Aprendizado por Reforço Profundo com Redes Recorrentes Aplicado à Negociação do Minicontrato Futuro de Dólar

Deep Reinforcement Learning with Recurrent Neural Networks applied to the trading of Mini US Dollar Future

Jonathan Kenji Kinoshita[1]
Douglas De Rizzo Meneghetti[1]
Reinaldo Augusto da Costa Bianchi[1]

[1]Departamento de Engenharia Elétrica, Centro Universitário FEI
Av. Humberto de Alencar Castelo Branco, 3972-B - Assunção
São Bernardo do Campo - SP, 09850-901

Abstract. *Recently, there has been considerable increase in the usage of machine learning applied in the financial market, primarily for trade stocks, in an attempt to forecast the movement of their prices. The proposal of this research is investigate an intelligent trade system for Mini US Dollar Future, based on Deep Recurrent Q-Network, a technique based in training a recurrent neural network to solve partially observable reinforcement learning problems. The training is based on the historical data from the asset and the agent performs three actions: buying, holding and selling the asset, with the goal of maximizing the return. Experiments made showed that the proposed system performs well, achieving a better result than using the Buy and Hold and the traditional DQN.*
Keywords: Deep Reinforcement Learning. Convolutional Neural Network. Recurrent Network. Long Short-Term Network. Deep Recurrent Q-Network. Future Market.

Resumo. *Recentemente, houve um aumento considerável no uso de técnicas de aprendizado de máquina no mercado financeiro, principalmente para negociação de ações, na tentativa de prever preços futuros. O objetivo desse projeto é investigar a aplicação do aprendizado por reforço em um sistema de negociação inteligente do minicontrato futuro de dólar, usando uma* Deep Recurrent Q-Network*, uma técnica baseada no treinamento de uma rede recorrente para resolução de problemas de aprendizado por reforço parcialmente observáveis. O treinamento foi baseado em uma base da dados históricos do ativo e o agente realizou três ações: comprar, vender ou manter o ativo, sempre visando maximizar o retorno financeiro. Os experimentos realizados indicam que o sistema pode alcançar desempenho superior à estratégia de* Buy and Hold *e a tradicional DQN.*
Palavras-chave: Aprendizado por Reforço Profundo. Redes Neurais Convolucionais. Redes Neurais Recorrentes. Long Short-Term Network. Deep Recurrent Q-Network. *Mercado Futuro.*

1. Introdução

O mercado financeiro é considerado o coração da economia mundial, no qual bilhões de dólares são negociados todos os dias. A natureza caótica, estocástica e não-

linear do mercado [Hsieh 1991], assim como sua evolução ao longo do tempo, fazem da predição de seu comportamento futuro uma tarefa desafiadora.

Um ativo financeiro é tudo que tem valor agregado e pode ser negociado no mercado financeiro, não sendo um bem ou mercadoria e não possuindo representação contratual além da documentação que o define. Existem atualmente duas vertentes para o estudo de um ativo. A análise fundamentalista visa o valor agregado a longo prazo. Já a análise técnica se baseia no movimento dos preços, seu histórico e seus possíveis valores futuros [Lo et al. 2000].

As negociações de ações brasileiras e outros ativos financeiros, como moedas e commodities, são realizadas na B3, a bolsa de valores oficial do Brasil. Dentro do ambiente da B3 está o mercado futuro, onde são negociados os contratos futuros (conhecidos como contratos cheios), que são acordos de compra e venda de ativos em uma data futura, ou seja, o investidor negocia hoje a expectativa do preço do ativo no futuro. Com o objetivo de atrair mais investidores para o mercado futuro, foram criados os minicontratos, que correspondem a 20% de um contrato cheio, os quais permitem que a margem de garantia necessária para operar esse ativo seja menor que o contrato cheio correspondente.

Um dos minicontratos negociados é o de dólar. O tamanho dele equivale a US$ 10000,00, com a sua cotação expressa em reais por R$ 1000,00. Como exemplo, caso o dólar valha R$ 5,00, a cotação do minicontrato será de R$ 5000,00 e representará a compra ou venda de R$ 50000,00. A cotação dos minicontratos de dólar varia diariamente e é representada por pontos.

