co-location*, intervalos curtíssimos de tempo entre a abertura e o fechamento de posições e o não carrego de posições significantes de um pregão para o outro.

Os testes na figura 13 a seguir foram realizados com o modelo especificado na seção 5.1. O treinamento e teste do agente foi realizado em 2 períodos e 3 instrumentos distintos. Diferentemente dos testes realizados até aqui, cada rodada de treinamento utilizou 10 pregões diferentes, sendo que estas rodadas foram repetidas 10 vezes, totalizando 100 episódios de treinamento no total[3]. Isso possibilitou ao agente checar se o que aprendeu em um pregão foi adequado para os outros.

Os períodos de treinamento compreenderam primeiro o dia 1 ao dia 14 de fevereiro e depois o dia 7 ao dia 20. O modelo foi testado com os vencimentos DI1F19, DI1F21

[2] O serviço de *cloud computing* da empresa *Domino Data Lab*, e a facilidade que oferecem para instanciar vários servidores diferentes ao mesmo tempo, foram de grande valia para esta dissertação

[3] Este procedimento foi descrito na subseção 4.3.1.

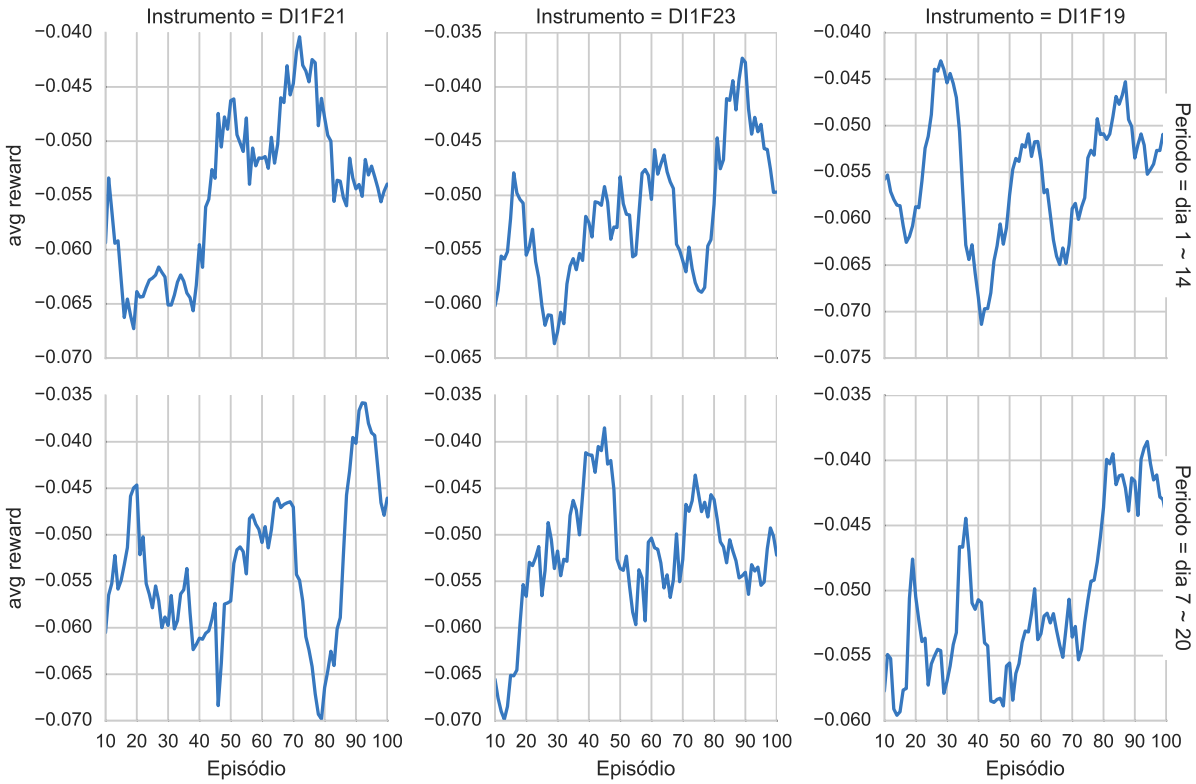

Figura 13 – Curvas de recompensa de funções treinadas em diferentes instrumentos e períodos.

e DI1F23. Em cada episódio (definido como um pregão), foi permitido que o agente se posicionasse em 10 contratos no máximo e usou *stop* de 3 *basis points* em relação ao preço médio da posição aberta. No final de cada episódio, o agente fechou qualquer posição aberta à mercado, tanto no treinamento como no teste.

