CENTRO UNIVERSITÁRIO FEI

FELIPE ALBERTO CAPATI

APRENDIZADO POR REFORÇO PROFUNDO MULTIAGENTE APLICADO A NEGOCIAÇÃO DE ATIVOS DE MERCADO FINANCEIRO

São Bernardo do Campo

2021

FELIPE ALBERTO CAPATI

APRENDIZADO POR REFORÇO PROFUNDO MULTIAGENTE APLICADO A NEGOCIAÇÃO DE ATIVOS DE MERCADO FINANCEIRO

Dissertação de Mestrado, apresentada ao Centro Universitário da FEI para obtenção do título de Mestre em Engenharia Elétrica. Orientado pelo Prof. Dr. Reinaldo A. C. Bianchi.

São Bernardo do Campo

2021

Capati, Felipe Alberto.
Aprendizado Por Reforço Profundo Multiagente Aplicado a Negociação de Ativos de Mercado Financeiro / Felipe Alberto Capati. São Bernardo do Campo, 2021.
102 f. : il.

Dissertação - Centro Universitário FEI.
Orientador: Prof. Dr. Reinaldo Augusto da Costa Bianchi.

1. Multi Agent Reinforcement Learning. 2. reinforcement learning. 3. maddpg. 4. stocks. 5. Aprendizado por Reforço Multiagente. I. Bianchi, Reinaldo Augusto da Costa, orient. II. Título.

Elaborada pelo sistema de geração automática de ficha catalográfica da FEI com os dados fornecidos pelo(a) autor(a).

centro universitário FEI

APRESENTAÇÃO DE DISSERTAÇÃO
ATA DA BANCA EXAMINADORA

Programa de Pós-Graduação Stricto Sensu em Engenharia Elétrica

Mestrado

PGE-10

Aluno: Felipe Alberto Capati **Matrícula:** 119121-2

Título do Trabalho: Aprendizado por reforço profundo multiagente aplicado a negociação de ativos de mercado financeiro.

Área de Concentração: Inteligência Artificial Aplicada à Automação e Robótica

Orientador: Prof. Dr. Reinaldo Augusto da Costa Bianchi

Data da realização da defesa: 05/04/2021

ORIGINAL ASSINADA

Avaliação da Banca Examinadora:

A banca foi realizada no dia 05 de abril de 2021 às 09:00 horas, e se iniciou com a apresentação do aluno, que foi muito boa, e seguiu para a arguição, onde o aluno respondeu a todas as questões de forma adequada demonstrando conhecimento pleno do tema. Foram sugeridas melhorias em relação ao texto para a versão final. A aprovação foi por unanimidade.

São Bernardo do Campo, 05 / 04 / 2021.

MEMBROS DA BANCA EXAMINADORA	
Prof. Dr. Reinaldo Augusto da Costa Bianchi	Ass.: _____
Prof. Dr. Guilherme Alberto Wachs Lopes	Ass.: _____
Prof.ª Dr.ª Anna Helena Reali Costa	Ass.: _____

A Banca Julgadora acima-assinada atribuiu ao aluno o seguinte resultado:

APROVADO ☒ REPROVADO ☐

VERSÃO FINAL DA DISSERTAÇÃO

APROVO A VERSÃO FINAL DA DISSERTAÇÃO EM QUE FORAM INCLUÍDAS AS RECOMENDAÇÕES DA BANCA EXAMINADORA

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Carlos Eduardo Thomaz

Dedico este trabalho a Deus e aos meus guias espirituais que me forneceram força para lutar e continuar; ao meu pai João Carlos Capati que sempre acreditou em mim e me incentivou a lutar pelos meus sonhos; a minha falecida mãe Solange Cristina Vasconcelos Capati, incentivando-me a estudar para mudar a realidade em que vivia e aos meus irmãos que me apoiaram para ser possível chegar a essa etapa da minha vida.

AGRADECIMENTOS

Gostaria de agradecer à CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), instituição pública que investe em pesquisa científica e proporciona o aumento do capital intelectual nacional, ao Centro Universitário FEI, instituição que cursei a graduação, e hoje o mestrado, devo a essa reconhecida instituição o patamar ao qual cheguei: tornei-me um profissional e um pesquisador. Sou muito grato a ambas as instituições por investirem na minha capacitação e acreditarem no meu trabalho, a ponto de disponibilizarem uma bolsa de estudos para o desenvolvimento do meu Mestrado. Sem essa bolsa, eu não estaria cursando o mestrado hoje e esta pesquisa não seria desenvolvida, não contribuiria com um grão de areia na imensidão do conhecimento científico, não levando assim o nome da CAPES e da FEI ao mercado de trabalho.

Sou muito grato a minha família que me apoiou em todos os meus momentos de dificuldade e de felicidade. Um agradecimento especial a minha falecida mãe, que desde muito pequeno me incentivou a estudar e me dedicar para mudar a realidade em que vivia, mesmo partindo deixou valores importantes que levo comigo sempre; ao meu pai, ou melhor, meu pamãe (meio pai e meio mãe) que me ajudou e me incentivou a lutar pelos meus sonhos; e aos meus irmãos que me apoiaram para ser possível chegar nessa etapa da minha vida.

Gostaria de agradecer a todos os meus amigos e colegas de turma que compartilharam conhecimento comigo e que, diretamente ou indiretamente, contribuíram para o meu fortalecimento nesse período de estudo. Por fim, e não menos importante, gostaria de agradecer ao meu orientador Reinaldo Bianchi que me instruiu neste trabalho e lapidou o meu conhecimento a ponto de ser possível desenvolvê-lo.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

“O estudo em geral, a busca da verdade e da beleza são domínios em que nos é consentido ficar crianças toda a vida”

Albert Einstein

“Por vezes sentimos que aquilo que fazemos não é senão uma gota de água no mar. Mas o mar seria menor se lhe faltasse uma gota”

Madre Teresa de Calcuta

RESUMO

O presente trabalho tem como motivação principal o estudo de modelos de aprendizado por reforço *multi-agent*, comumente utilizados quando o problema é episódico e a dinâmica do sistema é complexa de ser descrita analiticamente, aplicado à negociação de ações na bolsa de valores, desenvolvendo um sistema cooperativo composto por um agente que representa a força de compra dos ativos, e outro agente a força de venda. Os agentes devem interagir para decidir quanto será comprado ou vendido, de modo a optimizar o lucro obtido, criando o que foi denominado de: "gestor de carteira de investimentos". Ao término do desenvolvimento foram feitas análises do modelo *single-agent* e *multi-agent*, avaliações do impacto da modelagem do ator-crítico utilizando redes recorrentes e propostas de melhoria do projeto desenvolvido. O modelo *multi-agent* obteve resultados positivos, em sua maioria, sendo superiores ao *buy-and-hold* em mais de 88% no experimento executado com o ativo ITSA4, porém os resultados de forma geral não são suficientes para a construção de um modelo comercial. Ao término do projeto foram propostas melhorias e possíveis trabalhos futuros com o intuito de auxiliar o desenvolvimento da área.

Palavras-chave: Aprendizado por Reforço Multiagente (MARL), RL, MARL, MADDPG, Ações

ABSTRACT

The main motivation of the present academic work is the study of multi-agent reinforcement learning models, commonly used when the problem is episodic and the dynamic of the system is complex to be described analytically, applied to stock trading on the stock exchange, developing a cooperative system composed of one agent who represents the buying force of the stocks, and the other agent the sale force. The agents must interact to decide how much will be bought or sold, in order to optimize the profit obtained, creating what was called: investment portfolio manager". At the end of the development, analyzes of the single-agent and multi-agent model were made, evaluations of the impact of using recurrent networks in the actor-critic model and proposals for improvement of the developed project. The multi-agent model achieved positive results, mostly, being better then buy-and-hold by more than 88% in the experiment performed with the stock ITSA4, however the results in general are not enough to build a business model. At the end of the project, improvements and possible future work were proposed in order to assist the development of the area.

Keywords: Multi Agent Reinforcement Learning (MARL), RL, MADDPG, Stocks

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS

A2C	Advantage Actor Critic
AD	Chaikin Accumulation Distribution
ADOSC	Chaikin Accumulation Distribution Oscilator
ADX	Average Directional Movement Index
AI	Artificial Intelligence
ARCH	Autoregressive Conditional Heteroskedasticity
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
Bi-LSTM	Bi-Directional Long Short Term Memory
BOVESPA	Bolsa de Valores de Sao Paulo
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CCI	Commodity Channel Index
DDPG	Deep Deterministic Policy Gradients
DJIA	Dow Jones Industrial Average
DPG	Deterministic Policy Gradients
DQN	Deep-Q-Network
DRL	Deep Reinforcement Learning
EMA	Exponencial Moving Average
EOM	Ease of Movement
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
GDPG	Gated Deterministic Policy Gradient
GDQN	Gated Deep Q-learning
GPU	Graphics Processing Units
GRU	Gated Recurrent Units
IA	Inteligência Artificial
IBOVESPA	Índice da Bolsa de Valores de São Paulo
IDE	Integrated Development Environment
IFR	Índice de Força Relativa
JSE	JSE Limited Top 40
KAMA	Kaufman's Adaptive Moving Average
KOSDAQ	Korean Securities Dealers Automated Quotations

LSTM	Long Short Term Memory
MA	Moving Average
MACD	Moving Average Convergence/Divergence
MADDPG	Multi Agent Deep Deterministic Policy Gradients
MARL	Multi-Agent Reinforcement Learning
MDP	Markov Decision Processes
MEM	Média Exponencial Móvel
MLP	Multilayer Perceptron
MM	Média Movel
MSE	Mean Squared Error
OBV	On-Balance Volume
OHLC	Open High Low Close
OUN	Ornstein-Uhlenbeck Noise
PG	Policy Gradient
POMDP	Partially Observable Markov Decision Process
PPO	Proximal Policy Optimization
PSAR	Parabolic Stop And Reverse
RL	Reinforcement Learning
RNC	Redes Neurais Convolucionais
RNR	Redes Neurais Recorrentes
ROC	Rate Of Change Indicator
RSI	Relative Strength Index
SAC	Soft Actor-Critic
SENSEX	S&P BSE Sensex
SR	Sortino Ration
SVM	Support Vector Machine
TD	Temporal Difference
TD3	Twin Delayed Deep Deterministic Policy Gradient
TDQN	Trading Deep Q-Network
TRPO	Trust Region Policy Optimization
TSX	Toronto Stock Exchange Index
VDBE	Value Difference Based Exploration
VR	Volatility Ratio

VWAP	Volume Weighted Average Price
WVAD	Williams Variable Accumulation Distribution

LISTA DE SÍMBOLOS

α	Taxa de aprendizagem
A_{t}	Ação a no tempo t
A_{c_s}	Número de ações possíveis no estado s
ϵ	Probabilidade de executar uma ação aleatória utilizando uma política $\epsilon - greedy$
$E(X)$	Esperança dada a uma valiável aleatória X
γ	Fator de desconto do reforço, $\gamma \in [0, 1]$
G_{t}	Retorno no tempo t
$J(\pi)$	Recompensa cumulativa sob uma política π
π_{t}	Política π no tempo t
$p(s'\|s, a)$	Probabilidade da transição para o estado s', do estado s dada a ação a
$p(s', r\|s, a)$	Probabilidade da transição para o estado s', com retorno r, do estado s e a ação a
$p(s'\|s,t,\pi)$	Probabilidade de transição de s para s' sob uma política π no tempo t
$Q_\pi(s,a)$	Função ação-valor de um estado s, aplicando uma ação a sob uma política π
$Q^\star(s,a)$	Função ação-valor ótima de um estado s, aplicando uma ação a sob uma política π
R_{t}	Reforço no tempo t
s'	Estado s no tempo t + 1
s	Estado s no tempo t
$T(s, a, s')$	Operador Shapley dado no estado s, ação a, e estado s'
$V_\pi(s)$	Função valor de um estado s sob uma política π
$V^\star(s)$	Função valor ótima de um estado s sob uma política π
$X \sim p$	A variável aleatória X tem distribuição p
$\rho^\pi(s')$	Distribuição do estado descontado
$\rho_0(s_{\text{t}})$	Distribuição do estado inicial

SUMÁRIO

1 INTRODUÇÃO

O mercado de ações tem se tornado cada vez mais popular no Brasil, chegando a cerca de dois milhões de pessoas registradas na bolsa de valores segundo a B3. Tal popularidade está relacionada à disseminação do conhecimento necessário para realizar as operações financeiras, ao marketing de corretoras e à necessidade encontrada pelo investidor, com a queda da rentabilidade de investimentos de renda fixa, de buscar por outras categorias de investimentos mais rentáveis. Segundo Bittencourt et al. (2018), e com base na teoria de Mercados Eficientes (ELDER, 2005): o risco é a incerteza de determinado acontecimento, ou seja, a probabilidade do investidor ganhar menos do que o esperado; o retorno é a quantidade de capital ganho. Baseando-se na tolerância ao risco pode-se classificar os investidores em três grandes grupos (BITTENCOURT et al., 2018): o primeiro é composto por investidores mais conservadores, que preferem operações de baixo risco com menor rentabilidade e que selecionam ativos já consolidados, focando em operações de longo prazo e dividendos. O segundo são os investidor mais arrojados, realizam operações em ativos que tem a possibilidade de retorno maior e em operações de curto prazo. Esses se beneficiam das oscilações do mercado para obtenção de lucro. O terceiro grupo é o moderado, são investidores que aceitam correr riscos superiores aos conservadores, e inferiores aos arrojados, com a finalidade de alcançar uma rentabilidade maior.

Este trabalho focará seus estudos referenciando-se ao segundo grupo de investidores e baseando-se na Teoria de Dow (PUGA; RODRIGUES, 2010) de que "O preço reflete todos os fatores envolvidos no mercado", informações corporativas e econômicas que poderiam ter impacto na volatilidade das ações já estão atribuídas ao valor dessas antes mesmo que este conhecimento chegue a público, ou seja, com o estudo dos dados das ações é possível prever sua tendência e tomar decisões para a obtenção de lucro.

O objetivo deste trabalho é o estudo e o desenvolvimento de um algoritmo de Inteligência Artificial (IA) (RUSSELL; NORVIG, 2016), utilizando Aprendizado por Reforço Multi-Agente (MARL) (LITTMAN, 1994) o qual aplica-se à negociação de ações na bolsa de valores.

Para implementar esta proposta, foi construido o que denominou-se de Gestor de Carteira de Investimentos". Composto por dois agentes, um responsável pela ação de compra e outro pela venda do ativo, a interação cooperativa entre os dois agentes tem como finalidade gerior o montante investido.

A motivação principal da escolha de um sistema MARL dar-se pelo pouco estudo de sistemas *multi-agent* com foco em operações no mercado financeiro. Segundo a tabela 6, em

que têm-se os trabalhos mais relevantes que utilizaram RL para modelagem de sistemas capazes de operar um único ativo no mercado financeiro, apenas o trabalho do Lee e O (2002), Lee et al. (2007) e do Liu et al. (2020) apresentam abordagens similares a proposta nesta dissertação, sendo que os dois trabalhos apresentados por Lee são otimizações do mesmo trabalho.

Fundamentando-se no trabalho abordado em Lee e O (2002), em que os autores enfatizam que a segmentação da responsabilidade proposta por um sistema *multi-agent* faz com que não sobrecarregue um único agente e possibilite a criação de agentes especialistas nas tomadas de decisão necessárias em estados distintos (esta ideia de agentes especialistas em tomadas de decisão também é discutida no projeto de Yang et al. (2020), mesmo não sendo um projeto que aplica MARL). Partindo para a abordagem deste trabalho, ao invés de ter um único agente genérico, poderia-se ter agentes especializados em operar em tendências de baixa ou tendencias de alta, motivo da utilização de dois agentes.

O algoritmo que será estudado é o MADDPG (*Multi Agent Deep Deterministic Policy Gradients*) (LOWE et al., 2017), algoritmo de aprendizado por reforço profundo *multi-agent* baseado em ator-crítico podendo ser cooperativo ou competitivo. Modelos de aprendizado por reforço, baseados em ator-crítico, contêm uma estrutura responsável por aprender a política e executar a ação (ator), e outra responsável por criticar tais ações executadas pelo ator (crítico) que aprende a função valor.

O MADDPG é um modelo proposto para soluções de problemas que são *multi-agent*, estado contínuo e ação contínua, apresentados em tarefas de jogos (LOWE et al., 2017), tarefas de controle colaborativo de veículos aéreos não tripulados (QIE et al., 2019), entre outras aplicações. A abordagem proposta do "Gestor de Carteira"se enquadra dentro dessas três características: *multi-agent* (um responsável pela compra outro pela venda), estados contínuo (preço e indicadores relacionados ao preço) e ação contínua (proporcional do capital que será investido no ativo).

O restante desta monografia é organizado da seguinte maneira: o Capítulo 2 descreve a fundamentação teórica abordando conceitos da área de Inteligência Artificial, Aprendizado por Reforço e a base utilizada para o cálculo dos indicadores técnicos empregados no projeto; o Capítulo 3 apresenta uma breve revisão bibliográfica da área de pesquisa do projeto, apresentando trabalhos similares ao desta dissertação; o Capítulo 4 apresenta a metodologia utilizada na modelagem do problema proposto por este trabalho; o Capítulo 5 expõe como foi feita a implementação do trabalho e aborda alguns problemas de implementação; o Capítulo 6 demonstra os resultados experimentais obtidos; o Capítulo 7 aborda as análises dos resultados apresenta-

dos no Capítulo 6; e finalmente, o Capitulo 8 apresenta as conclusões do projeto com possíveis melhorias de modelagem.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Aprendizado por Reforço

O aprendizado por reforço é uma das formas mais naturais de aprendizado. A experimentação com o ambiente é vista, desde os primeiros meses de vida de um ser humano tentando engatinhar e andar. Tal proeza é recompensada pelo sorriso de satisfação de seus pais.

A experimentação com o ambiente e a realimentação da recompensa fazem com que seja aprendida não apenas a melhor ação para um estado único e imediato, e sim, a melhor ação que acarretará a maior recompensa futura. Fazendo um paralelo, podemos citar um agente dentro de um labirinto. Ele sabe que se seguir para uma determinada direção não terá uma recompensa imediata, no entanto estará mais próximo da saída, ou seja, do seu real objetivo.

Os componentes principais para um sistema de aprendizado por reforço são: o agente, o ambiente no qual ele interage, a política, o sinal de reforço, a função valor, a função ação-valor e o modelo do sistema (SUTTON; BARTO, 2018).

A política define como o agente vai se comportar em um determinado momento, tendo como base a visão psicológica, a qual seria um conjunto de regras ou associações de estímulo e resposta. Em alguns casos, a política pode ser uma função, uma tabela ou uma rede neural.

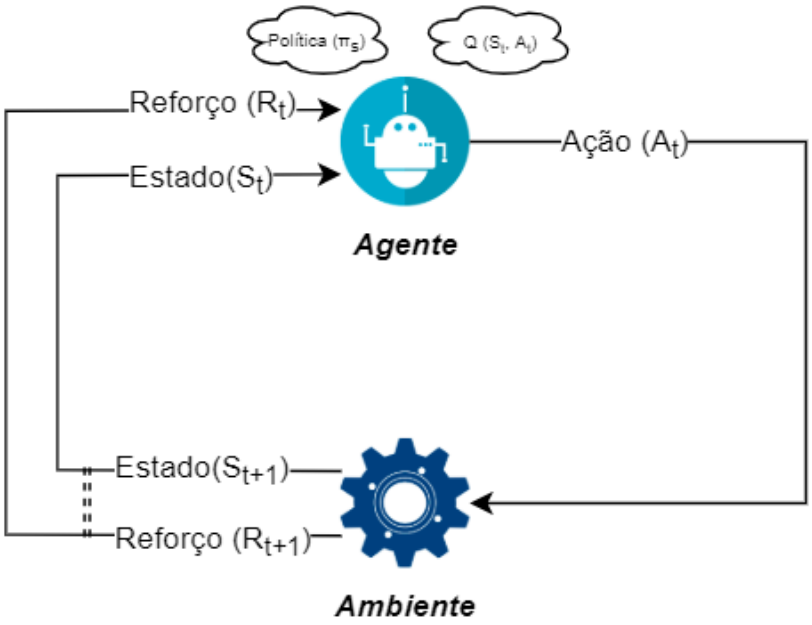
O sinal de reforço define o objetivo do problema, através do qual o agente recebe um reforço positivo ou negativo do ambiente. O reforço é responsável por guiar o agente que tem como objetivo maximizar o ganho fornecido pelo ambiente. Fazendo um paralelo com a biologia, pode-se dizer que o reforço é similar ao prazer e a dor.

A função valor transmite a ideia da recompensa no tempo, ou seja, enquanto tem-se o estado atual e a recompensa dada por esse estado, a função valor expressa a recompensa futura que será dada caso o ator faça uma decisão no estado atual. Na biologia, por exemplo, enquanto a recompensa é representada pelo prazer ou pela dor, a função valor é a prudência, ter a ciência de que uma decisão hoje, poderá lhe dar muito mais prazer futuramente.

O modelo do sistema é responsável por expressar o comportamento do ambiente, e permite inferências para este, ou seja, a partir do modelo, pode-se pressupor o comportamento do ambiente para planejar as melhores decisões a serem feitas. Os métodos que utilizam modelos do sistema são comumente chamados de *model-based*. Em oposição, há os métodos chamados de *model-free* que são modelos de ação livre que aprendem com a experimentação e não modelam o ambiente.

O modelo clássico de aprendizado por reforço pode ser visto na figura 1, onde percebem-se os elementos apresentados previamente.