Um ponto representa a diferença na cotação do minicontrato e equivale a R$ 10,00 por minicontrato, sendo esse valor escalonável, a depender do número de minicontratos negociados. Por exemplo, caso dois minicontratos sejam operados, cada ponto equivalerá R$ 20,00.

Técnicas de aprendizado de máquina têm mostrado sua capacidade em predizer o comportamento futuro dos preços de ativos. Redes neurais artificiais e máquinas de vetores de suporte são algumas das técnicas utilizadas com esse objetivo [Guresen et al. 2011, Kara et al. 2011].

Os avanços em aprendizado profundo da última década introduziram nossas topologias de redes neurais capazes de aprender automaticamente representações úteis para os dados, de acordo com a função de erro que tentam minimizar. Já as técnicas de aprendizado por reforço se concentram na solução de problemas de tomada de decisão sequencial, ao passo que não são focadas no aprendizado de representações ou na resolução de problemas cuja dimensionalidade dos dados seja alta.

À combinação dos dois campos dá-se o nome de Aprendizado por Reforço Profundo, cujas técnicas visam a criação de sistemas de tomadas de decisão sequenciais em ambientes de alta dimensionalidade de dados. Diversas pesquisas recentes aplicam técnicas de Aprendizado por Reforço Profundo na negociação de ativos [Li et al. 2020, Carapuço et al. 2018, Lee et al. 2019, Xiong et al. 2018].

Pela natureza do mercado financeiro, este dificilmente será totalmente explícito nos estados, para se tomar uma decisão é necessário um conhecimento mais amplo do ambiente, podendo assim ser considerado um Processo de Decisão Markoviano Par-

cialmente Observável (POMDP, do inglês *Partially Observable Markov Decision Process*) [Hausknecht and Stone 2017]. O objetivo deste trabalho foi investigar o funcionamento de um sistema de negociação de ativos, para realizar operações de compra e venda, baseado na Rede Q Profunda Recorrente (DRQN, do inglês *Deep Recurrent Q-Network*), a qual usa uma combinação entre Redes Neurais Recorrentes, do tipo *Long Short-Term Memory* (LSTM) com Redes Neurais Convolucionais (CNN, do inglês *Convolutional Neural Networks*).

2. Fundamentação Teórica

A área de aprendizado por reforço busca propor abordagens computacionais para solucionar problemas formulados como processos de tomada de decisões sequenciais [Watkins 1989]. Uma dessas formulações é o Processo de Decisão de Markov (MDP, do inglês *Markov Decision Process*), no qual o ambiente é descrito através da tupla $< S, A, P, R >$, onde:

- Espaço de estados (S): Diversos atributos são usados para representar os estados. No mercado financeiro, na maioria dos casos, é utilizado o histórico dos preços de alta, baixa, abertura, fechamento e volume de negócios dos papeis, juntamente com indicadores técnicos relacionados [Fischer and Krauss 2018].
- Espaço de Ações (A): São todas as ações que o agente pode realizar no instante de tempo t
- Matriz de Probabilidade (P): É a probabilidade de o agente ir para o estado S_{t+1}, dado S_t e A_t
- Função Recompensa (R): É a recompensa recebida pela mudança no ambiente gerada pela ação tomada pelo agente.

Em um MDP, supõe-se que todo os estados possuem a propriedade de Markov, a qual dita que um s_t captura todas as informações relevantes de seu histórico. Portanto, as funções r_{t+1} e s_{t+1} dependem somente do estado atual (s_t) e da ação (a_t), independendo de estados e ações passadas. Além disso, em um MDP as ações tomadas pelo agente não influenciam apenas a recompensa imediata, mas também, as futuras ações, e portanto as recompensas futuras. O agente, quando toma uma ação, deve aprender a equilibrar a recompensa imediata com a futura.

Em aprendizado por reforço, um agente inteligente aprende em um MDP através da interação com o ambiente, como exemplificado na Figura 1. Em cada instante de tempo t, o agente observa o estado atual s_t e escolhe e executa uma ação a_t, observando em seguida uma recompensa $r_{t+1} = r(s_t, a_t)$ e o estado seguinte $s_{t+1} = \delta(s_t, a_t)$, onde as funções δ e r são intrínsecas do ambiente e não necessariamente conhecidas pelo agente.