Como pode-se observar, as políticas resultantes ficaram instáveis e nenhuma convergiu para uma que gerasse *rewards* positivos. Esta instabilidade do *Q-learning*, quando aplicado à dados financeiros, foi observada em alguns trabalhos como em Spooner (2016) e Du, Zhai e Lv (2016). Ainda assim, nota-se que o algoritmo foi capaz de melhorar gradativamente sua política, resultado em *rewards* médios no final do treinamento mais altos que os observados no início.

Cada um dos modelos criados foi testado nos 4 pregões subsequentes à seu período de treinamento e o resultado financeiro da simulação foi comparada ao resultado obtido pelo *benchmark* definido na seção 4.3.2. Cada simulação, tanto do agente de aprendizado (ou *learner*) como do *benchmark* (ou *random*), foi repetida 20 vezes cada. A figura 14 mostra o resultado médio destas simulações em cada passo de tempo.

O período de teste para as políticas criadas com os dados do dia 1 ao dia 14 foi do dia 15 ao dia 20 de fevereiro, e as políticas obtidas com os dados do dia 7 ao dia 20 foram testadas com os dados do dia 21 ao dia 24 de fevereiro. Como pode-se observar acima,

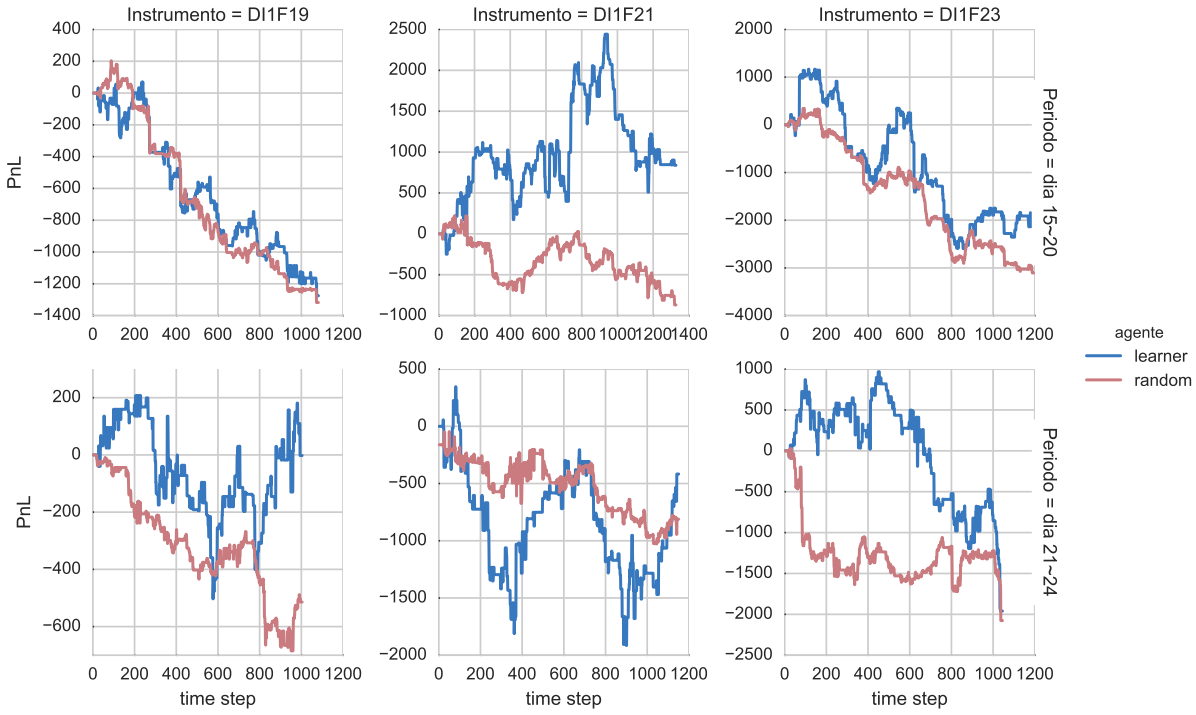

Figura 14 – Comparação de P&L em teste *out-of-sample* em diferentes instrumentos e períodos.

apenas em um dos testes a política obtida resultou em P&L positivo no final da simulação.