Figura 1 – Modelo simplificado de aprendizado por reforço



Fonte: Autor (2021)

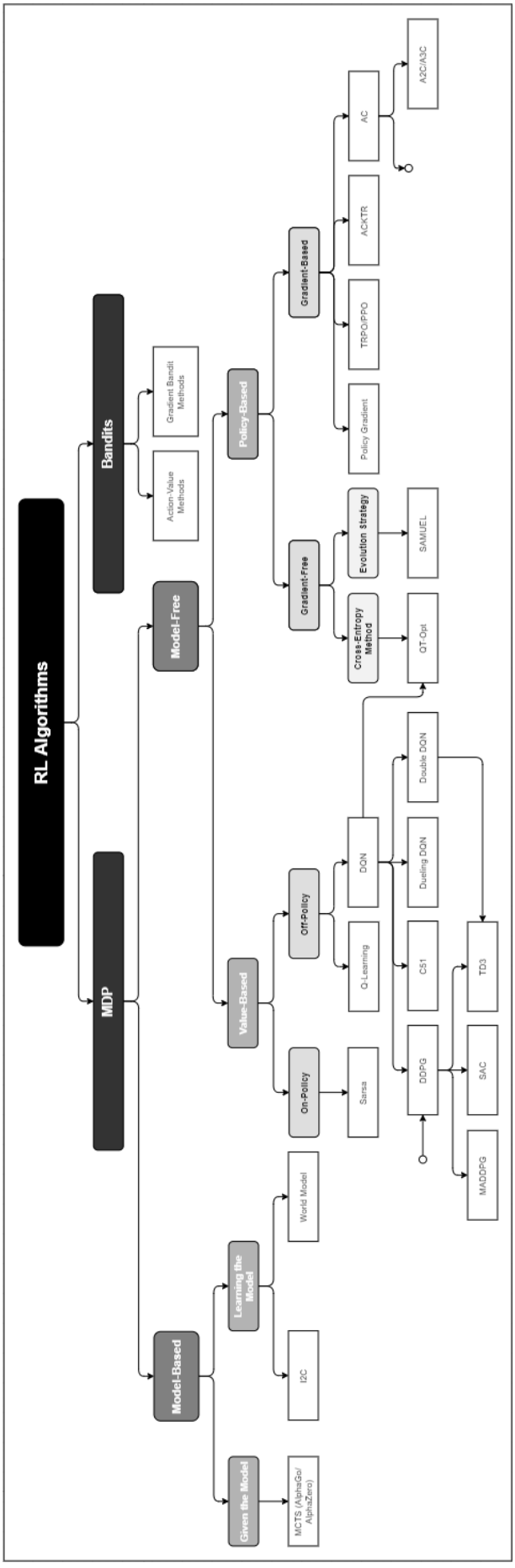
Segundo Kaelbling, Littman e Moore (1996), a maior diferença entre modelos de aprendizado por reforço e aprendizado supervisionado é que ao agente não é apresentado um par de dados (entrada e saída) para modelar um comportamento. Partindo do princípio que o ambiente é *determinístico*, ou seja, dado o mesmo estado para a mesma ação espera-se a mesma saída, sendo assim o agente atua no ambiente para compreendê-lo. Lembrando que os modelos mais simples de aprendizado por reforço fazem menção ao fato do ambiente ser *determinístico*, porém existem modelos de aprendizado por reforço para ambiente *não-determinístico*.

2.1.1 Taxonomia

Esta subseção tem como objetivo apresentar uma visão geral dos algoritmos desenvolvidos na área de RL. A taxonomia apresentada na figura 2 é uma adaptação feita do trabalho de Dong, Ding e Zhang (2020) para apresentar o contexto geral em que o algoritmo estudado se encaixa na área como um todo.

No ramo inicial há duas possíveis variações para os algoritmos de aprendizado por reforço: MDP e *Bandits*. Os algoritmos fundamentados no MDP são estudados e aplicados neste

Figura 2 – Taxonomia do aprendizado por reforço



Fonte: Adaptado de Dong, Ding e Zhang (2020)

trabalho, os *Bandits* não são estudados neste trabalho e aparecem na taxonomia para mostrar que existem outros ramos e sistemas diferentes do MDP quando se aplica RL.

O primeiro ramo do MDP apresenta os algoritmos *model-free* e *model-based* que são algoritmos que são baseados no modelo do ambiente ou não, quando o algoritmo é baseado no modelo do ambiente, ele pode aprender o modelo do ambiente ou este modelo pode ser fornecido para o algoritmo, criando-se assim mais duas subdivisões: *Given the Model* ou *Learning the Model*.

Os algoritmos livres de modelo são subdivididos em: *Value Based* ou *Policy Based*, sendo eles, respectivamente, modelos que otimizam a função ação valor para encontrar a melhor ação a ser executada e, modelos que otimizam diretamente a política de acordo com as interações e o reforço obtido pelo ambiente para encontrar a melhor ação.

Os algoritmos baseados em política se subdividem em *Gradient-Free* e *Gradient-Based*. Os algoritmos que são *Gradient-Based* aplicam o gradiente da política e são tipicamente utilizados quando a ação é contínua, beneficiando-se da escalabilidade para casos de alta dimensão. O *Gradient-Free* não aplica o gradiente da política, porém trem um processo de treinamento mais rápido por ser um processo mais simples e menos custoso computacionalmente.

2.1.2 *Exploitation* x *Exploration*

No processo de aprendizado por reforço, o agente deve tomar a melhor decisão no estado em que se encontra. Supondo que o agente aprendeu que em um determinado estado s ele deve tomar a ação a, se todas as vezes em que ele estiver no estado s, adotar a mesma decisão a, julgando que é a melhor a se fazer, dado uma experimentação prévia, ele poderá não explorar o ambiente por completo, deixando passar possibilidades as quais ele seria melhor recompensado que anteriormente.

O processo conservador do agente, que consiste em executar uma decisão que julga certa e será recompensado é chamado de *exploitation*, no entanto o agente pode ou não estar tomando a melhor decisão. Quando o agente opta por uma decisão incerta, nao tendo experimentado anteriormente, e não sabendo se haverá recompensa positiva ou negativa, chama-se de *exploration*, ou seja, o processo de exploração do ambiente pelo agente (SUTTON; BARTO, 2018).

Pode-se fazer um paralelo do *exploration* e *exploitation* com a vida. Supondo que um agente tem um emprego, ganhando um valor mensal, o fato de continuar no mesmo emprego,

com a mesma recompensa mensal, é caracterizada pelo *exploitation* e, caso o agente queira se aventurar em novos empregos, caracteriza-se como uma *exploration*.

Quando o agente não faz *exploration*, ele pode chegar ao máximo local da solução do problema desejado, fazendo com que existam outras possibilidades de decisões que o recompensariam mais. Se o agente só faz *exploration* e não faz *exploitation*, não estabilizará sua recompensa, ele pode não chegar ao objetivo proposto pelo reforço.

A melhor forma de ponderar *exploitation* e *exploration* é a chave para a construção de um sistema de aprendizado por reforço que tenderá para a melhor solução do problema proposto (SUTTON; BARTO, 2018).

2.1.3 Algoritmos para decidir entre *Exploration* ou *Exploitation*

A seção anterior (2.1.2) aborda o que é *Exploitation* e *Exploration*. Utilizando o conhecimento base anterior, esta seção tem como objetivo estudar os métodos clássicos para executar *exploration* e *exploitation*.

Épsilon-Greedy (e-Greedy)

Segundo Sewak (2019), o Epsilon-Greedy é uma das técnicas mais populares para decisão de *exploration* e *exploitation*. O valor de épsilon diz a porcentagem de chance de o agente executar uma ação aleatoria no estado s, por exemplo, para $\epsilon = 0.1$ o agente tem 10% de chance de executar uma ação aleatória no estado s, e 90% de chance de executar o que é denominado de ação gananciosa, ou seja, baseando-se na função ação-valor Q, escolhe-se a ação que tem o maior retorno.

Algoritmo 1 – Algoritmo aplicado para seleção da ação com base no epsilon: Autor (2021)

```
n ← Valor aleatório (distribuição uniforme) entre 0 e 1
se n < ϵ então
    A ← Ação aleatória do espaço de estados
fim
senão
    A ← max_a Q^π(s, a)
fim
Retorna A
```

Sewak (2019) escreve sobre técnicas de como escolher o melhor valor para épsilon, partindo do pressuposto que o MDP é determinístico, menor o valor de épsilon e em contrapartida,

para um MDP estocástico o valor de épsilon é maior. Tais técnicas podem ser sintetizadas em: quanto mais se conhece sobre o ambiente, menor o épsilon; quanto menos se conhece sobre o ambiente, maior o épsilon. Nota-se uma relação inversamente proporcional entre épsilon e o conhecimento sobre o ambiente.

Épsilon adaptativo em função do tempo

Utilizando como base o que foi abordado anteriormente nesta seção, tem-se recomendações que quanto mais se conhece sobre o ambiente, menor deve ser o valor de épsilon, porque as decisões anteriores já foram suficientes para nortear a melhor ação a ser executada.

O algoritmo do épsilon adaptativo utiliza um decaimento de épsilon no tempo, partindo do pressuposto de que, quanto maior for o tempo interagindo com o ambiente, maior será o conhecimento do agente sobre este ambiente e menor tem que ser o épsilon, pois não há a necessidade de executar *exploration* com tanta frequência.

A função de decaimento a seguir, foi retirada do livro Sewak (2019), *c* representa uma constante muito pequena para que o denominador da função não zere (c = 1e-7).

$$\epsilon = \frac{1}{log\,(time + c)} \tag{1}$$

Action Adaptive Epsilon Algorithms

Esta técnica utiliza o épsilon dinâmico, similar à tratada na seção anterior, porém leva-se em consideração o número de ações possíveis em um determinado estado. Segundo Sewak (2019) quanto maior o número de ações possíveis em um determinado estado, mais deve-se executar *exploitation*, ao invés de *exploration*, porque existem caminhos diferentes a ser tomados que podem aprimorar o conhecimento do agente no ambiente.

A equação descrita para a probabilidade de executar *exploitation* é (SEWAK, 2019):

$$P_{\text{explore}} = \frac{\epsilon}{|A_{c_s}|} \tag{2}$$

Value Difference Based Exploration

O VDBE (do inglês *Value Difference Based Exploration*) é um algoritmo que segue princípios similares aos algorítimos apresentados anteriormente. Tendo como base de que quanto maior o valor de épsilon, mais *exploitation* o agente fará e esta necessidade está relacionada com o conhecimento do ambiente pelo agente. O VDBE utiliza o TD (para o Q-Learning) de modo a quantificar o quanto o agente sabe sobre o ambiente e quanto mais ele o conhece, menor o valor do épsilon. As equações a seguir foram apresentadas no trabalho de Tokic (2010) e a equação 3 diz respeito à modelagem do comportamento, utilizando Boltzmann. Já a equação 4 é uma generalização que pode ser desenvolvida para modelos de comportamentos distintos.

$$f(s,a,\sigma) = \frac{1 - e^{\frac{-|\alpha TD-Error|}{\sigma}}}{1 + e^{\frac{-|\alpha TD-Error|}{\sigma}}} \tag{3}$$

$$\epsilon'(s) = \delta f(s, a_{\mathrm{t}}, \sigma) + (1 - \delta)\epsilon_{\mathrm{t}}(s) \tag{4}$$

Temos que σ é uma constante chamada de sensibilidade inversa, e o δ é um parâmetro que determina a influência da ação selecionada na taxa de exploração, na qual $\delta \in [0,1)$. Uma das possibilidades de equacionar δ está descrita pela equação 5 (TOKIC, 2010):

$$\delta = \frac{1}{|A_{\mathrm{s}}|} \tag{5}$$

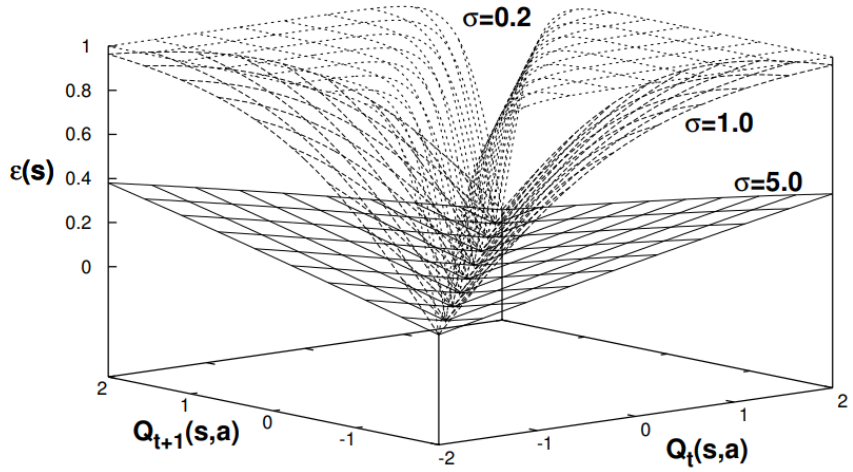
Os valores de σ podem fazer com que o sistema realize mais *exploitation* do que *exploration* em função do aprendizado sobre o ambiente. O gráfico da figura 3 foi apresentado no trabalho de Tokic (2010) e ilustra o efeito da variação do σ, na qual $\epsilon_{\mathrm{t=0}}(s) = 1$ para todos os valores amostrados de σ.

Ornstein-Uhlenbeck Noise

O processo de Ornstein-Uhlenbeck, em física, é aplicado para modelar a velocidade de uma partícula browniana sob a influência de fricção. O movimento browniano é o movimento aleatório de partículas quando suspensas em um fluido resultante de suas colisões com outras partículas no mesmo fluido (PALMAS et al., 2020).

O Ornstein-Uhlenbeck Noise (OUN) é uma técnica utilizada na etapa de *exploration* do treinamento. Este ruído está correlacionado temporalmente e é centrado em uma média zero.

Figura 3 – Gráfico que ilustra $f(s,a,\sigma)$



Fonte: Tokic (2010)

Como o agente não tem conhecimento do modelo, é necessário explorá-lo a fim de conhecer as melhores ações a se fazer em determinado estado em função da recompensa, o ruído de Ornstein-Uhlenbeck (Algoritmo 2) pode ser usado como uma amostra para gerar esse conhecimento (PALMAS et al., 2020).

Os autores não deixam claro o significado que cada variável do algorítimo, principalmente μ, θ e σ e, aparentemente, são definidos empiricamente e seus significados são função do problema analisado. Com base no algoritmo proposto é possível identificar que σ é uma constante multiplicadora que intensifica o valor da variável aleatória, μ é uma constante diferencial e θ é uma constante aplicada para intensificar o valor da diferença entre μ e x.

Algoritmo 2 – Algoritmo para cálculo do OUN, modificação da implementação de Palmas et al. (2020).

```
1 Inicia actionDimension, μ, θ, σ
2 x = ArrayWithOnes(actionDimension) * μ
3 dx = θ * (μ - x) + σ * ArrayRandomWithLength(Length(x))
4 xr = x + dx
5 Retorna xr
```

2.1.4 Modelos de comportamento

Existem alguns modelos de comportamento fundamentais no aprendizado por reforço, que serão relatados em mais detalhes nesta seção. Vale ressaltar que as informações a seguir foram escritas com base no trabalho Kaelbling, Littman e Moore (1996).

Segundo Kaelbling, Littman e Moore (1996) existem três modelos que são, mais frequentemente, utilizados nos trabalhos de aprendizado por reforço.

O primeiro diz respeito ao chamado horizonte finito. Este modelo é o mais intuitivo e simples de todos, na qual em um determinado momento no tempo, o agente otimiza o retorno esperado para os próximos *h* passos. A Equação 6 diz respeito ao horizonte finito, e o modelo de comportamento proposto é a esperança da somatória de todos os reforços dados nos instantes de tempo $t = 0$ até o instante de tempo *h*.

$$M_{\text{fh}} = E(\sum_{t=0}^{h} r_{\text{t}}) \tag{6}$$

O segundo modelo diz respeito ao chamado horizonte infinito descontado, similar ao horizonte finito, com o acréscimo de considerar a recompensa descontada no tempo, a partir de um fator de desconto γ, que pode variar de zero a um. O fator γ pode ser analisado como um fator de interesse, probabilidade de estar vivo no próximo passo, ou até mesmo uma técnica matemática para limitar a soma infinita. O modelo horizonte infinito descontado é mais utilizado devido a sua formulação matemática e pode ser visto na equação 7.

$$M_{\text{ihd}} = E(\sum_{t=0}^{h} \gamma^t r_{\text{t}}) \tag{7}$$

O terceiro modelo é chamado de recompensa média em que o agente executa uma ação que otimiza a recompensa média a longo prazo, podendo ser expresso pela equação 8

$$M_{\text{ar}} = \lim_{h \to \infty} E(\frac{1}{h} \sum_{t=0}^{h} r_{\text{t}}) \tag{8}$$

2.1.5 Propriedade de Markov

Sempre que se modela um sistema de aprendizado por reforço, o estado *s* representa o estado atual do ator. Quando tem-se um problema em que o estado futuro depende somente do estado atual, ou seja, o que o levou para onde ele está, não influenciará, necessáriamente, aonde ele irá, chamamos então de propriedade de Markov. Por exemplo, uma bala de canhão tem uma velocidade, altura, massa, força gravitacional em um ponto *p*. Pode-se dizer que ela está em um estado s_p e o que a levou para aquele estado, não terá influência em seu trajeto. Sua trajetória depende somente do estado atual s_p que contem as informações da velocidade, altura, massa

e força gravitacional. Outro exemplo é um jogo de damas. A melhor decisão a ser executada dependerá do estado atual das peças e não do que as levou para aquele estado.

A modelagem matemática proposta para o problema considera que há estados finitos no ambiente. Simplificando o modelo, é possível trabalhar com somas e probabilidades. Considerando um ambiente que possa responder no estado *t+1* a uma ação dada em *t* a dinâmica do sistema pode ser definida especificando a distribuição de probabilidade expressa na equação 9.

$$P_{\mathrm{r}}\left\{R_{\mathrm{t+1}}=r, S_{\mathrm{t+1}}=s'|S_0, A_0, R_1 ... S_{\mathrm{t\text{-}1}}, A_{\mathrm{t\text{-}1}}, R_{\mathrm{t}}, S_{\mathrm{t}}, A_{\mathrm{t}}\right\} \tag{9}$$

Se todos os estados *t+1* são propriedades de Markov, ou seja, se todos os estados futuros independem dos estados anteriores, pode-se simplificar a equação 9 para a equação 10.

$$p\left(s', r|s,a\right) = P_{\mathrm{r}}\left\{R_{\mathrm{t+1}}=r, S_{\mathrm{t+1}}=s'|S_{\mathrm{t}}, A_{\mathrm{t}}\right\} \tag{10}$$

A propriedade de Markov é fundamental quando se modelam sistemas de aprendizado por reforço porque ela simplifica o modelo de tomada de decisão. Quando no ambiente não há propriedade de Markov, convém a aproximação deste para que ele seja Markoviano, sendo assim, possível aplicar os estudos relacionados a aprendizado por reforço Markoviano.

2.1.6 Processo de Decisão de Markov

Na seção 2.1.5 foi descrito o que é a propriedade de Markov. Quando aplicada em problemas de aprendizado por reforço, pode-se chamar tal tarefa de Processo de Decisão de Markov. No caso particular em que a tarefa tenha estados finitos pode-se chamar, mais especificadamente, de: Processo Finito de Decisão de Markov.

Dado um estado qualquer *s* e a ação *a*, a probabilidade de cada par subsequente de estado e reforço *s'* e *r* pode ser expressa pela equação 11.

$$p\left(s', r|s,a\right) = P_{\mathrm{r}}\left\{S_{\mathrm{t+1}}=s', R_{\mathrm{t+1}}=r|S_{\mathrm{t}}=s, A_{\mathrm{t}}=a\right\} \tag{11}$$

Considerando que o ambiente é Markoviano, pode-se escrever o reforço em função do estado e da ação, como descrito na equação 12.

$$r\left(s,a\right) = \sum_{r\in R} r \sum_{s'\in S} p\left(s', r|s,a\right) \tag{12}$$

A probabilidade de transição de estado pode ser descrita como:

$$p\left(s'|s,a\right) = \sum_{r \in R} p\left(s', r|s,a\right) \tag{13}$$

Por fim, o retorno esperado de uma ação atual, estado atual e próximo estado é dado por:

$$r\left(s, a, s'\right) = \frac{\sum_{r \in R} rp\left(s',r|s,a\right)}{p\left(s'|s,a\right)} \tag{14}$$

A ideia por trás da função valor foi expressa na seção 2.1, em resumo, tenta quantificar o quanto é bom é para o agente estar em um determinado estado. Para um MDP, a equação 15 expressa a função valor de um estado *s* sob uma política π.

$$V_{\pi}\left(s\right) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} R_{\mathrm{t+k+1}} | S_{\mathrm{t}} = s\right] \tag{15}$$

Similar à função valor, tem-se a função ação-valor para uma política π que expressa o valor de uma ação *a* em um estado *s* sob uma política π, podendo ser expressa por:

$$Q_{\pi}\left(s, a\right) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} R_{\mathrm{t+k+1}} | S_{\mathrm{t}} = s, A_{\mathrm{t}} = a\right] \tag{16}$$

2.1.7 Programação Dinâmica

Segundo Vazirani (2009), o termo programação dinâmica foi expresso, pela primeira vez, por Richard Bellman na década de 1950, o termo programação tem muito pouco a ver com escrever código. Naquele período, programação significava planejamento, e a Programação Dinâmica era concebida para planejar processos de vários estágios de forma ideal. Em uma formulação típica de programação dinâmica, um problema é reduzido a subproblemas os quais são menores que o problema principal.

Neste trabalho, apresentaremos a programação dinâmica aplicada a aprendizado por reforço, e a utilização dela é feita em um conjunto de algoritmos desenvolvidos de modo a solucionar a equação de Bellman para a função valor, ou para a função ação-valor (17 e 18). Para isto, leva-se em consideração que o problema é um Processo de Decisão de Markov (MDP) e possui estados finitos e discretos. Para problemas contínuos, existem abordagens de discretização do problema para estados finitos, possibilitando a aplicação de programação dinâmica.

$$V^{\star}\left(s\right) = \max_{\mathrm{a}} \sum_{s',r} p\left(s',r|s,a\right)\left[r + \gamma V^{\star}\left(s'\right)\right] \tag{17}$$

$$Q^{\star}\left(s,a\right)=\sum_{s',r}p\left(s',r|s,a\right)\left[r+\gamma\max_{\mathrm{a}'}Q^{\star}\left(s',a'\right)\right] \tag{18}$$

2.1.8 Avaliação de Política

O algoritmo da avaliação de política consiste em calcular a função valor dada uma política iterando com o ambiente um número de repetições suficientes para que a variação da função valor seja muito baixa.

A Equação 20 descreve a função valor para *k+1*, quando $k \rightarrow \infty$ ele tende a função valor ótima de Bellman.

$$V_{\mathrm{k+1}}\left(s\right)=\sum_{a}\pi\left(a|s\right)\sum_{s',r}p\left(s',r|s,a\right)\left[r+\gamma V_{\mathrm{k}}\left(s'\right)\right] \tag{19}$$

Sendo assim, pode-se construir o algoritmo 3 com o objetivo de calcular a função valor ótima de Bellman com base na equação 20.