Em alguns algoritmos de aprendizado por reforço envolvem a otimização de uma função valor. A função valor é uma função dos estados e quantifica para o agente o quanto bom é estar em determinado estado. A equação 1 expressa a função valor de um estado s sob uma politica π.

$$V_\pi(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right] \tag{1}$$

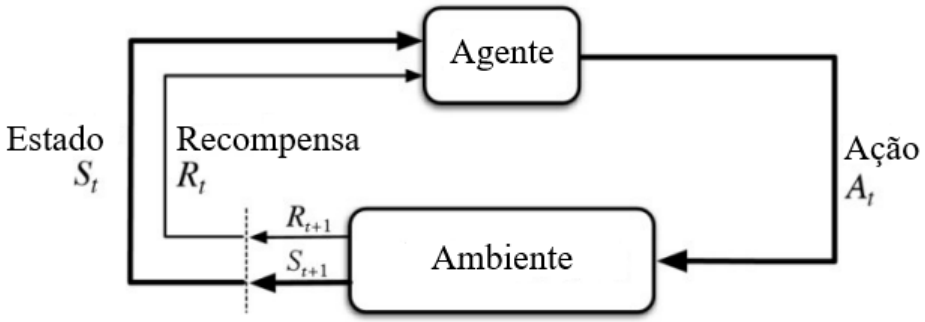

Figura 1. Interação entre agente e ambiente no Aprendizado por Reforço [Sutton and Barto 1998]

Similar à função valor, tem-se a função ação-valor, o qual para uma politica π expressa o valor de uma ação a em um estado s sob uma politica π, e que é expressa pela equação 2.

$$Q_\pi(s,a) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right] \tag{2}$$

Onde $0 < \gamma < 1$ é uma constante que determina o valor relativo das recompensas futuras versus imediatas.

Em problemas do mundo real, nem sempre o estado fica totalmente explícito para o agente, impossibilitando o estado (s) ser considerado de Markov, sendo assim, esse ambiente não pode ser modelado como uma MDP. Em vista disso, tem-se o POMDP que considera que o estado (s) representa apenas uma parte do ambiente, sendo então denominado como observação [Hausknecht and Stone 2017]. O conjunto que define o POMDP passa a ser $< S, A, P, R, \Omega, O >$, onde S,A,P,R são similares aos do MDP, e o agente recebe uma observação $o \in \Omega$, que é gerada pelo estado s seguindo a distribuição de probabilidades $o \sim O(s)$.

2.1. *Deep Q-Network*

Em aplicações práticas de aprendizado por reforço, ambientes interessantes podem ter representações de estados complexas. Para simplificar o aprendizado em tais ambientes, pode-se empregar redes neurais profundas, capazes de aprender representações úteis para os dados de acordo com a função de erro a ser minimizada.

A Rede Q Profunda (DQN, do inglês *Deep Q-Network*) [Mnih et al. 2015] mostrou resultados bem expressivos quando aplicada em vários jogos de Atari 2600, tendo como entrada apenas os pixels da tela. Ela é composta por três camadas de redes neurais convolucionais seguida por duas camadas de redes totalmente conectadas, sendo que a de saída do modelo contém as ações possíveis.

Pelo fato de que com o uso de uma rede neural artificial para aproximar os valores da função Q (ação-valor) o processo de aprendizado se tornava instável, utilizou-se de duas redes neurais artificiais além da adoção da memória de repetição.

O uso de uma segunda rede neural artificial chamada de rede alvo para a geração

dos valores $Q(s', a')$, além de sua atualização de parâmetros ser periódica, diferentemente da outra rede, denominada rede de política, faz com que o modelo se torne mais estável.

Para conseguir convergir, a DQN usa como estratégia de exploração do espaço de estados uma política gulosa (ϵ-*greedy*), que faz com que o agente escolha uma ação de acordo com a probabilidade ϵ, podendo assim, ser uma ação aleatória ou baseada na função Q. Ao decorrer dos episódios, o valor de ϵ diminui gradativamente, resultando em maiores chances da ação ser fundamentada na função Q. Estratégia de exploração é necessária no aprendizado por reforço para o agente conseguir conhecer novas possibilidade de ações nos estados fornecidos.