Ativo	Treinamento	Teste	P&L agente	*benchmark*	*t-value*	*p-value*
$F19(F21)$	dias $1 \sim 14$	$15 \sim 20$	R$ -1275.19	R$ -1317.79	0.22	0.414
$F21(F19)$*	dias $1 \sim 14$	$15 \sim 20$	R$ 838.57	R$ -869.40	1.62	0.058
$F23(F21)$*	dias $1 \sim 14$	$15 \sim 20$	R$ -1997.45	R$ -3105.15	1.55	0.063
$F19(F21)$*	dias $7 \sim 20$	$21 \sim 24$	R$ -2.24	R$ -514.14	1.69	0.049
$F21(F19)$	dias $7 \sim 20$	$21 \sim 24$	R$ -416.98	R$ -811.54	0.32	0.375
$F23(F21)$	dias $7 \sim 20$	$21 \sim 24$	R$ -1959.53	R$ -2076.66	0.11	0.455

Tabela 5 – Resultados no período de teste, comparados ao *benchmark*.

Os resultados finais das simulações estão expostos na tabela 5, bem como os resultados do teste t de *Welch*, mencionado na seção 4.3.2. Como explicado antes, a hipótese nula é que os valores esperados das distribuições comparadas são semelhantes. Como pode-se observar, rejeitou-se a hipótese nula à um nível de significância de 10% em três dos seis testes realizados, sugerindo que os resultados do agente de aprendizado foram superiores aos do agente aleatório nestes casos ($p\text{-}value < 0.1$; $t\text{-}value > 0$). Estes testes foram ressaltados com um asterisco na coluna *Ativo*. O ativo entre parênteses corresponde ao segundo instrumento que o agente observou para tomar suas decisões.

Ainda que o agente criado não tenha superado significantemente o aleatório em todos os testes realizados e não tenha convergido para uma política lucrativa na maioria dos casos, o P&L dele foi superior ao *benchmark* em todas as simulações. Este resultado

sugere que o *Q-learning* foi capaz de otimizar uma política mesmo em ambientes marcados por altos níveis de ruído, como o mercado financeiro.

O fato do agente aleatório não ter conseguido terminar nenhuma das simulações positivo, em média, pode indicar que a especificação do ambiente está inadequada, e as restrições impostas podem ter impedido o aprendizado de políticas que resultassem em resultados positivos. Entre estas restrições pode-se citar a espera de 30 segundos antes de mudar de ação, o uso *stops* fixos, a não inclusão de ofertas a mercado na aprendizagem, entre outras.

Tentou-se rodar as políticas aprendidas usando intervalos mais curtos de mudança de ação, porém o resultado não foi satisfatório. O agente começou a operar com muita frequência, algumas vezes gerando prejuízos apenas com custo de transação. A inclusão de ordens a mercado no conjuntos de ações permitidas talvez dispensasse o uso de *stops*. Estas ações poderiam ser permitidas, por exemplo, apenas quando o agente superasse um nível de lucro/prejuízo pré-estabelecido.

6 Conclusões

A presente dissertação apresentou um modelo para aprender uma estratégia de *trading* diretamente de dados históricos do livro de ofertas, utilizando o *framework* de aprendizado por reforço com generalização de função.

Em problemas de RL, não é dito ao agente quais são as decisões que deve tomar, mas sim fornecido um ambiente com o qual ele possa interagir e descobrir por conta própria a maneira ótima de agir. Para tanto, foi necessário criar um ambiente complexo de simulação, onde os *books* de ofertas de múltiplos instrumentos foram reconstruídos e simulados ao mesmo tempo, possibilitando ao agente observar e/ou atuar nestes diferentes mercados de forma sincronizada. Além dos custos de transação, foi permitido que o agente mantivesse várias ordens ativas ao mesmo tempo e foi adicionada latência às ofertas enviadas por ele, aumentando a aderência da simulação a condições reais.