Algoritmo 3 – Algoritmo que calcula a função Valor utilizando avaliação de política, de acordo com Sutton e Barto (2018).

Entrada π, política
Inicia um array V(s)=0, para cada s $\in S$
repita
    $\Delta \leftarrow 0$
    para cada $s \in S$ **faça**
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$
        $\Delta \leftarrow max(\Delta,|v - V(s)|)$
    fim
até $\Delta < \theta$ *(número positivo pequeno)*;
Retorna V $\approx v_{\pi}$

2.1.9 Q-Learning

O Q-Learning foi um dos algoritmos clássicos que proporcionou um avanço significativo na eficiência e aplicação de trabalhos de aprendizado por reforço, e é também utilizado como base para algoritmos de Deep Reinforcement Learning como o Deep-Q-Network (DQN).

Segundo Sutton e Barto (2018), pode-se descrever a atualização da função ação-valor (Q) da seguinte forma:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \tag{20}$$

O Q-Learning fundamenta-se no TD (do inglês *Temporal-Difference Learning*) com o objetivo de descobrir a função Q que mais se aproxima da função $Q^{\star}$ sem depender da política que está sendo seguida, simplificando o algoritmo possibilitando uma convergência mais rápida.

Os detalhes da implementação do Q-Learning podem ser visto no algoritmo 4.

Algoritmo 4 – Algoritmo que calcula a função ação-valor (Q) utilizando o Q-Learning, de acordo com Sutton e Barto (2018).

```
Inicia Q(s,a), ∀ s ∈ S, a ∈ A(s), arbitrariamente, e Q(s-terminal, .) = 0
para cada episódio faça
    Inicia S
    para cada passo do episódio até S ser terminal faça
        Escolha A de S usando a política derivada de Q(ex, e-greedy)
        Execute a ação A, observe R, S'
        Q(S,A) ← Q(S,A) + α[R + γ max_a Q(S',a) - Q(S,A)]
        S ← S'
    fim
fim
```

2.1.10 Modelo ator-crítico

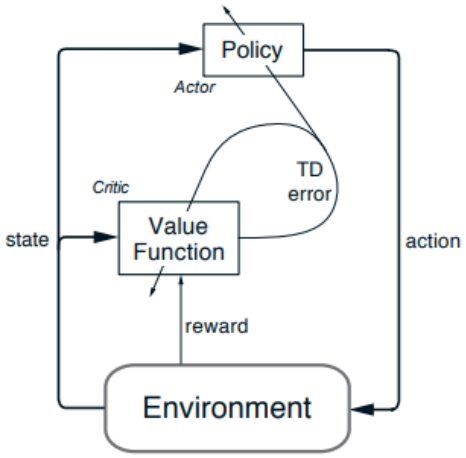
Segundo Sutton e Barto (2018), modelos de aprendizado por reforço baseados em ator-crítico contém uma estrutura responsável por aprender a política e executar a ação (ator) e uma outra responsável por criticar tais ações executadas pelo ator (crítico) que aprende a função valor. Após a execução do estado analisado, é o crítico que avalia o novo estado para determinar se o retorno foi melhor ou pior do que o esperado. Tal avaliação denomina-se erro TD e poderá ser calculado da forma apresentada na equação 23.

$$\delta = r + \gamma V(s') - V(s) \tag{21}$$

A figura 4 ilustra a arquitetura do sistema ator-crítico analisado nesta seção.

Existem duas grandes vantagens na utilização de um modelo ator-crítico: a primeira exige um custo computacional menor para tomada de decisão e, a segunda, a possibilidade de aprender explicitamente politicas estocásticas, facilitando o aprendizado em modelos psicológicos e biológicos.

Figura 4 – Arquitetura Ator-Crítico



Fonte: Sutton e Barto (2018)

2.1.11 Multi-Agent Reinforcement Learning (MARL)

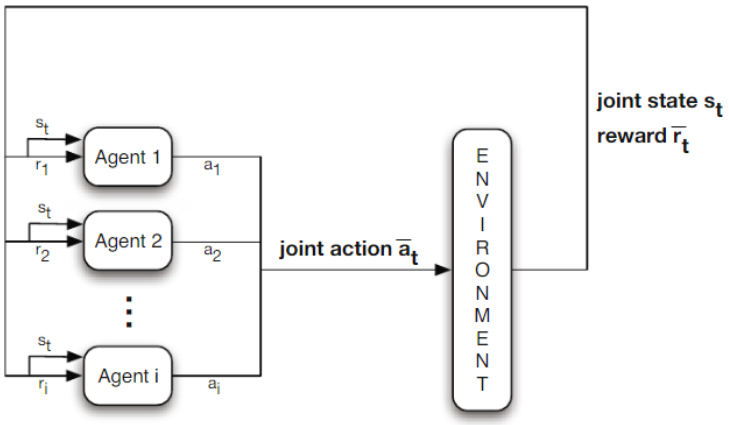
O aprendizado por reforço multiagente, do inglês *Multi-Agent Reinforcement Learning* (MARL), utiliza-se da base dos sistemas de aprendizado por reforço, juntamente com a teoria dos jogos a fim de se criar uma interação entre os agentes, proporcionando cooperação ou competição entre eles.

O MARL foi desenvolvido com a intenção de resolver problemas que necessitem da interação entre um ou mais agentes, compartilhamento dos estados e ações entre eles. A informação compartilhada entre os agentes é considerada para a tomada de decisão dos mesmos, os quais são entidades autônomas, e tem objetivos individuais e capacidades de tomada de decisões independentes. No entanto, são influenciados pelas decisões uns dos outros, diferentemente de abordagens de aprendizagem por reforço distribuído ou paralelo.

Existem alguns desafios quando se aplica MARL em relação ao RL comum. Por exemplo, dado um mundo de grades, onde dois agentes necessitam maximizar o retorno global, se não houver comunicação e mudança de postura de ambos os agentes pensando na maximização da recompensa global, chegarão em pontos de máximos locais. O desafio está em um agente executar uma ação que dê um retorno menor individual a fim de que o grupo se beneficie.

A figura 5 ilustra de forma simplificada a arquitetura de um sistema *multi-agent*.

Figura 5 – Arquitetura sistema *multi-agent*



Fonte: Nowe, Vrancx e De Hauwere (2012)

2.1.12 Jogos Estocásticos

Os jogos estocásticos, do inglês *stochastic game framework*, são uma extensão e aplicação da teoria do processo de decisão de markov (MDP), estudado anteriormente, empregada a sistemas *multi-agent*. Uma das complexidades abordadas quando aplica-se MDP em *multi-agent* é que os ambientes são inerentemente não estacionários, uma vez que outros agentes são livres para mudar o seu comportamento, pois eles também aprendem a se adaptar.

Similar ao MDP e diferenciando-se apenas que o próximo estado e o retorno dependem das ações conjuntas de todos os agentes. Segundo Bowling (1998), pode-se definir a estrutura de jogos estocásticos como uma tupla (n, S, $A_{1...n}$, T, $R_{1...n}$), em que n é o número de agentes, S são os estados, A_i são as ações possíveis de serem executadas pelo agente *i* (A é o espaço de ação conjunta de todos os agentes $A_1 \times ... \times A_n$), T é a função de transição $S \times A \times S \rightarrow [0,1]$ e R_i é o retorno dado ao agente *i* em que $S \times A \rightarrow R$.

No estado S, cada agente executa uma ação que é escolhida independentemente da ação dos outros, e cada um recebe uma recompensa R_i. Existem dois grupos de jogos que podem ser categorizados em função da recompensa conjunta: caso o total de recompensa satisfaça a equação 22, o jogo é chamado de *soma zero*, do inglês *zero sum*; caso satisfaça a equação 23, o jogo é chamado de *soma geral*, do inglês *general-sum* (HU; WELLMAN, 1999).

$$\phi = \sum_{i}^{n} R_i(S, A_1, A_2) = 0 \tag{22}$$

$$\phi = \sum_{i}^{n} R_{\mathrm{i}}(S, A_1, A_2) \neq 0 \tag{23}$$

O primeiro algoritmo foi dado por Shapley (algoritmo 5) como resultado de sua prova da existência de equilíbrios em jogos estocásticos de soma zero. O algoritmo usa uma técnica de diferenciação temporal para armazenar os valores dos próximos estados em uma matriz de jogos denominada $G_s(V)$. A função de valor V é então atualizada utilizando a matriz de jogos para cada estado.

Algoritmo 5 – Algoritmo que calcula a função valor (V) utilizando o Shapley, de acordo com Hu e Wellman (1999).

```
Inicia V arbitrariamente
para cada episódio faça
    para cada s ∈ S faça
        G_s(V) = R(s,a) + γ Σ_{s' ∈ S} T(s,a,s') V(s')
        V(s) ← [G_s(V)]
    fim
fim
```

Outra técnica de resolução é o algoritmo Minimax-Q desenvolvido por Littman, este aplica conceitos do Q-Learning para solucionar problemas de aprendizado por reforço *multi-agent* de jogos estocátiscos de soma zero. A função Q é estendida para descrever o valor das ações conjuntas, e o algoritmo 6 descreve o Minimax-Q.

Algoritmo 6 – Algoritmo para solução de Estrutura de Jogos Estocásticos utilizando o Minimax-Q (Littman), de acordo com Hu e Wellman (1999).

```
Inicia Q(s ∈ S, a ∈ A) arbitrariamente
Configura um valor fixo para α (taxa de aprendizado)
para cada episódio faça
    De um estado s selecione uma ação a_i que resolva a matriz de jogos
      [Q(s,a)_{a ∈ A}], utilizando exploration
    Observe a ação conjunta a, retorno r, e o próximo estado s',
    Q(s,a) ← (1 − α)Q(s,a) + α(r + γV(s'))
    onde,
    V(s) = Value([Q(s,a)_{a ∈ A}])
fim
```

2.1.13 Aprendizado por reforço profundo

Redes Neurais Profundas, do inglês *Deep Neural Network*, é um assunto predominantemente discutido nos últimos anos e, de forma genérica, pode-se dizer que é uma técnica de

aproximação de função que pode ser aplicada em áreas diferentes da ciência, tais como: processamento de imagem, voz, texto ou qualquer forma de dado estruturado.

Segundo (GRAESSER; KENG, 2019), a primeira vez que foram combinadas redes neurais profundas com aprendizado por reforço, ocorreu em 1991 quando Gerald Tesauro treinou uma rede neural para jogar gamão. Todavia, somente em 2015, com os trabalhos da DeepMind voltados a jogos de Atari, foi que a área do aprendizado por reforço profundo começou a se desenvolver.

Sistemas de aprendizado supervisionado necessitam de uma função alvo ou um rótulo que informe qual é a saída esperada para um conjunto de dados de entrada do sistema de forma a parametrizar a rede para que seja possível descrever a saída em função da entrada.

Os estados (dados de entrada) dos sistemas de aprendizado por reforço não são categorizados, ou seja, não se sabe se é um estado positivo ou negativo no instante inicial do treinamento. A interação do agente com o ambiente e a aplicação do algoritmo de aprendizado faz com que os estados, antes desconhecidos, sejam "categorizados" em função da experimentação. Em problemas simples que os estados são finitos e pequenos pode-se implementar tabelas que representam a função Q ou V, porém dependendo da complexidade do problema proposto a representação com tabelas é inviável e faz-se necessário o uso de redes neurais que abstraem a complexidade do problema e são responsáveis por aprender a função alvo (Q ou V).

2.1.14 Deep Q-Networks (DQN)

O Deep Q-Networks é um algoritmo de aprendizado por reforço *off-policy* e baseia-se no valor da diferença temporal (TD) aplicado em problemas com espaço de ação discretos, aprendendo a função ação-valor (Q).

A equação de Bellman para DQN pode ser analisada a seguir (equação 24), e esta independe da política aplicada no momento da coleta de dados, ou seja, DQN é um algoritmo denominado de *off-police*. Off-police significa que a política aplicada pelo agente, no momento da captura dos dados, é diferente da política que ele atualiza.

$$Q^{\pi}\left(s,a\right) \approx r + \gamma max_{\mathrm{a}'} Q^{\pi}\left(s', a'\right) \tag{24}$$

Segundo Graesser e Keng (2019), uma das vantagens de ser *off-police* quando se utiliza redes neurais para aprender a função ação-valor, é que devido à utilização do gradiente do erro

para atualizar a rede, tal atualização é mais suave quando o algoritmo é *off-police* do que *on-police*.

Os detalhes da implementação do DQN podem ser visto no algoritmo 7.

Algoritmo 7 – Algoritmo Deep Q-Learning, de acordo com Graesser e Keng (2019).

Inicia a taxa de aprendizagem α
Inicia a trajetória τ
Inicia o número de batches por passo de treinamento, B
Inicia número de atualizações por batch, U
Inicia o tamanho do batch, N
Inicia experiencia de repetição em memória com tamanho máximo K
Inicia a rede de maneira aleatória com os parâmetros θ
para *m = 1 ... MaxSteps* **faça**
    Colete e armazene h experiências(s_i, a_i, r_i, s'_i) utilizando a política atual **para** *b = 1 ... B* **faça**
        Amostre um batch de experiencias da memória de experiencia de repetição
        para *u = 1 ... U* **faça**
            para *i = 1 ... N* **faça**
                # Calcule o Q para cada exemplo
                $y_i = r_i + \delta_{s'i}\gamma \max_{a'i} Q^{\pi\theta}(s'_i, a'_i)$; onde $\begin{cases} \delta_{s'i} = 0 \text{ se } s'_i \text{ for terminal} \\ \delta_{s'i} = 1 \text{ caso contrário} \end{cases}$
            fim
            # Calcule a perda, por exemplo usando MSE
            $L(\theta) = \frac{1}{N} \sum_i (y_i - Q^{\pi\theta}(s_i, a_i))^2$
            # Atualiza os parâmetros da rede
            $\theta = \theta - \alpha \nabla_\theta L(\theta)$
        fim
    fim
    Decai τ
fim

2.1.15 Deterministic Policy Gradients (DPG)

O DPG é um método de aprendizado por reforço que aplica o gradiente da política utilizado em um espaço de ação contínuo, beneficiando-se da escalabilidade para casos de alta dimensão. O objetivo do agente é maximizar a recompensa cumulativa, com desconto do estado inicial, em uma visão esperada ou estimada, a qual pode ser descrita como:

$$J(\pi) = E_{\tau \sim \pi}[R(\tau)] \tag{25}$$

$$R(\tau) = \sum_{t=0}^{T} \gamma^t R_{\mathrm{t}} \tag{26}$$

R(τ) é o retorno esperado descontado com passos finitos e τ são as amostras da trajetória. Inicialmente, define-se o objetivo de desempenho para a política determinística:

$$J\left(\pi\right) = E_{s \sim \rho^{\pi}, a = \pi\left(s\right)} \left[\sum_{t=1}^{\infty} \gamma^{t-1} R(s,a) \right] \tag{27}$$

$$J\left(\pi\right) = \int_S \int_S \sum_{t=1}^{\infty} \gamma^{t-1} \rho_0(s) p(s'|s,t,\pi) R(s',\pi(s'))] d_{\mathrm{s}} d_{\mathrm{s}}. \tag{28}$$

$$J\left(\pi\right) = \int_S \rho^{\pi}(s) R(s,\pi(s)) d_{\mathrm{s}} \tag{29}$$

Simplificando o desenvolvimento dos cálculos, pode-se chegar à seguinte equação do gradiente (para mais detalhes analise Dong, Ding e Zhang (2020)):

$$\nabla_{\theta} J\left(\pi_{\theta}\right) = \nabla_{\theta} \int_S \rho_0(s) V^{\pi_{\theta}}(s) d_{\mathrm{s}} \tag{30}$$

$$\nabla_{\theta} J\left(\pi_{\theta}\right) = \int_S \rho_0(s) \nabla_{\theta} V^{\pi_{\theta}}(s) d_{\mathrm{s}} \tag{31}$$

$$\nabla_{\theta} J\left(\pi_{\theta}\right) = \int_S \int_S \sum_{t=0}^{\infty} \gamma^t \rho_0(s) p(s \rightarrow s', t, \pi_{\theta}(s)) \nabla_{\theta}(s') \times \nabla_{\mathrm{a}} Q^{\pi_{\theta}}(s',a)|_{a = \pi\theta(s')} d_{\mathrm{s}} \cdot d_{\mathrm{s}} \tag{32}$$

$$\nabla_{\theta} J\left(\pi_{\theta}\right) = \int_S \rho^{\pi_{\theta}}(s) \nabla_{\theta} \pi_{\theta}(s) \nabla_{\mathrm{a}} Q^{\pi_{\theta}}(s,a)|_{a = \pi\theta(s)} d_{\mathrm{s}} \tag{33}$$

A equação 33 faz referência ao DPG *on-policy*, analisando-a ela é função da política aplicada, porém existe uma variação do DPG *off-policy* que pode ser analisada nas equações 34, 35 e 36.

$$\nabla_{\theta} J_{\beta}(\pi_{\theta}) = \int_S \rho^{\beta}(s) (\nabla_{\theta} \pi_{\theta}(s) \nabla_{\mathrm{a}} Q^{\pi_{\theta}}(s,a) + \nabla_{\theta} Q^{\pi_{\theta}}(s,a))|_{a = \pi(s)} d_{\mathrm{s}} \tag{34}$$

$$\nabla_{\theta} J_{\beta}(\pi_{\theta}) \approx \int_S \rho^{\beta}(s) \nabla_{\theta} \pi_{\theta}(s) \nabla_{\mathrm{a}} Q^{\pi_{\theta}}(s,a) d_{\mathrm{s}} \tag{35}$$

$$\nabla_{\theta} J_{\beta}(\pi_{\theta}) = E_{S \sim \rho^{\beta}} [\nabla_{\theta} \pi_{\theta}(s) \nabla_{\mathrm{a}} Q^{\pi_{\theta}}(s,a)|_{s = \pi(s)}] \tag{36}$$

2.1.16 Deep Deterministic Policy Gradient (DDPG)

O DDPG é uma variante do DPG para aprendizado profundo, o algoritmo usa os fundamentos do DQN, ator-crítico e, com base no gradiente da política determinística, atualiza-se a política, utilizando uma abordagem de aprendizado profundo. Uma rede é dedicada para o ator, e outra rede para o crítico. Segundo Dong, Ding e Zhang (2020) o DDPG tem um aprendizado muito eficiente, porém apresenta dificuldades devido a sua fragilidade e sensibilidade a hiperparâmetros.

Considerando o estado s, o próximo estado s' o retorno r que são obtidos pela ação $a = \pi(s,{\theta_{\mathrm{t}}}^{\pi})$ através da política π, pode-se escrever:

$$Q^{\pi}(s,a) = E[r + \gamma Q^{\pi}(s', \pi(s'))] \tag{37}$$

$$Y_{\mathrm{i}} = r_{\mathrm{i}} + \gamma Q^{\pi}(s', \pi(s')) \tag{38}$$

Usando o gradiente descendente para minimizar a função de perda:

$$L = \frac{1}{N}\sum_{i}\left(Y_{\mathrm{i}} - Q(S_{\mathrm{i}}, A_{\mathrm{i}}|\theta^{Q})\right)^{2} \tag{39}$$

Aprendendo em lotes pequenos, tem-se:

$$\nabla_{\theta} J \approx \frac{1}{N}\sum_{i}\nabla_{\mathrm{a}} Q\left(s,a|\theta^{Q}\right)|_{s=S_{\mathrm{i}},\, a=\pi(S_{\mathrm{i}})}\nabla_{\theta}\pi(s|\theta^{\pi})|_{\mathrm{S_i}} \tag{40}$$

O DDPG adota redes neurais semelhantes ao DQN, porém atualiza os parâmetros da rede por uma suavização exponencial (equação 41 e 42), ao invés de copiá-los diretamente. Quando o hiperparâmetros ρ for muito menor que 1, os parâmetros da rede serão alterados mais lentamentamente, de forma a proporcionar mais estabilidade no aprendizado.

$$\theta^{Q'} \leftarrow \rho\theta^{Q} + (1-\rho)\theta^{Q'} \tag{41}$$

$$\theta^{\pi'} \leftarrow \rho\theta^{\pi} + (1-\rho)\theta^{\pi'} \tag{42}$$

Os detalhes da implementação do DDPG podem ser visto no algoritmo 8.

Algoritmo 8 – Algoritmo DDPG, de acordo com Dong, Ding e Zhang (2020).

Hiperparâmetros: Fator de atualização suave ρ, fator de desconto de recompensa γ
Entrada: Buffer de repetição vazio D
Inicia os parametros θ^Q da rede do crítico $Q(s,a|\theta^Q)$ e os parâmetros θ^π da rede do ator $\pi(s|\theta^\pi)$, redes alvos Q' e π'
Inicia a rede alvo Q' e π' com os pesos $\theta^{Q'} \leftarrow \theta^Q, \theta^{\pi'} \leftarrow \theta^{\pi'}$
para *episódio = 1 ... M* **faça**
    Inicia um processo randomico N para a ação de exploração
    Recebe o estado de observação inicial S_1
    para *t = 1 ... T* **faça**
        Seleciona ação $a = \pi(s|\theta^\pi) + N_t$
        Executa ação a, observa o retorno r e o novo estado s'
        Guarda a transição (s, a, r, s') em D
        $Y_i = R_i + \gamma(1 - D_t)Q'(s', \pi'(s',\pi' (s'|\theta^{\pi'})|\theta^{Q'})$
        # Atualiza o crítico minimizando a perda:
        L(θ) = $\frac{1}{N} \sum_i (y_i - Q^{\pi\theta}(s_i,a_i))^2$
        #Atualiza a politica do ator utilizando policy gradient:
        $\nabla_{\theta^\pi} J \approx \frac{1}{N} \sum_i \nabla_a Q(s,a|\theta^Q)|_{s = S_i, a = \pi(S_i)} \nabla_{\theta^\pi} \pi(s|\theta^\pi)|_{S_i}$
        # Atualiza as redes alvos:
        $\theta^{Q'} \leftarrow \rho\theta^Q + (1 - \rho)\theta^{Q'}$
        $\theta^{\pi'} \leftarrow \rho\theta^\pi + (1 - \rho)\theta^{\pi'}$
    fim
fim

2.1.17 Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

O MADDPG é uma variação do DDPG aplicado a sistemas multiagentes, ele é um algoritmo de aprendizado por reforço profundo *multi-agent* e baseia-se em ator-crítico, podendo ser cooperativo ou competitivo. Seu espaço de estados e de ações permitidos são indiferentes, podendo ser contínuos ou discretos.