A cada interação com o ambiente, o agente recebe o estado s_t, e de acordo com a probabilidade ϵ executa uma ação a_t, recebendo a recompensa r_t e o próximo estado s_{t+1}, formando assim, uma transição $e_t = (s_t, a_t, r_t, s_{t+1})$ que é armazenado na memória de repetição ($\mathcal{D}$). As transições são amostradas aleatoriamente da memória de repetição durante o treinamento. A equação 3 mostra a função de erro durante o treinamento.

$$L_i(\theta_i) = E_{(s_t,a_t,r_t,s_{t+1})} \sim \mathcal{D}\left[(r + \gamma max_a Q(s', a'; \theta^-) - Q(s, a; \theta_i))^2\right] \tag{3}$$

A DQN leva em consideração o maior valor da função Q em todas as ações, mesmo não sendo a melhor ação a ser tomada dado o estado, ocasionando em uma superestimação nos valores Q. Para contornar tal problema desenvolveu-se a Rede Q Profunda Dupla (DDQN, do inglês *Double Deep Q-Network*) [van Hasselt et al. 2015].

2.2. *Deep Recurrent Q-Network*

Para lidar com ambientes onde os estados (s) apresentam informações incompletas, foi desenvolvida arquitetura da Rede Q Profunda Recorrente (DRQN, do inglês *Deep Recurrent Q-Network*) [Hausknecht and Stone 2017]. A estrutura da rede neural artificial e o algoritmo são praticamente os mesmos dos apresentados na DQN [Mnih et al. 2015], exceto que, há a substituição da primeira camada totalmente conectada por uma camada do tipo *Long Short-Term Network* [Hochreiter and Schmidhuber 1997]. Sendo assim, como as redes neurais do tipo recorrentes possuem capacidade de memória em relação às suas entradas anteriores, elas conseguem uma melhor convergência dos valores Q, dado uma sequência de observações. E para cooperar com tal resultado, ao invés de usar transições aleatórias no treinamento é utilizado episódios com transições sequenciais

3. Trabalhos Relacionados

Propostas originalmente para a resolução de problemas de visão computacional, CNNs também têm demonstrado potencial ao serem aplicadas em séries temporais, com o propósito de extrair atributos dos dados de entrada. Em [Zheng et al. 2014], CNNs obteviram melhores resultados que um Perceptron Multi-Camadas (MLP, do inglês *Multilayer Perceptron*) e um classificador *K-Nearest Neighbors* na classificação de séries temporais multivariável.

As LSTMs [Gers et al. 2000] são muito usadas no processamento de sequências de dados, pois elas possuem a capacidade de armazenar informações de longos períodos de tempo, e resolvem o problema de desvanecimento de gradiente. Muitos trabalhos

atuais aplicaram elas em séries temporais financeiras e mostraram um desempenho promissor [Lim et al. 2019].

Recentemente, o Aprendizado por Reforço Profundo vem sendo muito empregado no mercado financeiro, demonstrando resultados expressivos. A eficácia das LSTM como modelo de rede neural foi demonstrada nos algoritmos da DQN, Gradiente de Políticas (PG, do inglês *Policy Gradient*) [Williams 1992], e *Advantage Actor-Critic* (A2C) [Mnih et al. 2016] em um sistema de negociação de ativos [Zhang et al. 2020]. Em [Théate and Ernst 2021] um sistema denominado *Trading Deep Q-Network* (TDQN) é proposto, baseado na DDQN que demonstrou um alto potencial de ganhos, sendo que em alguns papeis superou varias estratégias, como o Buy and Hold. Carapuço et al. (2018) apresentaram um modelo fundamentado na DQN, salientando a possibilidade de altos ganhos financeiros com o Aprendizado por Reforço Profundo, e que um dos maiores problemas da DQN aplicado no mercado financeiro é o fato do ambiente não ser estacionário e cheio de ruido, dificultando a generalização do modelo treinado.

As DQN, DDQN e DDQN Duelistas também demonstraram capacidade de tomar rápidas decisões de acordo com o movimento do mercado em [Li et al. 2020], os quais declaram que, dentre os algoritmos testados, a DQN teve um desempenho superior.

4. Materiais e Métodos

O projeto foi implementado em linguagem de programação Python 3.8, em conjunto com a biblioteca Tensorflow[1]. Os modelos foram treinados e testados em um computador com processador AMD Ryzen 5 3600X, 32 GB de RAM e placa de video NVIDIA GTX 1660, em sistema operacional Windows 10.