Dentro deste ambiente, foi dada ao agente a tarefa de aprender se deveria possuir ou não ordens nos melhores preços observados. A qualidade destas decisões foi avaliada não só pelo retorno financeiro consequente delas, como em grande parte dos trabalhos citados no capítulo 2, mas também pela coerência com a qual o agente posicionou suas ofertas. O aprendizado, sob estas condições, ocorreu por meio do modelo *Q-learning* e, devido a alta dimensionalidade dos estados que foram expostos ao agente, optou-se por generalizar a função criada utilizando o modelo de *tile coding*.

Depois do problema abordado nesta dissertação ser devidamente definido no capítulo 4, o agente foi treinado repetidamente usando a mesma base de dados. Ainda que o resultado financeiro dele tenha melhorado ligeiramente ao longo do treinamento, observou-se que não convergiu para uma estratégia lucrativa. Assim, no capítulo 5, seguiu-se explorando cada componente do *framework* de RL para encontrar a melhor combinação de elementos possível dentro do contexto criado. Todo o processo de otimização foi feito utilizando a mesma base de dados e avaliado sob a ótica da noção de recompensa adotada.

Escolhida a especificação final do modelo, o agente foi treinado utilizando 3 vencimentos em 2 períodos diferentes, contendo 10 pregões cada um. Iterou-se 10 vezes por cada período, totalizando 100 episódios de treinamento. Observou-se que o modelo *Q-learning* conseguiu melhorar o *reward* médio por episódio em todas as políticas criadas. Porém, nenhuma destas políticas convergiu para uma que gerasse *rewards* positivos e, em alguns casos, a função ficou instável.

Por fim, as políticas encontradas foram testadas nos 4 pregões seguintes aos períodos de treinamento. Os resultados destes testes foram comparados com os resultados de um agente aleatório, que operou sob as mesmas restrições, e serviu como *benchmark* desta

dissertação. Conclui-se que o agente de aprendizagem superou significantemente o aleatório em 3 dos 6 testes realizados, mas teve um resultado financeiro melhor em todos. Mesmo superando o *benchmark*, o agente criado não foi capaz de gerar lucro na maioria dos testes. Observou-se que a não convergência para uma política que gerasse resultados positivos pode ter relação com as restrições adotadas.

Ao longo desta dissertação, foram confirmadas várias das afirmações feitas por outros autores, como a utilidade de variáveis baseadas em microestrutura, sugerido por Fletcher (2012) e da importância da função de *reward* e do processo de *feature design* para atingir a performance ótima, como mencionado por Du, Zhai e Lv (2016) e Kearns e Nevmyvaka (2013). Diferentemente de Spooner (2016), neste trabalho foram encontradas evidências do benefício da inclusão do *queue imbalance* no espaço de estados do ambiente.

Os resultados desta pesquisa também sugerem que não há um único aspecto de um problema de RL que seja mais relevante do que os outros na aprendizagem de um agente. Por exemplo, não apenas os parâmetros do modelo influenciaram na política aprendida, como também impactaram a aprendizagem as particularidade do problema que se atribuiu ao ambiente, como o uso de *stops* compulsórios e o tempo de espera para modificar as decisões, e aquelas que se atribuiu ao agente, como o não uso de ordens a mercado.

Colocadas todas as observações acima, há algumas possíveis extensões deste trabalho. Primeiro, seria interessante pesquisar uma função de recompensa que seja adequada para treinar um agente de aprendizagem com intervalos mais curtos entre uma decisão e outra. Talvez a função aqui utilizada fosse aplicável, porém precisaria ser adaptada para as possíveis peculiaridades do intervalo escolhido.

Também poderia ser pesquisada outras maneiras de incorporar ordens de *stop* na simulação. Este trabalho optou por deixar esta decisão para o ambiente, porém isso poderia ser facilmente incorporado ao aprendizado do agente se fosse permitido que usasse ordens a mercado. Alguma punição ou recompensa adicional poderia ser pensada para tentar influenciar o agente a não agredir mercado por qualquer razão, por exemplo.

Ou ainda, o exemplo de Lee et al. (2007) poderia ser seguido, e o aprendizado poderia ser divido em mais de um modelo, cada um com um espaço de estados, de ações e função de *reward* diferentes. Assim, poderia se criar um modelo apenas para aprender quando manter uma oferta ou não no *book*, como realizado aqui, e outro para aprender quando fechar uma posição aberta.