Os detalhes da implementação do MADDPG podem ser visto no algoritmo 9.

A seção 2.1 foi escrita com o objetivo de explorar e fundamentar o conteúdo desenvolvido sobre aprendizado por reforço, desde os sistemas mais simples até os sistemas *multi-agent* mais complexos, os quais serão implementados para solucionar o problema proposto por este trabalho. A próxima seção (2.2) foi elaborada com o propósito de fundamentar os conceitos e os indicadores utilizados por investidores, a fim de executar as tomadas de decisão de compra ou venda de ativos.

Algoritmo 9 – Algoritmo MADDPG, de acordo com Lowe et al. (2017).

para *episódio = 1 ... M* **faça**
  Inicia um processo randomico N para a ação de exploração
  Recebe um estado inicial x
  para *t = 1 ... MaxEpisodeLength* **faça**
    Para cara agente i, selecione a ação $a_i = \mu_{\theta_i}(o_i) + N_t$ guarde a política atual
    Execute a ação a = (a_1, ..., a_N) e observe o retorno *r* e o novo estado x'
    Armazene (x, a, r, x') no buffer de repetição D
    $x \leftarrow x'$
    para *agente i = 1 ... N* **faça**
      Amostre randomicamente um minibatch de S amostras (x^j, a^j, r^j, x'^j) de D
      $y^j = r_i^j + \gamma Q_i^{\mu'}(x'^j, a'_1, ..., a'_N)|_{a'_k = \mu'_k(o_k^j)}$
      # Atualize o crítico minimisando a perda:
      $L(\theta_i) = \frac{1}{S}\sum_j (y^j - Q_i^{\mu}(x^j, a_1^j, ..., a_N^j))^2$
      # Atualize o ator usando policy gradient:
      $\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i \mu_i(o_i^j)} Q_i^{\mu}(x^j, a_1^j, ..., a_i, ..., a_N^j)\big|_{a_i = \mu_i(o_i^j)}$
    fim
    # Atualiza os parâmetros da rede alvo para cada agente i:
    $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$
  fim
fim

2.2 Teoria de Dow e Indicadores Técnicos

Charles Dow teve suas ideias compiladas e publicadas por W Hamilton, dando origem à teoria de Dow em 1922 (PUGA; RODRIGUES, 2010) e se resume em seis tópicos principais: o preço reflete todos os fatores envolvidos no mercado; o mercado se move em tendências; uma tendência tende a continuar até que haja indicação de sua reversão; os índices devem confirmar as tendências; a história se repete; o volume tem que confirmar a tendência.

Em suma, com os tópicos abordados na Teoria de Dow, nota-se que o preço é um indicador suficiente para descrever o movimento das ações e que, em teoria, os fatores externos que podem impactar no valor, estão embutidos no preço, além disso, comportam-se em tendências que podem ser modeladas e expressas por funções matemáticas.

Indicadores técnicos são índices calculados com base nos valores de preço de abertura, fechamento, máximo e mínimo diário ou volume de operação. São utilizados amplamente por especialistas em compra e venda de ativos financeiros, como ferramentas que indicam se a compra efetuada tem um fundamento técnico, assim informando que será uma operação lucrativa.

Segundo Schiavone (2019) os indicadores técnicos podem informar tendencias, valatilidade, momento, força de mercado, ciclo, entre outros. Tais indicadores têm três funções

principais: alertar a movimentação do mercado, confirmar um outro indicador que foi utilizado e prever a direção de preços futuros. O autor informa que a utilização dos indicadores é particular e o que deve ser levado em consideração é a utilização de indicadores que informam conteúdos diferentes. Não existe o melhor indicador, o conjuto deles e a expertise da pessoa que negocia os ativos faz com que seja possível definir bons pontos de compra e venda.

Os tópicos subsequentes são responsáveis por apresentar alguns dos indicadores técnicos utilizados por especialistas em negociação de ativos.

2.2.1 Média Movel (MM)

A média móvel, do inglês *moving average* (MA), é um indicador utilizado como filtro de tendência, e quanto maior o período da média móvel, mais a tendência a longo prazo será analisada, e menos as variações de curto prazo estão sendo levadas em consideração.

A equação da média móvel do preço de fechamento pode ser expressa pela seguinte equação:

$$MA = \frac{\sum_{i=1}^{n} P_{\text{Close}_\text{n}}}{n} \tag{43}$$

Em que:

a) MA: Média Móvel

b) P_{Close}: Preço de Fechamento

c) n: Número de períodos analisados

2.2.2 Média Exponencial Móvel (MEM)

A média exponencial móvel, do inglês *Exponencial Moving Average* (EMA), é um indicador similar ao MA. O EMA é um filtro utilizado comumente para altas frequenciais, sendo possível analisar a variação de curto prazo dos preços.

A equação da média exponencial móvel do preço de fechamento pode ser calculada pela seguinte equação:

$$EMA_\text{n} = P_{\text{Close}_\text{n}} FS + [EMA_{\text{n-1}} (1 - FS)] \tag{44}$$

Em que:

a) EMA: Média Exponencial Móvel

b) P_{Close}: Preço de Fechamento

c) FS: Fator de suavização

d) n: Número de períodos analisados

2.2.3 Índice de Força Relativa (IFR)

O Índice de Força Relativa, também conhecido como *Relative Strength Index* (RSI) é um oscilador que mede a força de uma tendência em porcentagem, ou seja, o quão rápido o valor está seguindo uma tendência de alta ou de baixa, comumente aplicado para um período de 14 amostras anteriores (KIM, 2015).

O RSI pode ser calculado a partir da seguinte equação:

$$RSI = 100 - \frac{100}{1 + \frac{AG}{AL}} \tag{45}$$

Em que:

a) RSI: Índice de Força Relativa

b) AG: Média de ganho em *n* períodos

c) AL: Média de perda em *n* períodos

2.2.4 Chaikin Accumulation Distribution (AD)

O AD é um indicador de volume proposto por Marc Chaikin em 1975, o qual utilizou como base, a equação WVAD (Williams Variable Accumulation Distribution, indicador calculado, utilizando-se de valores de fechamento e abertura do ativo) como base para criar um índice denominado Accumulation Distribution (AD) usando os valores de preço máximo, mínimo e fechamento de cada dia. A equação 46 descreve o calculo do AD (KNUTH, 2020).

$$AD = Volume\frac{(P_{\text{Close}} - P_{\text{Low}}) - (P_{\text{High}} - P_{\text{Close}})}{(P_{\text{High}} - P_{\text{Low}})} \tag{46}$$

Em que:

a) Volume: Volume

b) P_{Close}: Preço de Fechamento

c) P_{Low}: Preço Mínimo

d) P_{High}: Preço Máximo

2.2.5 Chaikin Accumulation Distribution Oscilator (ADOSC)

O ADOSC é um indicador que utiliza como base de cálculo o AD. O ADOSC é uma ferramenta para geração de sinais de compra e venda quando a ação é comparada com a movimentação do preço (RANGANATHAM, 2006).

A equação 47 descreve o cálculo do ADOSC.

$$ADOSC = (AD_{\text{EMA-3}} - AD_{\text{EMA-10}}) \tag{47}$$

Em que:

a) $AD_{\text{EMA-3}}$: Média Exponencial Móvel de 3 do AD.

b) $AD_{\text{EMA-10}}$: Média Exponencial Móvel de 10 do AD.

2.2.6 On-Balance Volume (OBV)

O OBV foi criado pelo Joseph Ganville. A ideia central é quantificar as operações de compradores e vendedores. Nos dias em que os preços fecham em alta, leva-se em consideração, que todo o volume é assumido pelos compradores, e nos dias em que os preços fecham em baixa, supõe-se que o volume é atribuido aos vendedores. O volume então é somado ou subtraído de um valor cumulativo, tendo assim o cálculo do OBV (WILEY, 2015).

Simplificando, pode-se calcular o OBV aplicando a equação 48.

$$OBV_{\text{t}} = OBV_{\text{t-1}} + \frac{Close_{\text{t}} - Close_{\text{t-1}}}{|Close_{\text{t}} - Close_{\text{t-1}}|} * Volume_{\text{t}} \tag{48}$$

A seção 2.2 foi escrita com o objetivo de explorar e embasar o conteúdo desenvolvido, com o propósito de fundamentar os conceitos e os indicadores utilizados por investidores a fim de que possam executar as tomadas de decisão de compra ou venda dos ativos. Para cada indicador existe uma estratégia de utilização que não foi abordada porque o objetivo do trabalho é construir um sistema que utilize tais informações para compor uma estratégia única, e não desenvolvida por investidores do mercado financeiro. O próximo capítulo aborda alguns trabalhos base que tem objetivos ou técnicas similares as abordadas nesta qualificação.

3 REVISÃO BIBLIOGRÁFICA

A avaliação de séries temporais e a capacidade de predizer e modelar padrões utilizando funções matemáticas cujo intuito é obter vantagens econômicas pressupondo o que deverá acontecer no futuro, é de interesse da comunidade de pesquisa em mercado de ações.

Existem linhas de pesquisas diferentes para modelagem das séries temporais que descrevem o preço das ações, algumas delas utilizam de ferramentas clássicas da estatística, tais como: ARIMA, ARMA, GARCH, ARCH encontradas em trabalhos como: (BANERJEE, 2014), (FRANSES; VAN DIJK, 1996) e (WILEY, 2011); outras utilizam sistemas híbridos, parte estatístico (utilizando ARIMA, ARMA, GARCH ou ARCH) e outra parte que faz uso de algoritmos de aprendizagem supervisionado, tais como SVM (PAI; LIN, 2005) e Redes Neurais Artificiais (RATNAYAKA et al., 2015); por fim, existem trabalhos que aplicam sistemas de aprendizado supervisionado e aprendizado por reforço, tais como: Redes Neurais Convolucionais (RNC), Redes Neurais Recorrentes (RNR) implementadas em trabalhos tal como: (SEZER; OZBAYOGLU, 2018), (HSIEH; HSIAO; YEH, 2011) e (LELE, 2020).

Dentre todas as linhas de pequisa descrita acima, este trabalho concentra-se em modelos baseados em aprendizado por reforço. Os trabalhos estudados a seguir, baseiam-se em aprendizado por reforço, com foco em compra e venda de ações.

3.1 A Multi-agent Q-learning Framework for Optimizing Stock Trading Systems

A proposta abordada por Lee e O (2002) é similar à ideia central abordada por este trabalho e será discutido em mais detalhes na seção 4.

Lee e Jangmin propuseram um sistema de aprendizado por reforço *multi-agent* baseado em Q-Learning. A ideia central é modelar um sistema de ação em função do ambiente a fim de se obter a maior recompensa possível, diferentemente de modelos de aprendizado supervisionado baseados em regressão.

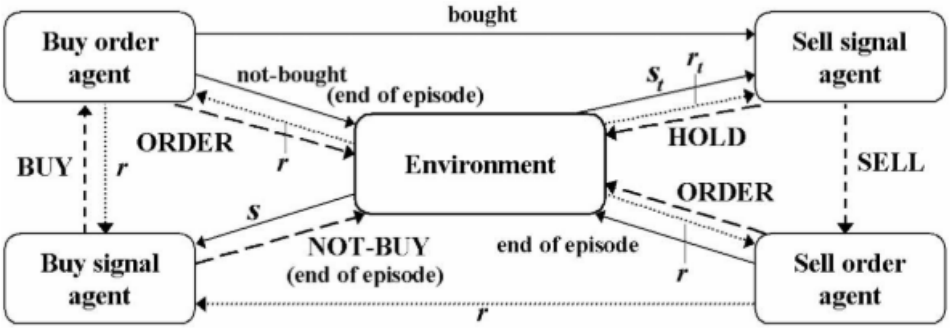
A arquitetura base do trabalho pode ser vista na figura 6.

O sistema é baseado em quatro agentes principais, cada um tem responsabilidades diferentes dentro do sistema que interage com o ambiente.

Buy signal agent: realiza previsões estimando as informações de longo e curto prazo dos estados das ações para produzir sinais de compra.

Buy order agent: determina os preços das ações que serão compradas.

Figura 6 – Arquitetura proposta por Lee e O (2002)



Fonte: Lee e O (2002)

Sell signal agent: realiza a previsão estimando as informações de longo e curto prazo dos estados das ações e os lucros atuais, para produzir sinais de venda.

Sell order agent: determina os preços das ações que serão vendidas.

O espaço de estado, ou seja, o valor do preço das ações e dos índices abordados por Lee e O (2002) são discretizados utilizando como base a tabela 1.

Tabela 1 – Tabela para discretização proposta por Lee e O (2002)

Profit	Coding	Profit	Coding
$+(0 \sim 5)$	00000001	$-(0 \sim 3)$	00010000
$+(5 \sim 12)$	00000010	$-(3 \sim 7)$	00100000
$+(12 \sim 20)$	00000100	$-(7 \sim 12)$	01000000
$+(20 \sim 30)$	00001000	$-(12 \sim 20)$	10000000

Fonte: Lee e O (2002)

A arquitetura proposta na figura 6 contém quatro agentes na qual o reforço do ambiente para os agentes é função do lucro obtido na operação e, apenas no momento da venda, o reforço é aplicado para todos os agentes do sistema.

Lee também propôs um trabalho muito similar em 2007, chamado de *"A Multiagent Approach to Q-Learning for Daily Stock Trading"* e segue a base de aprendizado supervisionado utilizando Q-Learning *multi-agent* utilizando a discretização dos estados estudados (LEE et al., 2007).

3.2 An Application of Deep Reinforcement Learning to Algorithmic Trading

O trabalho proposto por Thibaut Théate (2020) é baseado em uma arquitetura de aprendizado por reforço, utilizando aprendizado por reforço profundo, do inglês, *Deep Reinforcement Learning* (DRL), e, tal variação, proporcionada pelo autor, fez com que fosse denominada TDQN (Trading Deep Q-Network).

O espaço de estados proposto pelo autor são todas as informações que poderiam impactar o preço da ação. O algoritmo deveria ser capaz de distinguir quais das informações serão mais relevantes para a tomada de decisão. O espaço de estados é definido pelas informações: posição atual de negociação, número de ações detidas pelo agente, caixa disponível, OHLC (Open-High-Low-Close) da ação em um tempo *t*, MACD, RSI, ADX, taxa de juros, taxa de cambio, notícias, entre outros indicadores possíveis nos espaços abordados por (THIBAUT THÉATE, 2020).

O espaço de ações consiste em ações discretas, com um único agente responsável por executar: compra, venda e espera.

A recompensa dada pelo ambiente é função do lucro diário das operações, não são considerados apenas os instantes de compra e venda para ter-se a recompensa.

3.3 Adaptive Stock Trading Strategies with Deep Reinforcement Learning Methods

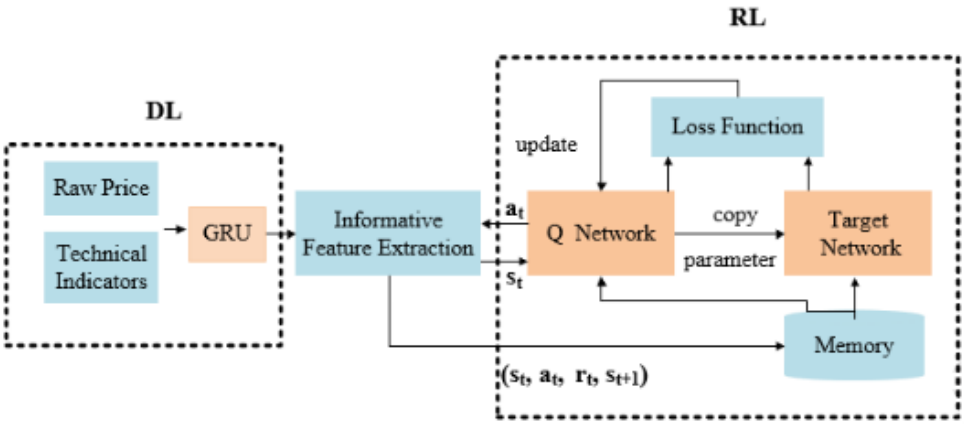
O trabalho proposto por Wu et al. (2020) contém uma arquitetura mais complexa onde se utiliza um único agente. Os autores propuseram os testes empregando dois algoritmos de aprendizado por reforço: o GDQN (Gated Deep Q-learning trading strategy) e o GDPG (Gated Deterministic Policy Gradient trading strategy). Tais algoritmos mostraram-se mais lucrativos que a estratégia denominada Turtle Trading System e de métodos mais simples de aprendizado supervisionado (DRL).

O espaço de estados é constituído a partir do preço de abertura, fechamento, máximo, mínimo e de indicadores relacionados ao preço, tais como: MA, EMA, MACD, BIAS, VR, OBV em que o autor utiliza uma Rede Neural Recorrente (GRU) para extrair as informações relevantes desses dados de modo a estruturar o espaço de estados.

O retorno do ambiente é função do lucro fornecido pela operação e também do risco da transação. Composto pelo índice Sortino Ration (SR) que é uma variação do índice Sharp Ration.

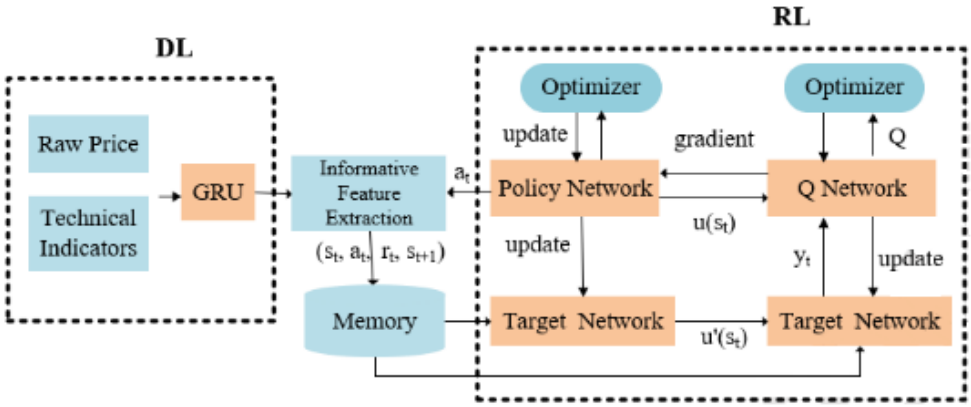
As arquiteturas propostas são apresentadas de forma simplificada nas figuras 7 e 8, onde são apresentados os modelos GDQN e GDPG, respectivamente, os quais baseiam-se em crítico (GDQN) e ator-crítico (GDPG).

Figura 7 – Arquitetura GDQN proposta por Wu et al. (2020)



Fonte: Wu et al. (2020)

Figura 8 – Arquitetura GDPG proposta por Wu et al. (2020)



Fonte: Wu et al. (2020)

Toda a parte experimental do trabalho foi realizada usando os dados históricos dos ativos do mercado dos Estados Unidos, Reino Unido e China.

3.4 Practical Deep Reinforcement Learning Approach for Stock Trading

O trabalho apresentado em Xiong et al. (2018) tem como objetivo a construção de um modelo de RL aplicado a negociação de ativos do mercado financeiro. Uma motivação da pesquisa dada pelos autores é que a capacidade de alocação de capital e negociação dos ativos

é primordial para a vitalidade de uma empresa de investimentos, ou seja, esse tipo de pesquisa pode contribuir para a manutenção e otimização do lucro de empresas do setor. Os autores propõem uma solução para o problema de negociação de ativos empregando o DDPG com o noise, como técnica de *exploration* para negociar os ativos que compõem o indice Dow Jones.

A modelagem do trabalho foi fundamentada no *mdp* e posteriormente foi formulado o objetivo comercial como um problema de maximização aplicando DRL.

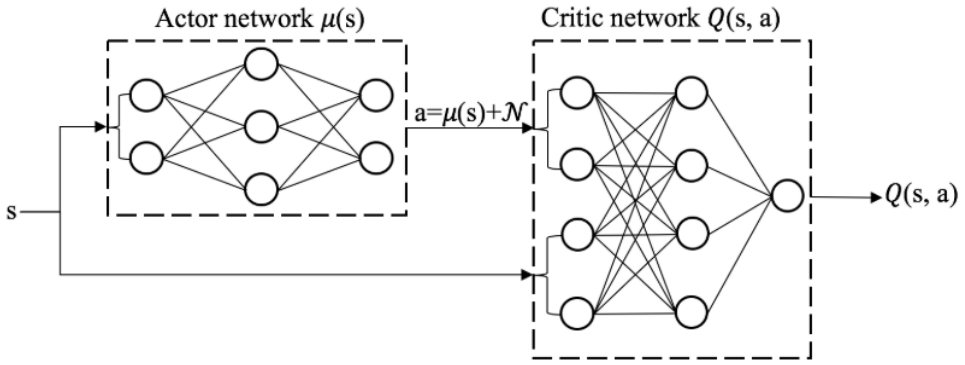
O espaço de estados da modelagem é caracterizado como multidimensional e contínuo, composto pelas informações de: preço dos ativos, quantidade de ativos comprados anteriormente, saldo restante.

As ações do agente são discretas, sendo elas: compra, venda, espera.

O reforço é função do lucro obtido a partir das ações executadas pelo agente.

A arquitetura das redes propostas está apresentada na figura 9.

Figura 9 – Arquitetura DDPG proposta por Xiong et al. (2018)



Fonte: Xiong et al. (2018)

Os dados de treinamento foram amostrados do período de 01/01/2009 até 01/01/2015, foi utilizado um ano de validação e cerca de um ano e meio para avaliação da performance do algoritmo.

A imagem da figura 10 faz uma análise comparativa da abordagem utilizando DDPG frente ao DJIA e ao Min-Var. O DDPG mostrou-se superior ao DJIA e ao Min-Var.

Figura 10 – Comparativo entre DDPG, DJIA e Min-Var proposta por Xiong et al. (2018)

Fonte: Xiong et al. (2018)

3.5 Application of deep reinforcement learning in stock trading strategies and stock forecasting

O trabalho apresentado em Li, Ni e Chang (2020) propõem uma análise comparativa entre três algoritmos de DRL, são eles: DQN, Double DQN, Dueling DQN utilizando o ϵ-greedy como técnica de *exploration*.

Os experimentos foram executados utilizando dez ativos selecionados de forma aleatória. A base de treinamento e validação foi dividida em uma proporção de 4:6.