Considerando o mercado de ativos como um POMDP, pois os os estados s representam apenas uma parte do ambiente, este trabalho consiste em utilizar um sistema fundamentado na *Deep Recurrent* Q*-Network*, para realizar operações de compra e venda do minicontrato futuro de dólar, visando sempre o maior retorno financeiro.

Em um instante de tempo t, as seguintes informações são observáveis pelo agente:

- preço de abertura do minicontrato do dólar;
- preço na alta do minicontrato do dólar;
- preço na baixa do minicontrato do dólar;
- preço de fechamento do minicontrato do dólar;
- volume negociado de minicontrato do dólar;
- preço de fechamento do minicontrato do dólar;
- volume negociado de minicontrato de índice;
- posicionamento do agente, em forma ortogonalizada:
  - comprado ([1,0,0]): no qual o agente compra o ativo com a expectativa de lucrar na sua venda;
  - vendido ([0,1,0]), no qual o agente espera lucrar com a descida do preço do ativo;
  - fora do mercado ([0,0,1]), onde o agente não possui nenhum ativo em posse.
- valor acumulado com a operação (compra/venda) em aberto.

[1]https://www.tensorflow.org/, acessado em dezembro, 2021

A presença do preço de fechamento e volume negociado do minicontrato de índice se justifica, uma vez que, em alguns momentos, os minicontratos de índice e dólar apresentam movimentos antagônicos em seus preços, também há a presença do lucro ou prejuízo do agente referente à sua operação em aberto (comprado ou vendido).

A representação de um estado s_t incorpora os 11 valores das variáveis supracitadas nos últimos 30 instantes de tempo, resultando em uma representação matricial de dimensões 30×11.

Adotou-se um espaço de ações discreto de 3 ações, sendo cada ação aplicada diretamente sobre o papel As ações são: compra, venda e manter posição atual. Se duas ações subsequentes forem iguais (e.g. comprar - comprar), a segunda ação é considerada como manter (comprar - manter). O agente pode operar comprado ou vendido.

Como o objetivo é ter o maior retorno financeiro, a função recompensa é apenas o lucro ou prejuízo obtido na operação concluída, subtraindo-se as taxas transacionais pré-determinadas, que somadas correspondem à U$$ 0,22 (taxas de registro e emolumentos)[2]. O agente só recebe a recompensa quando sua posição é fechada, ou seja, em caso de compra, o agente só ganha o reforço no momento da venda e o valor da recompensa será o valor do preço de venda, subtraindo-se o de compra e descontando-se as taxas. Vale ressaltar que, para os treinamentos e testes do modelo, não foi considerada a taxa do imposto de renda que incide sobre o lucro gerado no mês.

4.1. Arquitetura da Rede

Este trabalho empregou uma rede neural composta por CNNs, LSTM e MLP. Optou-se pela adoção das convoluções 2D para captar as inter-relações dos atributos. Cada camada das redes convolucionais possui 32 filtros, cada um com dimensões 3×3. A função de ativação usada nelas foi a função linear retificada, ReLU [LeCun et al. 2015]. A LSTM foi composta de 156 unidades, seguida de uma camada totalmente conectada com 32 unidades com função de ativação ELU [Clevert et al. 2016]. Como o modelo tem apenas três ações, a saída da rede neural contém apenas 3 neurônios sem função de ativação não-linear na última camada. A figura 2 ilustra o modelo da rede neural usado na DRQN.

Figura 2. Estrutura da rede neural

4.2. Base de Dados e Treinamento

Nos experimentos realizados, o modelo foi treinado e testado usando os preços históricos do minicontrato de dólar e índice, intervalados de 15 minutos, sendo assim, cada ação realizada pelo agente tem um hiato de 15 minutos. Os dados foram retirados da plataforma Metatrader 5[3] e englobam o período de 27/09/2019 até 09/10/2020. A tabela 1 apresenta um trecho da base de dados obtida após a junção dos preços históricos de ambos minicontratos.