Como observado anteriormente, as políticas aprendidas pelo *Q-learning* ficaram instáveis em alguns casos e esta característica do modelo, quando aplicado à ambientes com alto nível de ruído, já foi observada em outros trabalhos. Então, outra possível extensão poderia ser repetir a pesquisa realizada aqui, porém testando diferentes modelos de *reinforcement learning*, como o SARSA, o *double Q-learning*, também testado por

Spooner (2016), e o *recurrent reinforcement learning*, testado por Du, Zhai e Lv (2016). Também seria interessante estudar as vantagens e desvantagens em tratar os estados do ambiente por discretização direta, pelo agrupando deles por algum método de *clustering*, pela utilização do modelo *tile coding*, como feito nesta dissertação, e, por fim, utilizando algum método de *deep learning*.

Outra ideia seria utilizar *feedback* humano para ajudar a construir a funções de recompensa. Em colaboração recente entre os centros de pesquisa *OpenAI* e *DeepMind*, Christiano et al. (2017) obtiveram resultados interessantes ao usarem uma função criada a partir de dados classificados por supervisores humanos para incentivar um agente a aprender objetivos complexos. Os dados foram coletados interativamente, apresentando aos supervisores um vídeo do comportamento do agente de aprendizagem em duas situações diferentes. Foi pedindo, então, que classificassem qual destes vídeos mais se aproximava do objetivo desejado. Então, a função de *reward* era ajustada com os novos dados e uma nova rodada de treinamento era realizada, produzindo mais dois vídeos para serem classificados, e assim sucessivamente.

Esta abordagem poderia ser utilizada, em conjunto com o P&L do agente, para tentar ensiná-lo a apregoar de maneira ótima. O comportamento do agente em estados semelhantes, por exemplo, poderia ser apresentado para um operador humano para que classificasse onde o agente se comportou melhor. A limitação desta abordagem, como apontado pelos autores, é que a função de recompensa resultante desta classificação seria tão boa quanto a intuição do supervisor sobre o comportamento apropriado para as situações apresentadas.

Finalmente, seguindo uma linha ligeiramente diferente da adotada neste trabalho, e inspirada por Meucci (2011), o *framework* de RL poderia ser utilizado em conjunto com algum modelo de precificação, baseado na probabilidade neutra a risco $\mathbb{Q}$, para operar travas de DI. A trava de DI, também chamada de *spread* de juros, consiste em comprar um vencimento de DI e zerar a *duration* deste contrato em um segundo vencimento. A distância do preço de mercado desta estratégia para o preço justo estimado pelo modelo poderia ser utilizada dentro da representação de estado, ou ainda inserida dentro do ambiente, impedindo expressamente o agente de apregoar em certos preços. Medidas de inclinação da curva poderiam ser consideradas para fazer parte do estado do ambiente, e alguma noção de *reward* adequada para o problema precisaria ser pesquisada.

Referências

ARNTZ, M.; GREGORY, T.; ZIERAHN, U. The risk of automation for jobs in oecd countries: A comparative analysis. *OECD Social, Employment, and Migration Working Papers*, Organisation for Economic Cooperation and Development (OECD), n. 189, 2016.

BELLMAN, R. The theory of dynamic programming. 1954.

BUSONIU, L. et al. *Reinforcement learning and dynamic programming using function approximators*. Boca Raton, USA: CRC press, 2010. v. 39.

CHAN, N. T.; SHELTON, C. *An electronic market-maker*. Boston, USA, 2001.

CHRISTIANO, P. et al. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.

CONT, R.; KUKANOV, A.; STOIKOV, S. The price impact of order book events. *Journal of financial econometrics*, Oxford Univ Press, v. 12, n. 1, p. 47–88, 2014.

CUMMING, J.; ALRAJEH, D.; DICKENS, L. *An Investigation into the Use of Reinforcement Learning Techniques within the Algorithmic Trading Domain*. Dissertação (Mestrado) — Imperial College London, 2015.

DU, X.; ZHAI, J.; LV, K. Algorithm trading using q-learning and recurrent reinforcement learning. Citeseer, v. 1, p. 1, 2016.

FERRUCCI, D. et al. Building watson: An overview of the deepqa project. *AI magazine*, v. 31, n. 3, p. 59–79, 2010.

FLETCHER, T. *Machine learning for financial market prediction*. Tese (Doutorado) — UCL (University College London), 2012.