O espaço de estados é contínuo e multidimensional, composto por: MACD, KAMA, osciladores, PSAR, ROC, VWAP entre outros.

O espaço de ações do agente é discreto, podendo executar apenas compra, venda e espera.

O autor não explicita a equação utilizada para o cálculo do reforço do agente, porém fica implícita a utilização do reforço como função do lucro obtido pelo agente na transação dos ativos escolhidos.

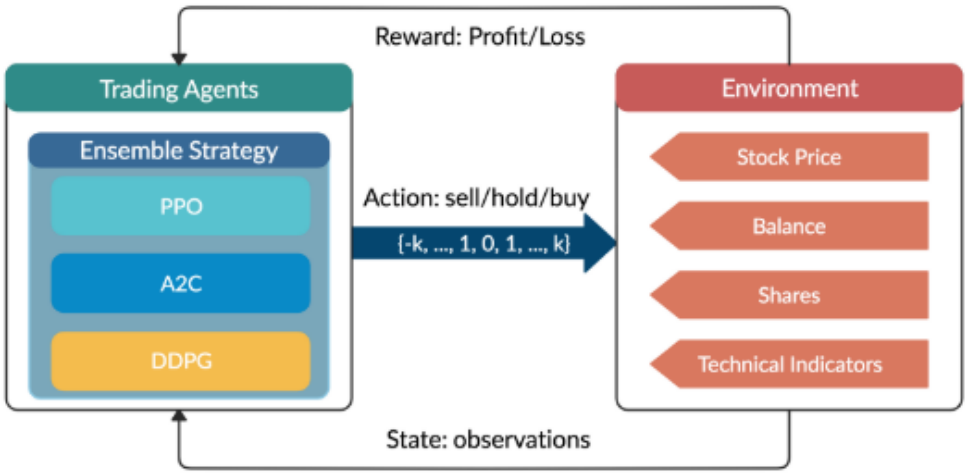
Na conclusão apresentada pelos autores são ressaltados resultados positivos, em sua maioria, o que viabiliza a aplicação de modelos de DRL aplicado em negociação de ativos. Uma

conclusão importante apresentada pelo autor é de que os algoritmos Double DQN e Dueling DQN apresentam uma performance melhor do que o DQN em ambientes de jogos, porém no ambiente de negociação dos ativos o DQN teve uma performace geral melhor do que suas optimizações (Double DQN, Dueling DQN).

3.6 Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy

O projeto apresentado em Yang et al. (2020) propõe uma arquitetura mista (figura 11) composta por três agentes que constituem a unidade principal de negociação. Os agentes são implementados utilizando os algoritmos A2C, DDPG e PPO.

Figura 11 – Arquitetura sistema RL proposta por Yang et al. (2020)



Fonte: Yang et al. (2020)

O objetivo dos autores foi criar uma solução altamente robusta empregando o melhor dos três algoritmos propostos em função do tempo de análise, sendo assim, ao decorrer do tempo todos os agentes são treinados e o agente que tiver o melhor SR é selecionado para executar as operações do próximo período. Ao fim do período de negociação, todos os agentes são treinados novamente e seleciona-se o melhor, em função do SR.

As ações possíveis são contínuas, escaladas de [-1, 1], e proporcionais ao número de ativos que é possível comprar naquele estado.

O espaço de estados é um vetor espaçado no tempo em 181 posições, composto por: saldo disponível, preço de fechamento ajustado, quantidade de ativos pertencentes ao agente, MACD, RSI, CCI, ADX.

Os resultados apresentados pelos autores são positivos (tabela 2), foi realizada uma análise individual utilizando apenas o PPO, A2C e o DDPG em relação ao modelo híbrido proposto inicialmente. Ambas as abordagens tiveram um retorno positivo no tempo, porém o PPO teve maior retorno acumulativo e o modelo híbrido teve o segundo melhor retorno acumulativo com o melhor SR.

Tabela 2 – Comparação e avaliação de desempenho apresentada em Yang et al. (2020)

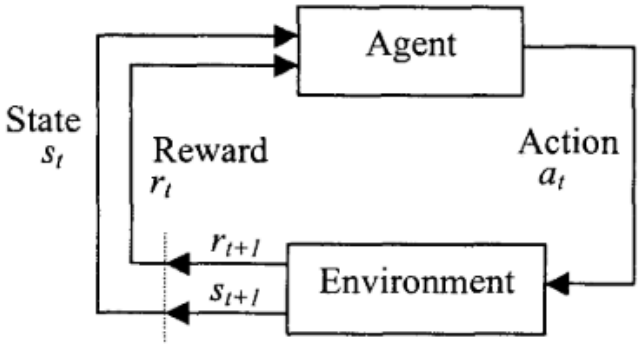
(2016/01/04-2020/05/08)	Ensemble (Ours)	PPO	A2C	DDPG	Min-Variance	DJIA
Cumulative Return	70.4%	83.0%	60.0%	54.8%	31.7%	38.6%
Annual Return	13.0%	15.0%	11.4%	10.5%	6.5%	7.8%
Annual Volatility	9.7%	13.6%	10.4%	12.3%	17.8%	20.1%
Sharpe Ratio	1.30	1.10	1.12	0.87	0.45	0.47
Max Drawdown	-9.7%	-23.7%	-10.2%	-14.8%	-34.3%	-37.1%

Fonte: Yang et al. (2020)

3.7 Stock Price Prediction Using Reinforcement Learning

O projeto de Lee (2001) apresenta uma arquitetura mais simples em relação as apresentadas anteriormente, ela é a aplicação direta de um modelo de RL e pode ser vista na figura 12. Os autores fundamentam a utilização de um modelo de RL em comparação com sistemas de aprendizado supervisionado, porque este tem limitações e não é adequado para aprender problemas com objetivos de longo prazo e recompensas atrasadas. A proposta não aborda um modelo DRL, os autores utilizaram um modelo TD.

Figura 12 – Arquitetura sistema RL proposta por Lee (2001)



Fonte: Lee (2001)

O espaço de estados é contínuo e constituído das informações de preço (OHLC), volume, variação de preço, variação de volume, MA, MACD, EOM, momentum, RSI, osciladores de volume e estocástico.

O retorno é função do lucro obtido na operação.

As ações são discretas, sendo possível realizar compra ou venda.

O autor não apresenta uma análise comparativa entre *buy-and-hold* e a proposta abordada por ele, porém afirma que há a viabilidade na geração de sinais de compra e venda.

3.8 A Multiagent Approach to Q-Learning for Daily Stock Trading

O projeto apresentado em Lee et al. (2007) é uma optimização do projeto prévio feito pelos autores em Lee e O (2002).

A proposta apresentada pelos autores é denominada de MQ-Trader, resumindo-se em uma aplicação MARL com foco em negociação de ativos aplicando Q-Learning com ϵ-greedy como técnica de *exploration*.

A arquitetura do projeto consiste em quatro agentes com responsabilidades distintas (Figura 13), sendo elas:

Buy signal agent: realiza previsões estimando as informações de longo e curto prazo dos estados das ações para produzir sinais de compra.

Buy order agent: determina os preços das ações que serão compradas.

Sell signal agent: realiza a previsão estimando as informações de longo e curto prazo dos estados das ações e os lucros atuais, para produzir sinais de venda.

Sell order agent: determina os preços das ações que serão vendidas.

O espaço de estados é discretizado e composto por: OHLC, MA, Gradiente MA e a distância normalizada.

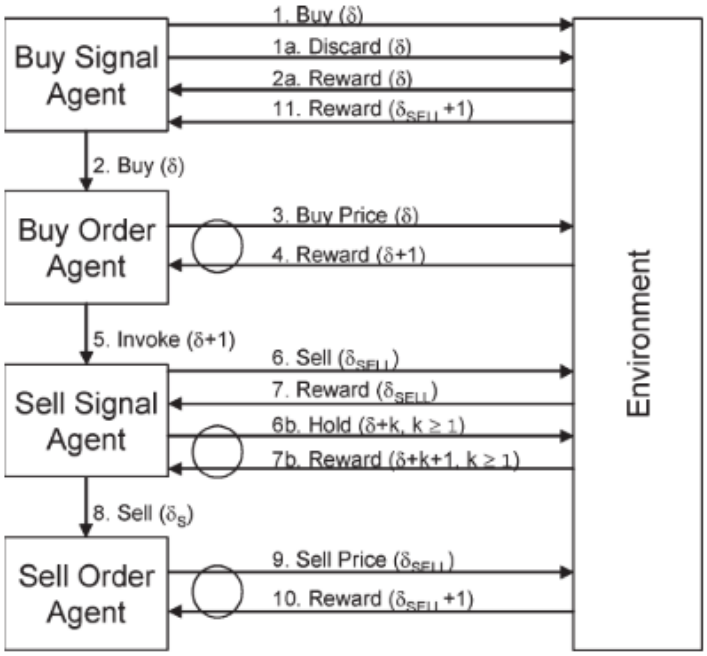
Os agentes podem efetuar apenas ações discretas, sendo elas compra, venda e espera.

Ao termino do trabalho, os autores afirmam que a abordagem proposta por eles são positivas quando aplicadas no mercado de ações coreano.

3.9 FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance

O projeto apresentado por Liu et al. (2020) propõem um paradigma interessante, além da pesquisa que foi executada pelos autores, todos os códigos utilizados foram publicados no GitHub e foi desenvolvido um *framework* de trabalho relacionados a área de finanças, chamado de FinRL. A abordagem é extremamente interessante por contribuir com o cenário de pequisa de

Figura 13 – Arquitetura do modelo MARL analisado em Lee et al. (2007)



Fonte: Lee et al. (2007)

forma extremamente positiva, possibilitando a comparação com diversos algoritmos estudados atualmente, ambientes distintos, entre outras funcionalidades apresentadas no trabalho.

O *framework* desenvolvido possibilita testes em compra e venda de ativos e alocação de portfólio. Os algoritmos possíveis são: DQN, Double DQN, Dueling DQN, DDPG, A2C, PPO, SAC, TD3 e MADDPG.

O espaço de estados dos agentes é composto por: saldo em conta, número de ações compradas, OHLC, volume, MACD, RSI, entre outros.

O espaço de ações pode ser discreto ou contínuo, podendo limitar em compra, venda e espera; ou aplicar um modelo proporcional que possibilita a compra ou venda de *k* ativos.

O reforço do agente é função do lucro financeiro e/ou do SR.

Ao termino do trabalho os autores apresentam resultados positivos provenientes do *framework* desenvolvido. Mais detalhes dos resultados pode ser analisados na tabela 3. Os valores à esquerda das análises (em laranja) são provenientes da técnica de negociação de múltiplos ativos e os valores à direita (em azul) são provenientes da técnica de alocação de portfólio.

Tabela 3 – Resultados apresentados pelo FinRL Liu et al. (2020)

2019/01/01-2020/09/23	TD3	DDPG	Min-Var.	DJIA
Initial value	1,000,000	1,000,000	1,000,000	1,000,000
Final value	1,403,337; 1,381,120	1,396,607; 1,281,120	1,171,120	1,185,260
Annualized return	21.40%; 17.61%	20.34%; 15.81%	8.38%	10.61%
Annualized Std	14.60%; 17.01%	15.89%; 16.60%	26.21%	28.63%
Sharpe ratio	1.38; 1.03	1.28; 0.98	0.44	0.48
Max drawdown	11.52% 12.78%	13.72%; 13.68%	34.34%	37.01%

Fonte: Liu et al. (2020)

3.10 Amulti-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning

O projeto apresentado em Carta et al. (2020) discute sobre a limitação apresentada em modelos de aprendizado supervisionado aplicado em negociação de ativos e, o motivo pelo qual, o aprendizado por reforço pode auxiliar a solução desse problema.

A proposta dos autores baseia-se em um modelo hibrido que contém uma etapa de pré-processamento dos dados utilizando redes neurais profundas, para em fim acoplar ao sistema de aprendizado por reforço.

Pode-se dividir o projeto nas seguintes camadas (A figura 14 ilustra a arquitetura desenvolvida pelos autores):

a) *Camada de Pré-Processamento*: responsável por converter os dados temporais das séries em imagem a ser processada pelas próximas etapas;

b) *Meta aluno*: Camada que utiliza as imagens artificiais criadas na etapa de pré-processamento e aplica aprendizado por reforço, neste caso, os autores aplicam Deep Double Q-Learning com o ϵ-greedy.

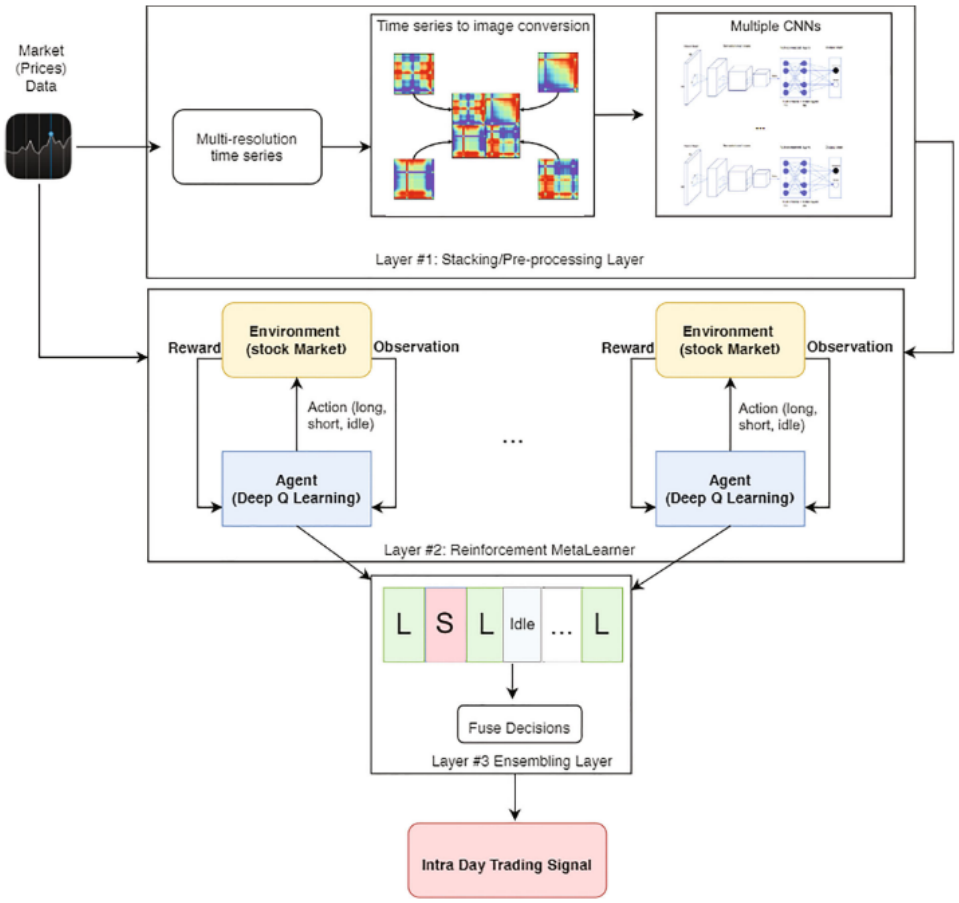
c) *Camada de Agrupamento*: Responsável por fundir diferentes sinais de diferentes interações de treinamento do meta aluno para obter a decisão final.

As ações dos agentes são discretas e podem ser dividas em: operar comprado, operar vendido e esperar.

O reforço fornecido é função do lucro financeiro obtido em relação à categoria de operação que está sendo executada: comprado ou vendido.

Ao término do projeto o autor faz uma comparação da abordagem proposta em relação a outras técnicas (CNN-TAr, LS-STM, Buy-and-Hold) no período de 2009 até 2015 em que

Figura 14 – Arquitetura proposta em Carta et al. (2020)



Fonte: Carta et al. (2020)

essa supera todos os outros modelos. A figura 4 apresenta os resultados obtidos a partir dos experimentos executados pelos autores.

Tabela 4 – Resultados apresentados em Liu et al. (2020)

	JPM stock	MSFT stock	*S&P*500 future
OUR PROPOSAL	**11.3%**	**17.53%**	**23.20%**
CNN-TAr	9.19%	4.44%	*n.a.*
LS-STM	*n.a.*	*n.a.*	~ 10.15%
Buy-and-Hold	-1.67%	-1.93%	21.33%

Fonte: Liu et al. (2020)

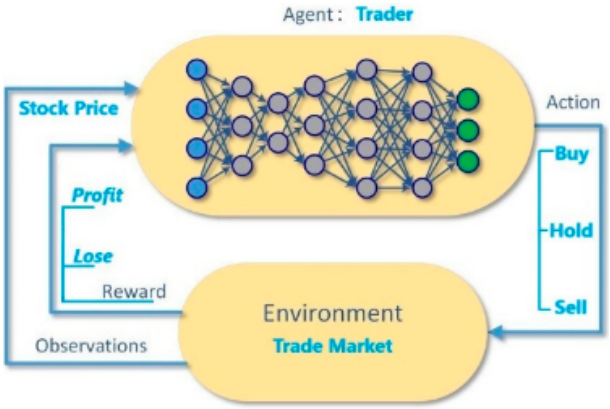
3.11 Recommending Cryptocurrency Trading Points with Deep Reinforcement Learning Approach

O projeto apresentado em Sattarov et al. (2020) expõem uma abordagem focada em criptomoeda utilizando Reinforcement Learning para negociação de moedas como: Bitcoin, Litecoin e Ethereum.

Os autores não deixam claro o algorítimo que foi empregado, falando superficialmente que foi aplicado DRL, deixando implícito a utilização de algoritmos da família do DQN.

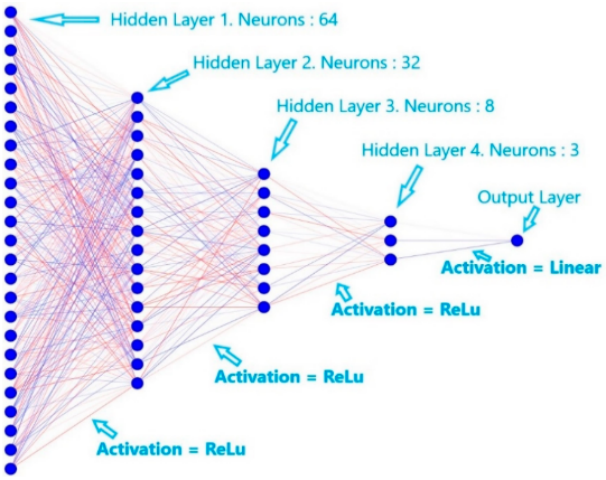
A arquitetura do projeto fundamenta-se em um modelo DRL simples (visto na figura 15) utilizando a rede proposta na figura 16.

Figura 15 – Arquitetura proposta em Sattarov et al. (2020)



Fonte: Sattarov et al. (2020)

Figura 16 – Modelo de rede proposto em Sattarov et al. (2020)



Fonte: Sattarov et al. (2020)

As ações possíveis dadas ao agente são compra, venda e espera.

O reforço é função do lucro obtido na operação.

Ao termino do projeto, os autores concluem que a proposta é viável e o modelo baseado em DRL é possível para operar criptomoedas, sendo lucrativo essas operações. As imagens 17 e 18 apresentam os resultados obtidos com Litecoin e Ethereum, respectivamente.

Figura 17 – Resultados obtidos experimentais utilizando Litecoin

	Invested Money ($)	Number of Actions	Money after Trading ($)	Quality of Trading %
Double Cross Strategy	10 814	12	10 576	97.8 (Lost 2.2)
Swing Trading	10 814	4	11 217	103.6 (Grew 3.6)
Scalping Trading	10 814	134	13 382	123.7 (Grew 23.7)
DRL Application	10 000	642	17 467	174.6 (Grew 74.6)

Fonte: Sattarov et al. (2020)

Figura 18 – Resultados obtidos experimentais utilizando Ethereum

	Invested Money ($)	Number of Actions	Money after Trading ($)	Quality of Trading %
Double Cross Strategy	11 739	9	11 400	97.1 (Lost 2.9)
Swing Trading	11 739	4	10 643	90.6 (Lost 9.4)
Scalping Trading	11 739	112	11 462	97.6 (Lost 2.4)
DRL Application	10 000	294	14 140	141.4 (Grew 41.4)

Fonte: Sattarov et al. (2020)

3.12 A Deep Reinforcement Learning Approach to Stock Trading

O trabalho apresentado em Gran, Holm e Søgård (2019) propõem uma abordagem utilizando DDPG e Noise, como técnica de *exploration*, aplicado a negociação dos índices: DJIA (USA), TSX (Canada), JSE (Africa do Sul) e SENSEX (India).

O espaço de estado do agente é contínuo e composto pelo retorno diário do ativo, volume e informações da ferramenta de acompanahmento de tendencias do Google (Google Trend) no que diz respeito ao ativo.

O espaço de ação do agente é contínuo. Os autores enfatizam que existem projetos na área que discretizam a compra e a venda dos ativos, porém o problema é essencialmente contínuo e com o DDPG é possível fazer operações desse tipo.

O reforço fornecido aos agentes é função do lucro financeiro obtido em todas as operações.

Para otimizar o processo de treinamento do agente, os autores utilizaram um modelo pré treinado utilizando Algoritmos Genéticos.

Ao término do estudo os autores chegam a resultados positivos aplicando o modelo proposto, superando em mais de 8% o *benchmarking* baseado em *buy-and-hold*.

3.13 Deep reinforcement learning for trading

O trabalho apresentado em Zhang, Zohren e Roberts (2019) utiliza os seguintes algoritmos de Reinforcement Learning: DQN, PG, A2C.

O espaço de estados é um vetor espaçado no tempo em 60 posições, composto pelo: preço, retorno em diferentes horizontes e indicadores técnicos (MACD e o RSI).

O estudo executado pelos autores propõem dois modelos de espaço de ações: um discreto, composto por {-1, 0, 1}, em que, -1 representa a posição máxima vendida e 1 representa a máxima posição comprada e 0 espera; o segundo, compõem a mesma ideia do discreto, porém pode-se variar de forma contínua entre os valores de [-1, 1].

O reforço dado ao agente é função do lucro e do risco.

Para os algoritmos estudados que são ator-crítico, a construção da rede neural, responsável pelo ator e pelo crítico, foi construída utilizando LSTM, podem ser vistos mais detalhes no projeto dos autores. O motivo da escolha da LSTM foi o aumento da utilização desta em projetos voltados a modelos financeiros.