[2]https://www.b3.com.br/pt_br/produtos-e-servicos/tarifas/listados-a-vista-e-derivativos/moedas/tarifas-de-dolar-dos-estados-unidos/futuros-de-dolar/, acessado em dezembro, 2021)

[3]https://www.metatrader5.com/pt, acessado em dezembro, 2021

Tabela 1. Estrutura dos dados utilizados

Data	Tempo	Abertura	Alta	Baixa	Fechamento	Volume	Fechamento (Índice)	Volume (Índice)
03/08/2020	09:00	5235,0	5242,5	5216,5	5239,5	137665	103840	237021
03/08/2020	09:15	5239,5	5246,0	5234,0	5241,5	96845	103955	221627
03/08/2020	09:30	5242,0	5252,0	5238,5	5250,0	88794	103885	178212
03/08/2020	09:45	5250,0	5271,0	5249,0	5271,0	153845	103795	192113
03/08/2020	10:00	5271,0	5293,0	5267,0	5291,5	183569	103680	480949

A tabela 2 exibe a divisão da base de dados para os experimentos executados. Foi empregado o sistema de treinamento de janela deslizante, que seguiu a proporção 14:4, considerando-se 14 semanas de treinamento para quatro de teste. Após o primeiro experimento, os dados utilizados para treinamento e teste se deslocam quatro semanas à frente para realização do experimento seguinte. O treinamento possuiu, em média, 2400 dados enquanto o teste 700. O tamanho das janelas foi escolhido de forma a se balancear o número de experimentos realizados com o extenso tempo de treinamento dos modelos, uma vez que, para cada janela, um novo modelo foi treinado, com tempo médio de conclusão de 50 a 60 horas.

Tabela 2. Plano de testes

Experimento	Treinamento	Teste
1	27/09/2019 - 03/01/2020	06/01/2020 - 31/01/2020
2	25/10/2019 - 31/01/2020	03/02/2020 - 28/02/2020
3	22/11/2019 - 28/02/2020	02/03/2020 - 27/03/2020
4	20/12/2019 - 27/03/2020	30/03/2020 - 24/04/2020
5	17/01/2020 - 24/04/2020	27/04/2020 - 22/05/2020
6	14/02/2020 - 22/05/2020	25/05/2020 - 19/06/2020
7	13/03/2020 - 19/06/2020	22/05/2020 - 17/07/2020
8	09/04/2020 - 17/07/2020	20/07/2020 - 14/08/2020
9	08/05/2020 - 14/08/2020	17/08/2020 - 11/09/2020
10	05/06/2020 - 11/09/2020	14/09/2020 - 09/10/2020

Na fase inicial do treinamento, os valores das variáveis de estado relacionadas ao posicionamento do agente e ao valor acumulado com a operação em aberto são mantidos como $[0; 0; 1]$ (manter) e zero, respectivamente, até o agente completar as 30 observações necessárias para iniciar suas interações com o ambiente. Uma etapa de pré-processamento envolve normalizar as variáveis de estado reais dentro do intervalo $[0; 1]$.

Uma vez completo e pré-processado, o estado é enviado ao agente para a seleção da ação. O ambiente transiciona para um próximo estado e o envia, junto à recompensa, ao agente. Esse processo se repete até o final da base de dados do experimento, completando assim um episódio. Cada experimento foi treinado por 4500 episódios.

A cada interação com o ambiente, a transição (s_t, a_t, r_t, s_{t+1}) é armazenada na memória de repetição, sendo 128 transições amostradas a cada n passos para treinamento da rede neural, assim como descrito por [Huang 2018]. A DRQN empregada faz uso de uma rede de política, que tem seus parâmetros atualizados constantemente, e uma rede alvo, que tem seus parâmetros atualizados ao final de cada episódio. Para garantir uma exploração adequada do espaço de estados foi usado o ϵ-*greedy*, com ϵ variando de 1 até

0,1.

O otimizador empregado foi o Adam [Kingma and Ba 2014] com taxa de aprendizagem inicial de 0,00025, $\gamma = 0,99$ e *minibatch* de 128 transições sequenciais, diferentemente da DRQN de [Hausknecht and Stone 2017] que utiliza n episódios completos com transições sequenciais. Além disso, a amostragem da memória de repetição e sua consequente descida de gradiente são realizados em um intervalo de 4 instantes de tempo. Fatos estes que resultam em um menor custo computacional e consequentemente em um menor tempo de treinamento. A arquitetura da rede neural e os hiperparâmetros foram definidos empiricamente, após a realização de vários testes. O modelo apresentado foi aquele que apresentou o melhor desempenho.