FREY, C. B.; OSBORNE, M. A. The future of employment: how susceptible are jobs to computerisation? *Technological Forecasting and Social Change*, Elsevier, v. 114, p. 254–280, 2017.

GOULD, M. D.; BONART, J. Queue imbalance as a one-tick-ahead price predictor in a limit order book. *Market Microstructure and Liquidity*, World Scientific, v. 2, n. 02, p. 165, 2016.

GOULD, M. D. et al. Limit order books. *Quantitative Finance*, Taylor & Francis, v. 13, n. 11, p. 1709–1742, 2013.

HALL, T.; KUMAR, N. *Why Machine Learning Models Often Fail to Learn: QuickTake Q&A*. 2017. [Online; publicado em 20-Novembro-2016]. Disponível em: <https://goo.gl/aHBWko>.

HASBROUCK, J.; SAAR, G. Low-latency trading. *Journal of Financial Markets*, Elsevier, v. 16, n. 4, p. 646–679, 2013.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning*. New York, USA: Springer New York, 2009.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, v. 4, p. 237–285, 1996.

KEARNS, M.; NEVMYVAKA, Y. Machine learning for market microstructure and high frequency trading. 2013.

KIM, A. J.; SHELTON, C. R.; POGGIO, T. Modeling stock order flows and learning market-making from data. 2002.

KPMG International. *Transformative change: How innovation and technology are shaping an industry*. 2016.

LEE, J. W. et al. A multiagent approach to q-learning for daily stock trading. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, IEEE, v. 37, n. 6, p. 864–877, 2007.

LI, X. et al. An intelligent market making strategy in algorithmic trading. *Frontiers of Computer Science*, Springer, v. 8, n. 4, p. 596–608, 2014.

LUESKA, L. C. *Modelo HJM multifatorial com processo de difusão com jumps aplicado ao mercado brasileiro*. Dissertação (Mestrado) — Fundação Getulio Vargas, 2016.

MELO, F. S.; MEYN, S. P.; RIBEIRO, M. I. An analysis of reinforcement learning with function approximation. p. 664–671, 2008.

MEUCCI, A. 'P' versus 'Q': Differences and commonalities between the two areas of quantitative finance. *GARP Risk Professional*, p. 47–50, 2011.

MITCHELL, T. *Machine Learning*. Boston, USA: McGraw-Hill, 1997. (McGraw-Hill International Editions).

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of Machine Learning*. London, UK: The MIT Press, 2012.

NEVMYVAKA, Y.; FENG, Y.; KEARNS, M. Reinforcement learning for optimized trade execution. p. 673–680, 2006.

O'HARA, M. High frequency market microstructure. *Journal of Financial Economics*, Elsevier, v. 116, n. 2, p. 257–270, 2015.

RONCALLI, T. *Big Data in Asset Management*. 2014. Big Data in Finance and Insurance, 7th Financial Risks International Forum. Disponível em: <https://goo.gl/9yzMx4>.

SHERSTOV, A. A.; STONE, P. Function approximation via tile coding: Automating parameter choice. p. 194–205, 2005.

SHREVE, S. *Stochastic calculus for finance I: the binomial asset pricing model*. Pittsburgh, USA: Springer Science & Business Media, 2012.

SILVER, D. et al. Mastering the game of go with deep neural networks and tree search. *Nature*, Nature Publishing Group, v. 529, n. 7587, p. 484–489, 2016.

SPOONER, T. L. *Reinforcement Learning for High-Frequency Market Making in Limit Order Book Markets*. Dissertação (Mestrado) — the University of Liverpool, 2016.

SUTTON, R. S.; BARTO, A. G. Reinforcement learning: An introduction. Em progresso. Acessado em <http://incompleteideas.net/sutton/book/the-book-2nd.html>. 2017.

WASKOW, S. J. *Aprendizado por reforço utilizando tile coding em cenários multiagente*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, 2010.

WATKINS, C. J.; DAYAN, P. Q-learning. *Machine learning*, Springer, v. 8, n. 3-4, p. 279–292, 1992.

WATKINS, C. J. C. H. *Learning from delayed rewards*. Tese (Doutorado) — University of Cambridge England, 1989.

XAVIER, A. *The Distribution/Intermediation Industry in Brazil: Challenges and Trends*. 2015. [Online]. Disponível em: <https://goo.gl/v2BENs>.