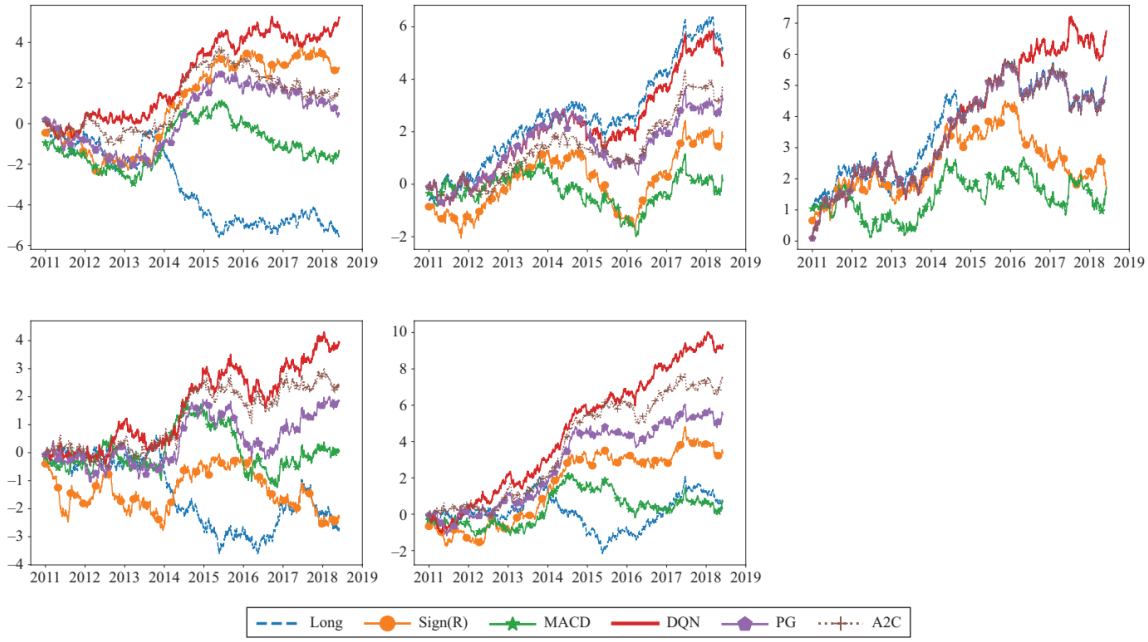
Ao término do projeto os autores apresentam resultados positivos as abordagens executadas utilizando RL. Os gráficos da figura 19 indicam que o algoritmo que obteve a melhor performance foi o DQN.

3.14 Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments

O trabalho Lowe et al. (2017) não faz relação alguma à parte econômica abordada neste trabalho, no entanto tem um importante trabalho comparativo de algoritmo de aprendizado por reforço, principalmente no algoritmo que será utilizado nesta pesquisa (abordado em mais detalhes na seção 4).

O ambiente proposto para comparação é constituído de n agentes e l marcos habitados em um mundo bidimensional com espaço de tempo discreto. Cada agente pode realizar ações

Figura 19 – Resultados apresentados em Zhang, Zohren e Roberts (2019)

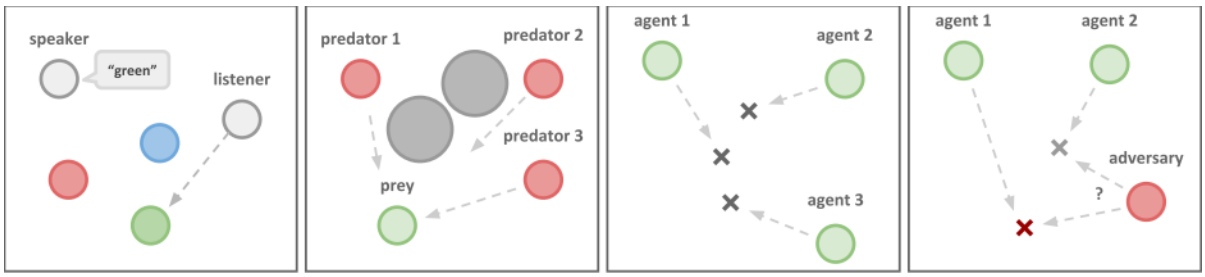


Fonte: Zhang, Zohren e Roberts (2019)

físicas e de comunicação no ambiente. O objetivo principal varia para testar a aderência do algoritmo a ambientes diferentes.

A figura 20 indica os ambientes em que foram executados os teste e os objetivos de cada abordagem.

Figura 20 – Modelos de Ambientes Propostos em Lowe et al. (2017)



Fonte: Lowe et al. (2017)

O primeiro ambiente (primeira imagem da esquerda para a direita da figura 20), propõem uma abordagem de comunicação cooperativa. A tarefa consiste em dois agentes, um locutor e um ouvinte, que são colocados em um ambiente com três marcos de cores diferentes. A cada episódio, o ouvinte deve navegar até um ponto de referência de uma determinada cor e obtem a recompensa com base na distância até o ponto de referência correto. No entanto, embora o ouvinte possa observar a posição relativa e a cor dos pontos de referência, ele não sabe para qual

ponto de referência deve navegar. Por outro lado, a observação do falante consiste na cor correta do ponto de referência e pode produzir uma saída de comunicação a cada passo de tempo que é observado pelo ouvinte. Assim, o falante deve aprender a produzir a cor de referência com base nos movimentos do ouvinte.

O segundo ambiente consiste na abordagem Predador-Presa. Onde N agentes cooperantes mais lentos devem perseguir o adversário mais rápido em torno de um ambiente gerado aleatoriamente com L grandes marcos impedindo o caminho. Cada vez que os agentes da cooperativa colidem com um adversário, os agentes são recompensados enquanto o adversário é penalizado. Os agentes observam as posições e velocidades relativas dos agentes e as posições dos pontos de referência.

O terceiro ambiente consiste em uma navegação cooperativa em que os agentes devem cooperar por meio de ações físicas para atingir um conjunto de L marcos. Os agentes observam as posições relativas de outros agentes e pontos de referência e são recompensados coletivamente com base na proximidade de qualquer agente a cada ponto de referência. Em outras palavras, os agentes têm que "cobrir" todos os marcos. Além disso, os agentes ocupam um espaço físico significativo e são penalizados quando colidem uns com os outros.

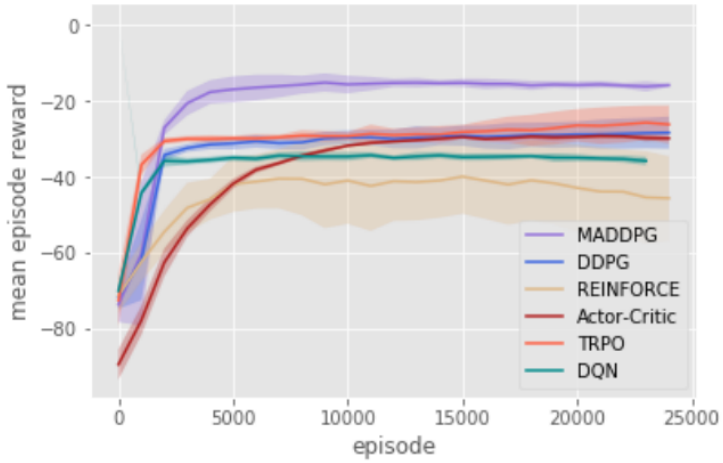
O quarto ambiente foi denominado de *Physical Deception* pelos autores em que N agentes cooperam para alcançar um único ponto de referência de um total de N pontos de referência. Eles são recompensados com base na distância mínima de qualquer agente ao alvo (portanto, apenas um agente precisa chegar ao ponto de referência do alvo). No entanto, um adversário solitário também deseja alcançar o marco-alvo. O problema é que o adversário não sabe qual dos marcos é o correto. Assim, os agentes cooperantes, que são penalizados com base na distância do adversário ao alvo, aprendem a espalhar e percorrer todos os marcos para enganar o adversário.

Dentre os algoritmos estudados, o MADDPG foi o que obteve a maior recompensa média por episódio (Mean Episode Reward), e uma convergência acentuada, como pode ser visto na figura 21.

A tabela 5 mostra a porcentagem de episódios nos quais os agentes atingiram seus objetivos utilizando comunicação cooperativa após 25.000 episódios.

A tabela comparativa traz alguns algoritmos estudados e citados neste trabalho, tais como:

Figura 21 – Repompensa Média por Episódio analisada em Lowe et al. (2017)



Fonte: Lowe et al. (2017)

Tabela 5 – Tabela de desempenho comparativo MADDPG

Agent π	**Target reach %**	**Average distance**
MADDPG	**84.0%**	**0.133**
DDPG	32.0%	0.456
DQN	24.8%	0.754
Actor-Critic	17.2%	2.071
TRPO	20.6%	1.573
REINFORCE	13.6%	3.333

Fonte: Lowe et al. (2017)

DQN (Deep Q-Networks) é um algoritmo de aprendizado por reforço que se baseia no valor da diferença temporal (TD) e tem como objetivo aprender a função ação-valor (Q). Mais detalhes podem ser vistos na seção 2.1.14.

DDPG (Deep Deterministic Policy Gradient) é um algoritmo de aprendizado por reforço baseado no DPG, com a variante de utilizar aprendizado profundo. Mais detalhes da implementação podem ser vistos na seção 2.1.16.

O MADDPG (Multi-Agent Deep Deterministic Policy Gradient) é uma variante do DDPG para sistemas *multi-agent*, os detalhes da implementação podem ser analisados em 2.1.17.

O Actor-Critic foi tratado de forma simples na seção 2.1.10, e seus fundamentos são aplicados em modelos mais complexos como o DDPG e o MADDPG.

o TRPO (Trust Region Policy Optimization) é um método baseado no gradiente da política, similar ao DDPG, mais detalhes de sua implementação e particularidades podem ser analisadas em Lapan (2018).

O REINFORCE, também conhecido como Monte Carlo Policy Gradients, é um algoritmo baseado no gradiente da política, detalhes de sua implementação podem ser observadas em Lapan (2018).

Importante frisar que os algoritmos utilizados para comparação não são essencialmente *multi-agent* e sim adaptados para executar tarefas multiagente.

3.15 Análise geral dos projetos

Tendo em vista a literatura e os principais projetos abordados no contexto acadêmico, esta seção visa listar alguns trabalhos similares a este e suas principais características.

A tabela 6 resume os artigos discutidos nas subseções anteriores de modo a analisar as principais características para que seja possível compreender a relevância e o contexto em que este trabalho se insere.

Dentre os trabalhos estudados anteriormente existem frentes de relevância diferentes, sendo eles, relevância por: similaridade de pesquisa, embasamento teórico, inspiração, inovação.

Os trabalhos mais relevantes quando se aborda a similaridade de pesquisa são Xiong et al. (2018), Gran, Holm e Søgård (2019), Yang et al. (2020) e Liu et al. (2020). Os trabalhos de Xiong et al. (2018), Gran, Holm e Søgård (2019) e Liu et al. (2020) apresentam abordagens com DDPG *single-agent*, porém fundamentam a possibilidade da utilização de algoritmos de gradiente de política para solucionar o problema de negociação de ativos. O trabalho de Yang et al. (2020) apresenta uma metodologia diferente, um misto entre algoritmos, porém o DDPG ainda faz parte desse misto. O Liu et al. (2020) apresenta uma abordagem pensando na comunidade de pesquisa e cria um *framework* de DRL, mesmo que em seu projeto não apresente explicitamente os resultados do MADDPG é possível modelar o problema utilizando o MADDPG.

Todos os trabalhos têm relevância no sentido de embasamento teórico, porém os mais inspiradores foram Lee e O (2002), Lee et al. (2007) e Wu et al. (2020). Existe uma motivação pessoal para o estudo de MARL e o trabalho de Lee (2001) e Lee et al. (2007) provam ser possível essa abordagem no contexto de negociação de ativos e o projeto de Wu et al. (2020) ilustra outras possibilidades mistas para construção das redes, no caso ele utilizou GRU.

Os trabalhos mais inovadores são: (CARTA et al., 2020) por construir um sistema totalmente diferente de todos os outros projetos estudados e juntar redes convolucionais com

aprendizado por reforço; por fim, os trabalhos de Yang et al. (2020) e Liu et al. (2020) por construírem uma abordagem de Risk-sensitive Reinforcement Learning utilizando o índice de turbulência financeira.

O trabalho mais simples e que apresenta menor relevância ao estudo feito nessa dissertação foi o do Lee (2001), talvez por ser o trabalho mais antigo e apresentar um modelo TD relativamente antigo no contexto atual que se utiliza DRL. Além desses pontos os autores deixam subentendido algumas questões da metodologia, método utilizado nos experimentos e os resultados não são tão claros, porém afirmam a viabilidade de modelos de RL para negociação de ativos do mercado financeiro.

Tabela 6 – Tabela comparativa das abordagem do RL em negociação de ativos

ID	Nome do Projeto	Ano de Publicação	Numero de Agentes	Aversão a Risco	Algoritmo	Técnica de Exploração	Reforço	Indicadores Técnicos	Espaço de Estados	Papeis Aplicados	Ação	Espaço de Ações	Numero de Episódios Treinados
1	Stock Price Prediction Using Reinforcement Learning	2001	1	Não Especificado	TD	e-greedy	Lucro	OHLC, Volume, MACD, EOM, Momentum, RSI, MA, Volume Oscilladores	Continuo	Internacional	Discreta	[Buy, Sell]	3000
2	A Multi-agent Q-learning Framevork for Optimizing Stock Trading Systems	2002	4	Não Especificado	Q-Learning	e-greedy	Lucro	Disparity; MA	Discreto	Internacional	Discreta	[Buy, Hold, Sell]	20000
3	A Multiagent Approach to Q-Learning for Daily Stock Trading	2007	4	Não Especificado	Q-Learning	e-greedy	Lucro	OHLC, MA, Gradiente MA, Distancia Normalizada	Discreto	Internacional	Discreta	[Buy, Hold, Sell]	Não Especificado
4	Practical Deep Reinforcement Learning Approach for Stock Trading	2018	1	Não Especificado	DDPG	Noise	Lucro	OHLC, quantidade de ativos e saldo restante	Contínuo	Internacional	Discreta	[Buy, Hold, Sell]	Não Especificado
5	A Deep Reinforcement Learning Approach to Stock Trading	2019	1	Não Especificado	DDPG	Noise	Lucro	Preço, Volume, Google Trend	Contínuo	Internacional	Continuo	[n, m]	Não Especificado
6	An Application of Deep Reinforcement Learning to Algorithmic Trading	2020	1	Não Especificado	DQN	e-greedy	Lucro	MACD, RSI; ADX	Contínuo	Internacional	Discreta	[Buy, Hold, Sell]	100
7	Adaptive Stock Trading Strategies with Deep Reinforcement Learning Methods	2020	1	Não Especificado	GDPG; GDQN	e-greedy	Lucro + Risco	OHLC, MA, EMA, MACD, BIAS, VR, OBV	Contínuo	Internacional	Discreta	[Buy, Hold, Sell]	200
8	Application of deep reinforcement learning in stock trading strategies and stock forecasting	2020	1	Não Especificado	DQN, DDQN, Dueling DDQN	e-greedy	Lucro	MACD, KAMA, Osciladores, PSAR, ROC, Momentum, VWAP e outros	Contínuo	Internacional	Discreta	[Buy, Hold, Sell]	Não Especificado
9	Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy	2020	3	Sim	PPO, A2C, DDPG	Não Especificado	Lucro + Risco	Saldo, Valor Comprado, Close Price, MACD, RSI, CCI, ADX	Contínuo	Internacional	Contínuo	[-1, 1]	Não Especificado
10	FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance	2020	1	Sim	DQN, Double DQN, Dueling DQN, DDPG, A2C, PPO, SAC, TD3, MADDPG	N.E	Lucro + Risco	Saldo, Valor Comprado, OHLC, Volume, MACD, RSI, entre outros	Contínuo	Internacional	Discreta ou Contínuo	[Buy, Hold, Sell] [-1, 1]	Não Especificado
11	A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning	2020	1	Não Especificado	DQN	e-greedy	Lucro	Não Especificado	Contínuo	Internacional	Discreta	[Long, Short, Idle]	Não Especificado
12	Recommending Cryptocurrency Trading Points with Deep Reinforcement Learning Approach	2020	1	Não Especificado	DQN	Não Especificado	Lucro	Preço	Contínuo	Cryptocurrency	Discreta	[Buy, Hold, Sell]	Não Especificado
13	Deep Reinforcement Learning for Trading	2020	1	Não Especificado	DQN, PG, A2C	N.E	Lucro + Risco	Preço, Retorno em diferentes horizontes, MACD, RSI,	Contínuo	Internacional	Discreta e Contínuo	[Long, Short, Idle] [-1, 1]	Não Especificado
14	Aprendizado Por Reforço Profundo Multiagente Aplicado a Negociação de Ativos de Mercado Financeiro	2021	2	Não	MADDG e DDPG	Noise	Lucro	OHLC, MA, EMA, RSI, AD, OBV e MACD	Contínuo	Mercado Brasileiro	Contínuo	[-1, 1]	50000

Fonte: Autor (2021)

4 METODOLOGIA

Tendo como base as referências analisadas na seção 3, este trabalho tem como objetivo propor uma arquitetura para um sistema de aprendizado por reforço, o qual irá interagir com um ambiente que possibilite operações de compra e de venda a descoberto.

A proposta consiste em um sistema *Multi Agent Reforcement Learning* (MARL) composto por dois agentes que cooperam entre si, cujo objetivo é obter o maior lucro no tempo, utilizando um recurso único, o capital investido. Os agentes são responsáveis por dizer quando e quanto deve ser investido em um único ativo analisado. A interação entre os dois agentes tem o que denominou-se de "Gestor de Carteira", ou seja, os dois agentes agem de forma a construir um sistema de gestão de carteira de investimentos, gerenciando qual porcentagem do capital será negociado e a porcentagem do capital que fica em caixa para possíveis oportunidades futuras. Busca-se que os agentes aprendam a investir em função do risco aprendido nas transações, ou seja, para um estado que pode-se ter ganho porém o risco é alto ele deve ser capaz de inferir um investimento menor. Vale ressaltar que não foi utilizada uma modelagem com Risk-sensitive Reinforcement Learning como apresentada em projetos como Yang et al. (2020) e Liu et al. (2020) em que a aversão a risco é modelada em função do índice de turbulência financeira.

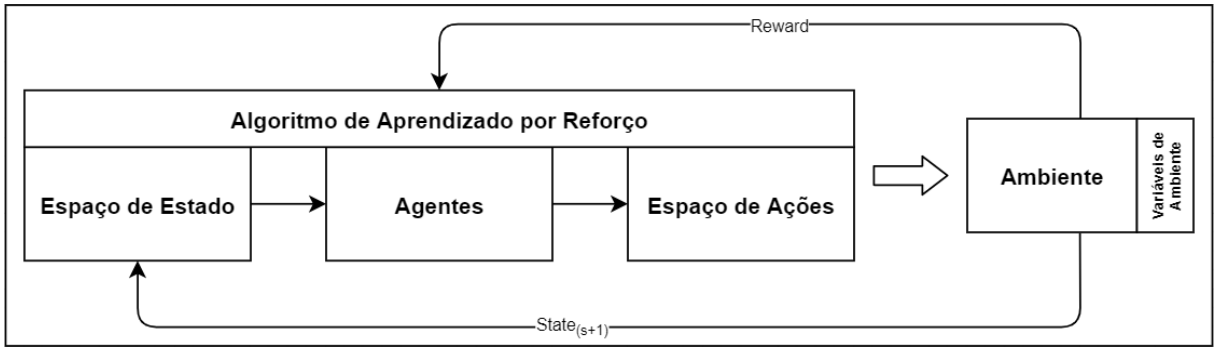
A modelagem do projeto foi executada aplicando algumas simplificações para que seja possível considerar o problema como um MDP. Simplificações similares são também apresentadas nos projetos analisados na Revisão Bibliográfica, sendo elas: dado um estado em que o agente se encontra, todas as informações necessárias para a tomada de decisão são fornecidas no estado atual, em que o agente tem visão global do mundo. Os indicadores relacionados ao preço tem a finalidade de trazer informações relevantes do passado ao estado atual. Partindo do princípio que os dados de preço e os indicadores relacionados ao preço são suficientes para que uma pessoa possa analisar e decidir qual a melhor decisão a ser tomada, infere-se que os agentes também devem ser capazes de executar tais análises, simplificando o modelo proposto a um MDP.

O objetivo dos agentes é o lucro, para isto tem-se um *fully cooperative environment* em que um agente não tem visão das ações do outro agente, porém recebem o mesmo reforço. Os dois agentes são igualmente propensos a risco e o motivo da utilização de dois agentes é a segmentação de responsabilidade, em que, um agente deverá aprender a operar em tendências de alta e outro a operar em tendências de baixa.

A escolha do algoritmo é fundamentada na premissa de que o sistema é um MDP estocástico com política determinística e os agentes devem ser capazes de executar ações contínuas em estados contínuos. Em função dessa necessidade tem-se os algoritmos de Política de Gradiente Deterministico Profundos e em virtude da modelagem ser *multi-agent*, escolheu-se o MADDPG.

A figura 22 é uma simplificação da arquitetura do trabalho, a qual será abordada em mais detalhes nos tópicos que sucedem esta proposta.

Figura 22 – Arquitetura simplificada do trabalho



Fonte: Autor (2021)

4.1 Algoritmo de Aprendizado por Reforço

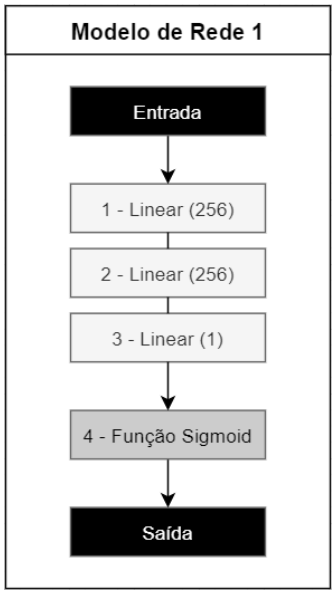
O algoritmo de aprendizado por reforço escolhido é o *Multi Agent Deep Deterministic Policy Gradients* (MADDPG) que é um algoritmo ator-crítico *multi-agent*. A escolha do algoritmo foi embasada no trabalho Lowe et al. (2017) através do qual foi feito um estudo que comprovou sua eficácia superior ao DDPG, REINFORCE, Actor-Critc, TRPO e DQN para um problema cooperativo. No entanto, não há nenhuma menção à efetividade do mesmo em problemas do setor econômico. Os detalhes do algoritmo podem ser vistos na revisão bibliográfica (tópico 2.1.17).