Em cada episódio (de treinamento ou teste), o agente começa com a sua carteira zerada, ou seja, possuindo zero pontos de dólar. Devido as experimentos considerarem a compra ou venda de apenas um minicontrato, cada ponto equivale à R$ 10,00. Toda vez que o agente completa uma operação (lucrativa ou não), o valor adquirido é somado ao seu portfólio.

O valor final do portfólio foi comparado com a estratégia do *Buy and Hold* (BH), na qual há a compra do ativo com a expectativa de sua valorização a longo prazo, e com um modelo DQN, que foi treinado identicamente à DRQN, sendo as únicas diferenças existentes: a rede neural, onde há a substituição da camada LSTM por uma camada totalmente conectada, mantendo as mesmas 156 unidades; e o *minibatch*, que ao invés de ser amostrado sequencialmente, ele é amostrado aleatoriamente. Para a estratégia de *Buy and Hold*, foi considerada a compra de um minicontrato de dólar no primeiro momento de teste e a sua venda no final do período em questão.

Para a realização do treinamento e teste dos experimentos da tabela 2, adotou-se a metodologia da janela deslizante, ilustrada na figura 3, na qual o modelo é treinado em uma janela de dados no tempo e testado numa janela menor imediatamente subsequente à de treinamento. Com o decorrer dos experimentos, as janelas deslizam cronologicamente sobre a base de dados.

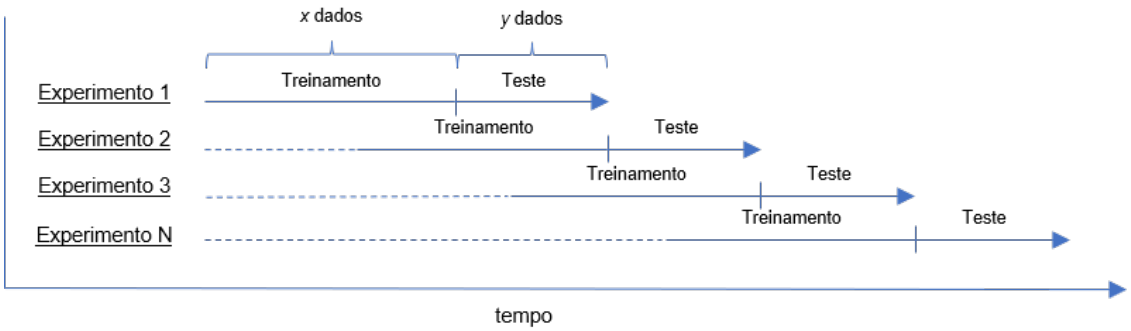

Figura 3. Janela deslizante

5. Resultados

A tabela 3 apresenta as quantidades finais de pontos presentes em seus portfólios, utilizando-se de ambas estratégias em todos os experimentos referentes à tabela 2.

Tabela 3. Resultado dos experimentos

Experimento	*Buy and Hold*	DQN	DRQN
1	224	123	-40,8
2	221	-150	134,5
3	603,5	-363	678,4
4	468,5	287	247,3
5	-26	306	259,6
6	-207	-137	-113,6
7	108,5	72	247
8	52,5	539	325,2
9	-95,5	-139	415,7
10	224,5	111	-114,2
Soma	1574	649	2039,1
Imposto de Renda (IR)	236,1 (-15%)	129,8 (-20%)	407,86 (-20%)
Soma - IR	1337,9	519,2	1631,4

Somando-se todos os pontos obtidos em cada período de teste, os pontos obtidos pela DRQN foram 29,55% superiores que a estratégia do *Buy and Hold*. Em comparação com a DQN, a DRQN foi melhor em 214,22%, indicando melhora no desempenho devido ao emprego de camadas recorrentes e treinamento com estados que incorporam informações históricas de interação com o ambiente.

Em um exemplo de *Buy and Hold* que realizasse a compra de um minicontrato no primeiro passo de teste do experimento 1 e sua venda ao final do experimento 10, o lucro resultante seria de 1477 pontos ao longo de 40 semanas. Neste caso, a DRQN ainda acumularia 38,05% mais pontos que o *Buy and Hold*.