Uma segunda abordagem foi testada a fim de realizar uma análise comparativa, além do sistema multiagente utilizando o MADDPG, foi desenvolvido um sistema utilizando apenas um único agente empregando DDPG para avaliar o impacto do MARL na solução proposta.

Ambos os algorítimos (MADDPG e o DDPG) baseam-se em ator crítico. Na implementação foram construidas duas propostas de rede: uma utilizando uma rede MLP e outra que utiliza uma camada Bi-LSTM no início da rede, responsável por abstrair a temporalidade do

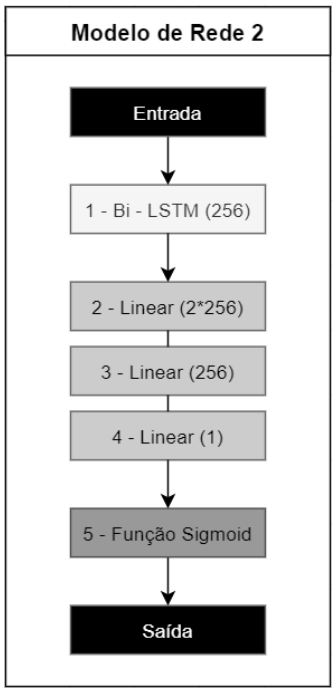
problema proposto. As figuras 23 e 24 ilustram as redes utilizadas, sem ou com uma camada Bi-LSTM, respectivamente.

Figura 23 – Modelo de rede: MLP (Ator e Crítico)



Fonte: Autor (2021)

Figura 24 – Modelo de rede: Bi-LSTM (Ator e Crítico)



Fonte: Autor (2021)

4.2 Espaço de Estados

O espaço de estado que será abordado na proposta diz respeito à análise dos dados de preço das ações que compõem o índice IBOVESPA.

Existem duas categorias de dados que serão trabalhados. A primeira são os dados de preço e volume do ativo negociado: abertura, fechamento, máximo, mínimo e volume diário negociado. A segunda são os dados referentes aos indicadores relacionados ao preço e volume: MA, EMA, RSI, AD, OBV e MACD.

Após a aquisição e cálculo desses dados, há duas formas de prosseguir com a criação do espaço de estados. A primeira diz respeito à utilização dos dados no instante de tempo t. A segunda consiste em deslocar n dados anteriores e trazê-los ao instante de tempo t para executar a análise. Em suma, pode-se ter os dados no ponto de análise ou os dados deslocados. Neste projeto realizamos uma comparação entre ambas as abordagens de construção do espaço de estados.

4.3 Agentes

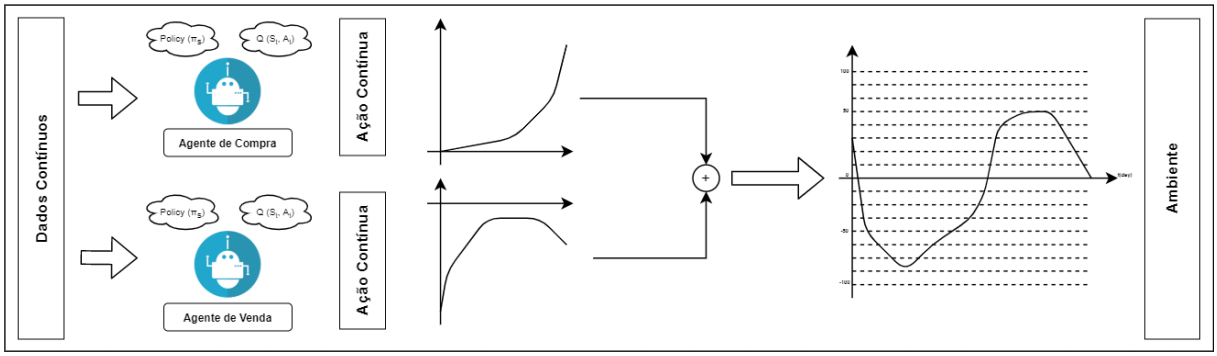
O trabalho foi abordado utilizando MARL, e é composto por dois agentes. O primeiro agente é responsável pelo estímulo de compra e pode decidir se compra ou não um determinado papel, como também a intensidade de compra em relação ao montante de investimento. O segundo agente é similar ao primeiro, porém é responsável pelo estímulo de venda, o qual decide se deve ou não vender e a intensidade da venda em relação ao montante de investimento.

A cooperação entre esses dois agentes combinam um sinal que diz a porcentagem do montante total que será investido, ou seja, o que será comprado ou vendido, sendo válidas operações de compra e de venda a descoberto.

A interação entre os dois agentes caracteriza o que denominou-se de "Gestor de Carteira", ou seja, os dois agentes agem de forma a construir um sistema de gestão de carteira de investimentos, gerenciando qual porcentagem do capital será negociado, ou se mantem em caixa para possíveis oportunidades futuras. Essa interação está ilustrada na figura 25.

Um segundo modelo foi proposto a fim de realizar uma análise comparativa. Este é constituído de apenas um agente responsável por executar as ações de compra ou de venda a descoberto, as ações deste único agente são proporcionais a ação conjunta do sistema multiagente.

Figura 25 – Interação entre agentes e ambiente



Fonte: Autor (2021)

4.4 Espaço de Ações

O agente responsável pela compra informa ao sistema a "força dos compradores", tal valor será escalado $[0 \rightarrow 1]$ e o agente responsável pela venda informa ao sistema a "força dos vendedores"que será escalado $[0 \rightarrow -1]$. A soma dos dois sinais resulta na operação que será realizada (equação 49).

$$A_{\text{joint}} = A_{\text{buy}} + A_{\text{sell}} \tag{49}$$

O sinal resultante pode variar de 1 até -1, e os valores positivos têm uma representação porcentual da compra a qual será executada no ativo. Valores negativos representam o porcentual de venda a descoberto que será executado no ativo.

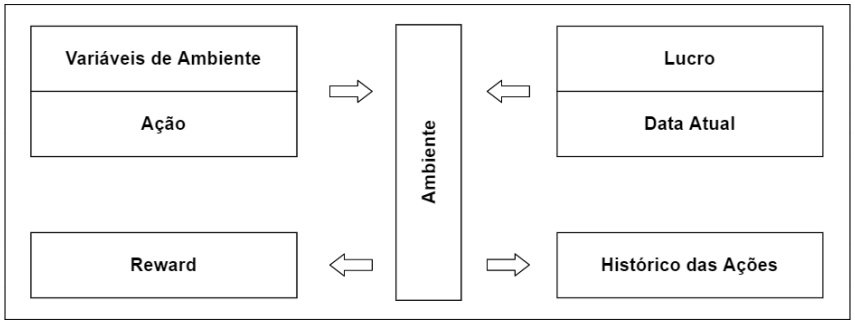
No segundo modelo, o único agente pode executar ações que variam de 1 até -1, sendo equivalente ao resultado da ação conjunta proposto no modelo multiagente.

4.5 Ambiente

O ambiente é responsável pela dinâmica do trabalho. Ele é composto pelos dados temporais da ação, armazenando os históricos das operações para calcular o lucro obtido no tempo. Ele também contem variáveis globais de ambiente (descritos em mais detalhes a seguir) que serão utilizadas, por fim, o ambiente recebe a ação conjunta dos agentes, guarda a informação da data atual que está em processo e retorna o reforço para o agente.

A dinâmica simplificada do ambiente pode ser vista na figura a seguir:

Figura 26 – Dinâmica do Ambiente



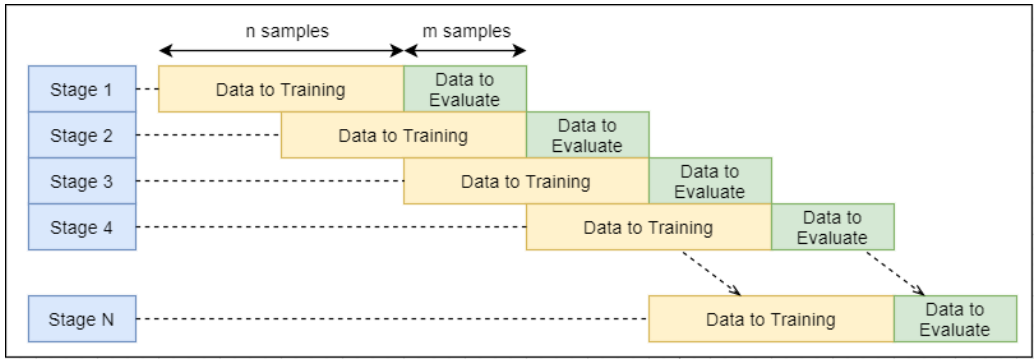
Fonte: Autor (2021)

As variáveis de ambiente que serão utilizados são: estágio de análise, capital inicial investido e o nome do ativo negociado.

O estágio de análise diz respeito ao tempo e os dados que serão abordados no treinamento e no teste do sistema.

A avaliação será feita utilizando *slide-window* similar ao ilustrado na figura a seguir, *n* diz respeito ao número de amostras de treinamento, e *m* ao número de amostras para calcular a eficiência do modelo.

Figura 27 – Estágio de Análise



Fonte: Autor (2021)

Neste projeto utilizou-se 7 estágios de análise, dois anos no período de treinamento e seis meses na etapa de validação do modelo.

A tabela 7 detalhada os estágios e os respectivos períodos de análise.

Tabela 7 – Tabela detalhando os estágios de análise

Stage	Treinamento		Validação	
	Data Inicial	Data Final	Data Inicial	Data Final
0	01/01/15	01/01/17	01/01/17	01/07/17
1	01/07/15	01/07/17	01/07/17	01/01/18
2	01/01/16	01/01/18	01/01/18	01/07/18
3	01/07/16	01/07/18	01/07/18	01/01/19
4	01/01/17	01/01/19	01/01/19	01/07/19
5	01/07/17	01/07/19	01/07/19	01/01/20
6	01/01/18	01/01/20	01/01/20	01/07/20

Fonte: Autor (2021)

4.6 Recompensa

O objetivo principal dos agentes é obter o maior lucro financeiro possível dentro do tempo de estudo. Para isto, a recompensa será exatamente o lucro obtido no tempo.

A primeira instância a recompensa será calculada segundo o lucro no tempo dos agentes, a equação 50 expressa a função recompensa.

$$L_{\text{total}} = C_{\text{investido}} - C_{\text{aportado}} + C_{\text{em caixa}} - T_{\text{transacionais}} \tag{50}$$

$$C_{\text{investido}} = \phi \sum_{i=1}^{n} \psi_{\text{i}} \tag{51}$$

$$T_{\text{transacionais}} = T_{\text{liquidação}} + T_{\text{emolumentos}} + T_{\text{empréstimo}} \tag{52}$$

Em que:

- L_{total} é o lucro total
- $C_{\text{investido}}$ é o capital investido
- C_{aportado} é o capital aportado
- $C_{\text{em caixa}}$ é o capital em caixa
- $T_{\text{transacionais}}$ são as taxas transacionais;
- $T_{\text{liquidação}}$ é a taxa de liquidação
- $T_{\text{emolumentos}}$ é a taxa de emolumentos
- $T_{\text{empréstimo}}$ é a taxa de emprestimo
- ϕ é o valor do ativo no dia atual;
- ψ_{i} é a quantidade comprada no dia i e não vendida;

4.7 Plano de Teste

O plano de teste é constituído de modo a validar o ganho financeiro obtido pelos agentes em um determinado tempo de análise, em função do estágio de análise listado na tabela 7.

Tabela 8 – Plano de testes

ID	Ambiente	Agentes	Algoritmo	Estado	Rede (Ator-Crítico)	Stock
1	IBOVESPA	2	MADDPG	Pontual	MLP	PETR3
2	IBOVESPA	2	MADDPG	Pontual	MLP	VALE3
3	IBOVESPA	2	MADDPG	Pontual	MLP	ITSA4
4	IBOVESPA	2	MADDPG	Deslocado (20)	MLP	PETR3
5	IBOVESPA	2	MADDPG	Deslocado (20)	MLP	VALE3
6	IBOVESPA	2	MADDPG	Deslocado (20)	MLP	ITSA4
7	IBOVESPA	1	DDPG	Pontual	MLP	PETR3
8	IBOVESPA	1	DDPG	Pontual	MLP	VALE3
9	IBOVESPA	1	DDPG	Pontual	MLP	ITSA4
10	IBOVESPA	1	DDPG	Deslocado (20)	MLP	PETR3
11	IBOVESPA	1	DDPG	Deslocado (20)	MLP	VALE3
12	IBOVESPA	1	DDPG	Deslocado (20)	MLP	ITSA4
13	IBOVESPA	1	DDPG	Deslocado (20)	Bi-LSTM + MLP	PETR3
14	IBOVESPA	1	DDPG	Deslocado (20)	Bi-LSTM + MLP	VALE3
15	IBOVESPA	1	DDPG	Deslocado (20)	Bi-LSTM + MLP	ITSA4

Fonte: Autor (2021)

As validações foram feitas entre papéis que compõem o índice IBOVESPA, tais como: PETR3, ITSA4 e VALE3, consistindo em comparar o ganho percentual financeiro obtido utilizando *buy-and-hold* com a técnica proposta por este trabalho, utilizando MARL.

Baseando-se nesse proposíto, foi implementado diferentes ambientes que compoem os cenários de testes. A tabela a seguir lista a proposta do plano de teste detalhando o ambiente em função das especificações propostas ao decorrer da metodologia.

5 IMPLEMENTAÇÃO DO PROJETO

O projeto foi implementado utilizando a IDE da Jet Brains (Pycharm Community), a versão do Python foi a 3.6. Escolheu-se o Pytorch como *framework* para implementação das redes neurais necessárias.

O Pytorch foi desenvolvido pelo Facebook's AI Research Lab e é uma biblioteca de código aberto utilizada para o desenvolvimento de aplicações que utilizem Redes Neurais. Ele prove computação baseada em tensores utilizando GPU e o desenvolvimento automático de Redes Neurais Profundas.

Analisando os *frameworks* de RL tem-se dois muito conhecidos: OpenAI e o RlLib. Esses *frameworks* auxiliam no desenvolvimento e no teste das aplicações que utilizam aprendizado por reforço, contém os algoritmos estudados neste trabalho já implementados e a possibilidade de testa-los em ambientes pré-configurados, porém não há uma documentação detalhada de quando há a necessidade de criar sistemas especiais e dedicados, ou seja, quer-se criar um ambiente totalmente diferente dos providos pelos *frameworks* ou modificar a arquitetura da rede utilizada na implementação do agente.

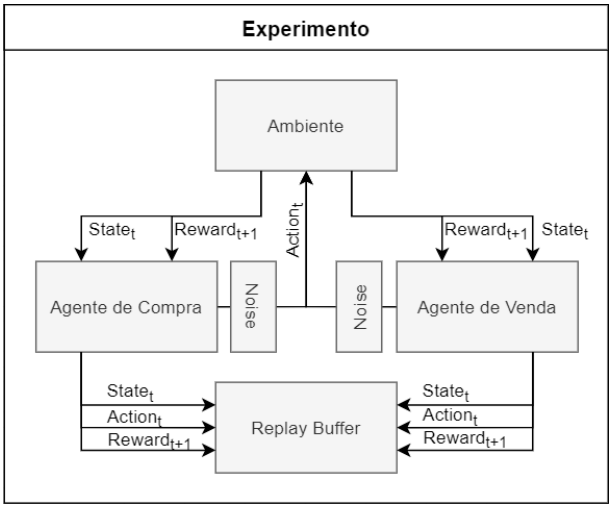
A escolha da utilização do Pytorch dar-se pela necessidade de querer estudar o impacto da construção das redes que compõem os agentes e do fato de ter um ambiente totalmente diferente do comumente aplicado aos problemas de Aprendizado por Reforço. Uma segunda possibilidade seria utilizar o TensorFlow no lugar do Pytorch, no que diz respeito a desempenho, há uma pequena variação que faz com que o TensorFlow tenha um desenpenho superior, porém o Pytorch é muito mais simples de utilizar. A escolha dele foi embasada nos motivos apresentados anteriormente, facilidade de utilização, necessidade de criar um ambiente diferente e redes neurais com arquiteturas distintas.

Os sistemas de DRL podem ser ser divididos, basicamente, em duas etapas: uma etapa experimental de interação com o ambiente; a segunda etapa de treinamento executada ciclicamente após uma série de interações experimentais.

As figuras 28 e 29 ilustram como é feito o processo de interação dos agentes com o ambiente.

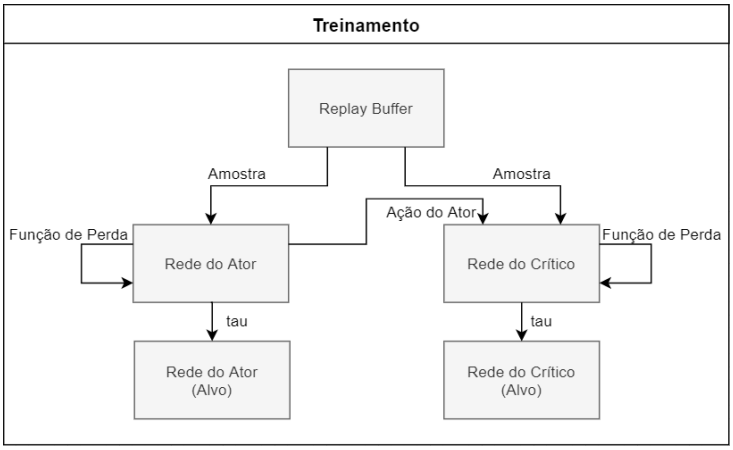
A primeira etapa (figura 28) os agentes executam suas ações no ambiente, a ação dada pelo agente passa por um gerador de ruido (a influencia do ruido no sinal final decai com o tempo de experiemento), todos os estados experimentados pelos agentes são armazenados no buffer de replay.

Figura 28 – DRL: Etapa de experimentação com o ambiente



Fonte: Autor (2021)

Figura 29 – DRL: Etapa de treinamento



Fonte: Autor (2021)

A segunda etapa consiste no treinamento das redes. Para isso, amostra-se uma fração dos dados do buffer de replay, atualiza-se os pesos das redes a partir da função de perda e, em função do tau, atualiza-se os pesos da rede alvo de forma mais suave.

Mais detalhes da teoria aplicada na implementação do MADDPG pode ser visto na seção 2.1.17. O algoritmo 10 refere-se a implementação do ciclo de experimentação e treinamento aplicados neste trabalho.

Após o treinamento do modelo entra-se na fase de validação. Dado o estágio que está sendo executado existe um período que é utilizado para comprovação do modelo (a descrição desse período em função do estágio está descrito na tabela 7).

Algoritmo 10 – Algoritmo aplicado para o treinamento dos agentes

```
Inicia environmentID, stockName, analysisStage, episodes, stepsPerUpdate,
  noiseScale, learningRate, τ
env = GetEnvironmentByID(environmentID)
para cada episode in episodes faça
    para cada step in LengthExperiment faça
        state = env.GetState()
        action = GetAgentsActions(state)
        nextObservation, reward = env.Result(action)
        ReplayBuffer.Add(state, action, reward, nextObservation)
        globalSteps = globalSteps + 1
        se globalSteps is multipleOf(stepsPerUpdate) então
            sample = ReplayBuffer.GetSample()
            loss = LossCalculation(sample)
            OptimizationWithAdam(loss)
            NetworkUpdate(τ)
        fim
    fim
fim
Retorna ModelEvaluation()
```

Enquanto o modelo está sendo treinado são salvos dados dos experimentos em disco, tais como: retorno médio por episódio, ação dos agentes, modelo Pytorch da rede treinada. No instante da validação dos dados são salvos os dados referente às ações dos agentes (agente de compra, agente de venda, ação conjunta dos agentes), o retorno por passo de análise, o preço de abertura, o número de papéis comprados pelo agente e o dinheiro total que o agente tem. Todos os dados salvos em disco foram feitos por meio do *TensorBoard*. Esta é uma ferramente criada pela mesma equipe do *TensorFlow* que possibilita a visualização do experimento em tempo de execução e deixa salvo todos os experimentos que foram executados. O impacto negativo da utilização do TensorBoard é que por salvar os dados em disco pode-se ter um gargalo dado pela unidade de disco do computador.

Os experimentos que foram executados são muito custosos computacionalmente, a tabela 9 apresenta o tempo de processamento médio para um experimento de um único estágio de análise, no caso foram utilizados 7 estágio de análise que compõem a janela deslizante do experimento, isso para apenas um ativo. Se fosse serializar todos os experimentos do trabalho teria-se cerca de 147 dias de processamento para um único ativo e 441 dias de processamento para todos os três ativos propostos na metodologia.

Uma forma de minimizar esse efeito é a utilização de Docker, com eles é possível criar instancias diferentes do projeto que podem ser processadas em paralelo, porém como foi men-

Tabela 9 – Análise do tempo de processamento em função do experimento

Experimento	Tempo de processamento médio por estágio de análise [hrs]	Tempo de processamento serializado com 7 estágios de análise [hrs]
ibovespa_1_1	19	133
ibovespa_1_20	23	161
ibovespa_2_1	26	182
ibovespa_2_20	60	420
ibovespa_lstm_1_20	377	2639

Fonte: Autor (2021)

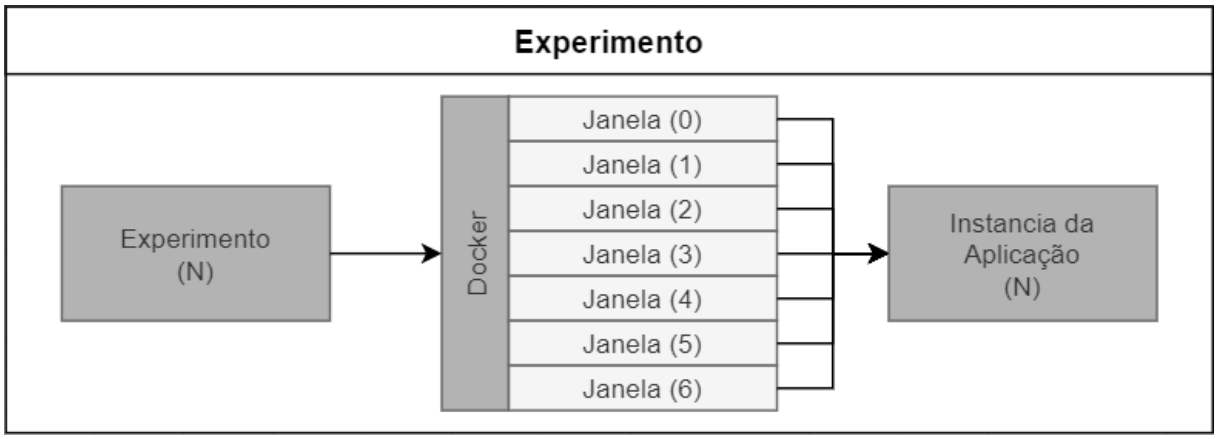
cionado anteriormente, o gargalo do processamento é o disco, mesmo com um servidor com alta capacidade de processamento, memória e GPU o processo fica limitado ao disco.