Ao se considerar a incidência do imposto de renda nos resultados de todos os métodos, a DRQN continua sendo o método que acumula mais pontos, mesmo tendo maior incidência de impostos devido à prática de *day trading* Considerando-se a incidência do imposto de renda, mesmo sendo que no *Buy and Hold* a taxa seja de 15% e nos modelos DQN e DRQN de 20%, a DRQN conseguiu ter um desempenho superior em relação à *Buy and Hold* e a DQN.

Nos experimentos em que a DRQN obteve menos retorno em relação ao *Buy and Hold*, uma possível explicação é a discrepância entre os movimentos do dólar presentes nos conjuntos de treinamento e teste do modelo. Mesmo assim, a DRQN conseguiu bons resultados quando estourou a pandemia mundial decorrente ao coronavírus, em março de 2020 (experimento 3), superando, mesmo que pouco, o *Buy and Hold*. Apesar do período de testes ser curto (aproximadamente 10 meses), a DRQN experienciou diversos cenários e movimentos na cotação do dólar e conseguiu, na maioria das vezes, lidar com as oscilações de preço do ativo.

6. Conclusão

Este trabalho investigou a aplicação da *Deep Recurrent Q-Network* na negociação do minicontrato futuro de dólar, objetivando o maior retorno financeiro possível. Mesmo considerando todas as taxas envolvidas na negociação do ativo e a natureza intrínseca do

mercado futuro, o modelo apresentado conseguiu encontrar uma política rentável.

Quando comparada à estratégia de *Buy and Hold* e a um modelo DQN, a DRQN apresentou resultados superiores. Em trabalhos futuros, propõe-se analisar os efeitos de outras proporções da janela deslizante, assim como o uso do algoritmo da DDQN, ao invés da DQN, na implementação da DRQN.

Referências

Carapuço, J., Neves, R., and Horta, N. (2018). Reinforcement learning applied to Forex trading. *Applied Soft Computing*, 73:783–794.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (elus).

Fischer, T. and Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural Computation*, 12:2451–2471.

Guresen, E., Kayakutlu, G., and Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, 38(8):10389–10397.

Hausknecht, M. and Stone, P. (2017). Deep Recurrent Q-Learning for Partially Observable MDPs. *arXiv:1507.06527 [cs]*. arXiv: 1507.06527.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hsieh, D. A. (1991). Chaos and Nonlinear Dynamics: Application to Financial Markets. *The Journal of Finance*, 46(5):1839–1877.

Huang, C. Y. (2018). Financial trading as a game: A deep reinforcement learning approach.

Kara, Y., Boyacioglu, M. A., and Baykan, O. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications*, 38(5):5311–5319.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–44.

Lee, J., Kim, R., Koh, Y., and Kang, J. (2019). Global Stock Market Prediction Based on Stock Chart Images Using Deep Q-Network. *IEEE Access*, 7:167260–167277.

Li, Y., Ni, P., and Chang, V. (2020). Application of deep reinforcement learning in stock trading strategies and stock forecasting. *Computing*, 102(6):1305–1322.

Lim, B., Zohren, S., and Roberts, S. (2019). Enhancing time-series momentum strategies using deep neural networks. *The Journal of Financial Data Science*.

Lo, A. W., Mamaysky, H., and Wang, J. (2000). Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation. *The Journal of Finance*, 55(4):1705–1765.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: an introduction*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass.

Théate, T. and Ernst, D. (2021). An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173:114632.

van Hasselt, H., Guez, A., and Silver, D. (2015). Deep Reinforcement Learning with Double Q-learning.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Xiong, Z., Liu, X.-Y., Zhong, S., Yang, H., and Walid, A. (2018). Practical Deep Reinforcement Learning Approach for Stock Trading. *arXiv:1811.07522 [cs, q-fin, stat]*. arXiv: 1811.07522.

Zhang, Z., Zohren, S., and Roberts, S. (2020). Deep Reinforcement Learning for Trading. *The Journal of Financial Data Science*, 2(2):25–40.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., and Zhao, J. L. (2014). Time series classification using multi-channels deep convolutional neural networks. In *WAIM*.