A solução foi criar grupos de máquinas com Docker que processam experimentos diferentes em paralelo. Foram fornecidas sete máquinas pela FEI (seis Windows e uma Linux) e foram utilizadas quatro máquinas virtuais em nuvem (todas Linux) para o processamento de todos os experimentos executados nesse trabalho.

O Docker contém uma imagem base da aplicação que será executada e containers que representam uma instancia da aplicação (cópia da imagem base) rodando um experimento pré configurado na etapa de inicialização do container. As variáveis configuradas na etapa de inicialização do container são: nome do ambiente, nome do ativo e estágio de análise.

A figura 30 exemplifica de forma genêrica processo.

Figura 30 – Docker: Conteinerização do projeto



Fonte: Autor (2021)

A instalação do Docker em um ambiente Linux pode ser feita com o seguinte comando:

"*sudo apt update sudo apt install apt-transport-https ca-certificates curl software-properties-common curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key*

add - sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable" sudo apt update apt-cache policy docker-ce sudo apt install docker-ce sudo systemctl status docker"

A criação da imagem do projeto pode ser feita utilizando o comando quando se está na raiz do projeto:

"*sudo docker build -t capati-model-training* ."

A criação de um novo container do projeto foi dada pelo seguinte comando em um ambiente Linux e Windowns, respectivamente:

"*docker run –name ibovespa_1_20_itsa4_2 -e ENV_ID=ibovespa_1_20 -e STOCK_NAME =ITSA4 -e ANALYSIS_STAGE=2 -v /home/felipe/project/stock-trading-with-rl/models:/code/models -d capati-model-training*"

"*docker run –name ibovespa_1_20_itsa4_0 -e ENV_ID=ibovespa_1_20 -e STOCK_NAME =ITSA4 -e ANALYSIS_STAGE=0 -v C:/Users/felipecapati/project/stock-trading-with-rl-without-lstm/models:/code/models -d capati-model-training*"

O Docker possibilita a criação de volumes que são pontes criadas entre container e servidor hospedeiro, esses volumes facilitam a movimentação dos dados gerados pelos containers e, neste caso, armazenam as informações referentes aos experimentos executados. A parametrização dos volumes é dada pelo parâmetro *-v* passada no momento que é iniciado o container.

6 RESULTADOS EXPERIMENTAIS

O objetivo deste capítulo é apresentar os resultados obtidos experimentalmente nos testes efetuados seguindo a tabela 8. Avançaremos em seções distintas para abordar os experimentos, os títulos são referentes a cada experimento explicitando o ambiente utilizado, número de agentes, deslocamento temporal. As análises foram executadas frente ao *buy-and-hold*.

6.1 Ibovespa 1 agente - tempo 1

Neste experimento aborda-se o modelo aplicando o DDPG utilizando o MLP para a constituição da rede neural aplicada no ator-crítico. O estado é definido no instante *t*, sem deslocamento temporal. Os ativos aplicados foram PETR3, ITSA4 e VALE3. Lembrando que o treinamento é individual, ou seja, os agentes são treinados separadamente nos ativos aplicados.

Os resultados apresentados na tabela 10 são referentes aos experimentos executados no ambiente *ibovespa_1_1* e todos os seus estágios de análise referentes à janela deslizante, sendo que P_{i} e $P\mathbf{f}$ significam, respectivamente, preço inicial do ativo e preço final do ativo.

A tabela 11 diz respeito ao consolidado feito das análises apresentadas na tabela 10. Partindo do princípio que a aplicação passa pelos 7 estágios, o consolidado final é dado pela equação 53, em que $\mathrm{e_i}$ representa o lucro percentual do estágio de análise na posição *i*.

$$\psi = \left[\prod_{i=0}^{6} (1 + e_{\mathrm{i}}) - 1\right] * 100 \tag{53}$$

Os resultados apresentados na tabela 11 são positivos para o modelo proposto, superando o *buy-and-hold* em mais de 34% quando analisado o papel ITSA4 e 19.63% quando analisado o papel PETR3. As operações em VALE3 foram inferiores ao *buy-and-hold*, chegando em -16.83%.

Seguindo a revisão bibliográfica abordada anteriormente, este modelo é o mais similar aos apresentados na maioria dos projetos revisados, tendo um resultado positivo mesmo com quedas no valor do ativo.

6.2 Ibovespa 1 agente - tempo 20

Neste experimento aborda-se o modelo aplicando o DDPG utilizando o MLP para a constituição da rede neural aplicada no ator-crítico. O estado é definido deslocando 20 posições

Tabela 10 – Ibovespa_1_1: Análise dos experimentos por estágio de análise

Ativo	Episódios	Stage	Reward				Pi	Pf	Buy-And-Hold	Lucro Final
			Min	Max	Mean	Std				
PETR3	50000	0	0	0	0	0	16.83	13.05	-22.460	0
PETR3	50000	1	0	0	0	0	13.58	16.76	23.417	0
PETR3	50000	2	-6.34	1.44	-0.324	1.269	21.68	19.14	-11.716	1.44
PETR3	50000	3	0	0	0	0	21.27	24.56	15.468	0
PETR3	50000	4	0	0	0	0	30.86	30.11	-2.430	0
PETR3	50000	5	0	0	0	0	27.01	32.81	21.474	0
PETR3	50000	6	0	0	0	0	31.33	22.17	-29.237	0
ITSA4	50000	0	-14.9	2.307	-6.215	4.871	8.959	8.055	-10.090	-11.21
ITSA4	50000	1	0	0	0	0	8.973	9.845	9.718	0
ITSA4	50000	2	0	0	0	0	12.73	8.91	-30.008	0
ITSA4	50000	3	-9.71	20.21	5.1723	8.691	10.02	11.6	15.768	14.17
ITSA4	50000	4	0	0	0	0	13.19	13.06	-0.986	0
ITSA4	50000	5	0	0	0	0	12.57	14.2	12.967	0
ITSA4	50000	6	0	0	0	0	13.01	9.8	-24.673	0
VALE3	50000	0	45.5	53.18	49.939	1.978	45.5	52.07	14.440	14.440
VALE3	50000	1	0	0	0	0	44.45	53.99	21.462	0
VALE3	50000	2	49.53	62.2	54.91	3.201	49.53	50.1	1.151	1.151
VALE3	50000	3	40.61	56.46	47.416	4.226	45.51	47.85	5.142	5.142
VALE3	50000	4	30.94	40.01	34.172	2.158	31.89	40.01	25.463	25.463
VALE3	50000	5	0	0	0	0	33.61	29.12	-13.359	0
VALE3	50000	6	0	0	0	0	52.55	55.44	5.500	0

Fonte: Autor (2021)

Tabela 11 – Resultados do experimento Ibovespa 1 1

Papel	***Buy-And-Hold [%]***	***Modelo(DDPG + MLP) [%]***
PETR3	-18.183	1.44
ITSA4	-32.652	1.375
VALE3	69.531	52.7

Fonte: Autor (2021)

anteriores ao estado atual. Os ativos aplicados foram PETR3, ITSA4 e VALE3. Lembrando que o treinamento é individual, ou seja, os agentes são treinados separadamente nos ativos aplicados.

Os resultados apresentados na tabela 12 são referentes aos experimentos executados no ambiente *ibovespa_1_20* e todos os seus estágios de análise referentes à janela deslizante.

A tabela 13 diz respeito ao consolidado feito das análises apresentadas na tabela 12. Partindo do princípio que a aplicação passa pelos 7 estágios, o consolidado final é dado pela equação 53, em que e_i representa o lucro percentual do estágio de análise na posição *i*.

Tabela 12 – Ibovespa_1_20: Análise dos experimentos por estágio de análise

Ativo	Episódios	Stage	Reward				Pi	Pf	Buy-And-Hold	Lucro Final
			Min	Max	Mean	Std				
PETR3	50000	0	-23.3	2.761	-10.94	5.36	16.83	13.83	-17.825	-19.46
PETR3	50000	1	0	0	0	0	13.58	15.9	17.084	0
PETR3	50000	2	-2.4	46.97	13.282	10.84	21.68	21.49	-0.876	5.026
PETR3	50000	3	0	0	0	0	21.27	27.06	27.221	0
PETR3	50000	4	0	0	0	0	30.86	28.95	-6.189	0
PETR3	50000	5	0	0	0	0	27.01	31.42	16.327	0
PETR3	50000	6	0	0	0	0	31.33	20.67	-34.025	0
ITSA4	50000	0	0	0	0	0	8.959	8.173	-8.773	0
ITSA4	50000	1	0	0	0	0	8.973	9.509	5.973	0
ITSA4	50000	2	0	0	0	0	12.73	10.19	-19.953	0
ITSA4	50000	3	0	0	0	0	10.02	11.66	16.367	0
ITSA4	50000	4	0	0	0	0	13.19	12.14	-7.961	0
ITSA4	50000	5	0	0	0	0	12.57	13.11	4.296	0
ITSA4	50000	6	0	0	0	0	13.01	8.84	-32.052	0
VALE3	50000	0	0	0	0	0	33.61	27.15	-19.220	0
VALE3	50000	1	-18.3	0	-8.352	6.029	31.89	35.7	11.947	-11.94
VALE3	50000	2	-10.2	22.29	2.9286	9.264	45.51	50.21	10.327	12.5
VALE3	50000	3	-2.78	21.61	9.3145	5.642	49.53	49.69	0.323	-1.2
VALE3	50000	4	0	0	0	0	45.5	49.71	9.253	0
VALE3	50000	5	0	0	0	0	44.45	50.93	14.578	0
VALE3	50000	6	0	0	0	0	52.55	52.82	0.514	0

Fonte: Autor (2021)

Tabela 13 – Resultados do experimento Ibovespa 1 20

Papel	***Buy-And-Hold [%]***	***Modelo(DDPG + MLP) [%]***
PETR3	32.406	-15.41
ITSA4	-17.116	0
VALE3	25.939	-2.121

Fonte: Autor (2021)

Os resultados apresentados na tabela 13 não são positivos para o modelo proposto, superando o *buy-and-hold* apenas para o ITSA4.

Acredita-se que a modelagem deslocando estados anteriores ao atual não auxia na solução e, além disso, deixa o estado mais complexo de ser interpretados, dificultando o reconhecimento dos padrões necessários para identificar a melhor transação a ser feita.

6.3 Ibovespa 2 agente - tempo 1

Neste experimento aborda-se o modelo aplicando o MADDPG utilizando o MLP para a constituição da rede neural aplicada no ator-crítico. O estado é definido no instante *t*, sem deslocamento temporal. Os ativos aplicados foram PETR3, ITSA4 e VALE3. Lembrando que o treinamento é individual, ou seja, os agentes são treinados separadamente nos ativos aplicados.

Os resultados apresentados na tabela 14 são referentes aos experimentos executados no ambiente *ibovespa_2_1* e todos os seus estágios de análise referentes à janela deslizante.

Tabela 14 – Ibovespa_2_1: Análise dos experimentos por estágio de análise

Ativo	Episódios	Stage	Reward				Pi	Pf	Buy-And-Hold	Lucro Final
			Min	Max	Mean	Std				
PETR3	100000	0	0	0	0	0	16.83	13.05	-22.460	0
PETR3	100000	1	-31.4	0.031	-17.8	6.79	13.58	16.76	23.417	-23.38
PETR3	100000	2	0	0	0	0	21.68	19.14	-11.716	0
PETR3	100000	3	-1.8	51.56	27.273	16.3	21.27	24.56	15.468	29.04
PETR3	100000	4	-188	12.34	-19.39	53.99	30.86	30.11	-2.430	-186.2
PETR3	100000	5	-200	0.125	-76.26	97.86	27.01	32.81	21.474	-200.05
PETR3	100000	6	0	0	0	0	31.33	22.17	-29.237	0
ITSA4	100000	0	0	0	0	0	8.959	8.055	-10.090	0
ITSA4	100000	1	-16	0.031	-9.402	3.844	8.973	9.845	9.718	-9.961
ITSA4	100000	2	0	0	0	0	12.73	8.91	-30.008	0
ITSA4	100000	3	-5.13	27.81	10.975	10.17	10.02	11.6	15.768	21.39
ITSA4	100000	4	0	0	0	0	13.19	13.06	-0.986	0
ITSA4	100000	5	-5.82	12.58	4.6069	4.878	12.57	14.2	12.967	12.02
ITSA4	100000	6	-1.41	39.01	27.623	9.327	13.01	9.8	-24.673	26.93
VALE3	100000	0	-199	0.031	-98.42	89.24	33.61	29.12	-13.359	-180
VALE3	100000	1	0	0	0	0	31.89	40.01	25.463	0
VALE3	100000	2	-7.92	4.919	-3.579	2.616	45.51	47.85	5.142	-7.92
VALE3	100000	3	0	0	0	0	49.53	50.1	1.151	0
VALE3	100000	4	0	0	0	0	45.5	52.07	14.440	0
VALE3	100000	5	0	0	0	0	44.45	53.99	21.462	0
VALE3	100000	6	-32.7	6.474	-10.58	11.13	52.55	55.44	5.500	5.891

Fonte: Autor (2021)

A tabela 15 diz respeito ao consolidado feito das análises apresentadas na tabela 14. Partindo do princípio que a aplicação passa pelos 7 estágios, o consolidado final é dado pela equação 53, em que e_i representa o lucro percentual do estágio de análise na posição *i*.

Os resultados apresentados na tabela 15 são positivos para o modelo proposto, superando o *buy-and-hold* em mais de 88% quando analisado o papel ITSA4 e 3.4% quando comparado com o PETR3. No experimento do VALE3 houve sérios problemas de tomada de decisão que

Tabela 15 – Resultados do experimento Ibovespa 2 1

Papel	***Buy-And-Hold [%]***	***Modelo(MADDPG + MLP) [%]***
PETR3	-18.131	-14.731
ITSA4	-32.652	55.408
VALE3	69.531	-178

Fonte: Autor (2021)

fizeram com o que o modelo perdesse em 178% o capital inicial investido. Acredita-se que seja um ponto fora da curva na tomada de decisão e necessitaria de mais experimentos para ter uma posição assertiva sobre a viabilidade do modelo.

6.4 Ibovespa 2 agente - tempo 20

Neste experimento aborda-se o modelo aplicando o MADDPG utilizando o MLP para a constituição da rede neural aplicada no ator-crítico. O estado é definido utilizando um deslocamento de 20 posições para o estado atual. Os ativos aplicados foram PETR3, ITSA4 e VALE3. Lembrando que o treinamento é individual, ou seja, os agentes são treinados separadamente nos ativos aplicados.

Os resultados apresentados na tabela 16 são referentes aos experimentos executados no ambiente *ibovespa_2_20* e todos os seus estágios de análise referentes à janela deslizante.

A tabela 17 diz respeito ao consolidado feito das análises apresentadas na tabela 16. Partindo do princípio que a aplicação passa pelos 7 estágios, o consolidado final é dado pela equação 53, em que e_i representa o lucro percentual do estágio de análise na posição *i*.

Os resultados apresentados na tabela 15 são positivos para o modelo proposto, superando o *buy-and-hold* em mais de 37% quando operando em tendencia de baixa, como apresentado em ITSA4 e PETR3. No experimento do VALE3 trabalhou-se com uma tendência de alta e o modelo não foi capaz de operar de forma a superar o *buy-and-hold*, porém manteve-se positivo.

6.5 Ibovespa LSTM 1 agente - tempo 20

Neste experimento aborda-se o modelo aplicando o DDPG utilizando uma camada Bi-LSTM para a constituição da rede neural aplicada no ator-crítico. Este teste busca estudar a influencia da modelagem da rede do ator-crítico que contribuirá positivamente ou negativamente na tomada de decisão. O estado é definido utilizando um deslocamento de 20 posições

Tabela 16 – Ibovespa_2_20: Análise dos experimentos por estágio de análise

Ativo	Episódios	Stage	Reward				Pi	Pf	Buy-And-Hold	Lucro Final
			Min	Max	Mean	Std				
PETR3	100000	0	0	0	0	0	16.83	13.83	-17.825	0
PETR3	100000	1	0	0	0	0	13.58	15.9	17.084	0
PETR3	100000	2	0	0	0	0	21.68	21.49	-0.876	0
PETR3	100000	3	0	0	0	0	21.27	27.06	27.221	0
PETR3	100000	4	0	0	0	0	30.86	28.95	-6.189	0
PETR3	100000	5	0	0	0	0	27.01	31.42	16.327	0
PETR3	100000	6	0	0	0	0	31.33	20.67	-34.025	0
ITSA4	100000	0	0	0	0	0	8.959	8.173	-8.773	0
ITSA4	100000	1	0	0	0	0	8.973	9.509	5.973	0
ITSA4	100000	2	0	0	0	0	12.73	10.19	-19.953	0
ITSA4	100000	3	0	0	0	0	10.02	11.66	16.367	0
ITSA4	100000	4	0	0	0	0	13.19	12.14	-7.961	0
ITSA4	100000	5	-12.2	5.84	-3.397	4.803	12.57	13.11	4.296	-4.108
ITSA4	100000	6	0	0	0	0	13.01	8.84	-32.052	0
VALE3	100000	0	-2.62	30.54	18.95	7.323	33.61	27.15	-19.220	25.26
VALE3	100000	1	0	0	0	0	31.89	35.7	11.947	0
VALE3	100000	2	0	0	0	0	45.51	50.21	10.327	0
VALE3	100000	3	0	0	0	0	49.53	49.69	0.323	0
VALE3	100000	4	0	0	0	0	45.5	49.71	9.253	0
VALE3	100000	5	-14.8	2.62	-6.899	4.177	44.45	50.93	14.578	-13.99
VALE3	100000	6	-1.47	32.92	15.002	8.794	52.55	52.82	0.514	-1.471

Fonte: Autor (2021)

Tabela 17 – Resultados do experimento Ibovespa 2 20

Papel	***Buy-And-Hold [%]***	***Modelo(MADDPG + MLP) [%]***
PETR3	-12.645	0
ITSA4	-41.263	-4.108
VALE3	25.939	6.151

Fonte: Autor (2021)

para o estado atual, devido a utilização da rede Bi-LSTM. Os ativos aplicados foram PETR3, ITSA4 e VALE3. Lembrando que o treinamento é individual, ou seja, os agentes são treinados separadamente nos ativos aplicados.

Os resultados apresentados na tabela 18 são referentes aos experimentos executados no ambiente *ibovespa_lstm_1_20* e todos os seus estágios de análise referentes à janela deslizante.

A tabela 19 diz respeito ao consolidado feito das análises apresentadas na tabela 18. Partindo do princípio que a aplicação passa pelos 7 estágios (O stage 6 foi desconsiderado para o cálculo do consolidado do PETR3 devido ao fato de ser um ponto fora da curva com valor

Tabela 18 – Ibovespa_lstm_1_20: Análise dos experimentos por estágio de análise

Ativo	Episódios	Stage	Reward				Pi	Pf	Buy-And-Hold	Lucro Final
			Min	Max	Mean	Std				
PETR3	100000	0	-4	7.41	0.13	2.846	13.65	13.11	-3.956	-2.906
PETR3	100000	1	-0.02	7.457	3.193	1.702	15.48	16.91	9.238	4.349
PETR3	100000	2	-12.8	17.95	-1.297	7.48	22.73	19	-16.410	-7.675
PETR3	100000	3	-0.02	18.91	10.519	4.746	21.89	24.78	13.202	6.953
PETR3	100000	4	-6.58	2.049	-2.33	1.745	32.7	30.44	-6.911	-1.702
PETR3	100000	5	-2.63	5.109	2.636	2.009	29.55	32.38	9.577	4.172
PETR3	100000	6	-3.25	34.95	23	10.61	11	22.5	104.545	4.172
ITSA4	100000	0	-5.49	3.006	-2.03	2.128	8.86	8.18	-7.675	-3.374
ITSA4	100000	1	-2.84	2.198	-0.295	1.351	9.92	9.83	-0.907	-0.32
ITSA4	100000	2	-15.6	2.109	-6.409	5.838	12.14	9.15	-24.629	-12.71
ITSA4	100000	3	-0.41	14.88	9.3722	3.836	9.41	11.9	26.461	12.02
ITSA4	100000	4	-4.22	4.073	0.3309	2.463	12.5	13	4.000	3.834
ITSA4	100000	5	-3.4	4.11	1.654	1.96	13	14.13	8.692	3.852
ITSA4	100000	6	-5.72	8.331	1.4154	3.652	8.8	9.67	9.886	4.114
VALE3	100000	0	-9.57	0.356	-4.523	2.194	31.15	28.95	-7.063	-2.491
VALE3	100000	1	-4.55	9.189	2.2206	4.701	35.05	40.26	14.864	9.189
VALE3	100000	2	-1.37	16.64	8.2288	4.21	41.1	49.7	20.925	9.604
VALE3	100000	3	-4.98	6.08	-0.793	3.206	54.1	30.31	-43.974	-4.239
VALE3	100000	4	-4.01	2.773	0.627	1.904	50.53	53.1	5.086	2.774
VALE3	100000	5	-4.72	5.013	0.3166	2.845	48.62	53.65	10.346	3.814
VALE3	100000	6	-3.12	22.68	15.115	7.312	37.5	56.08	49.547	22.44

Fonte: Autor (2021)

muito alto em relação a todos os pontos de *buy-and-hold*), o consolidado final é dado pela equação 53, em que e_i representa o lucro percentual do estágio de análise na posição *i*.

Tabela 19 – Resultados do experimento Ibovespa lstm 1 20

Papel	***Buy-And-Hold [%]***	***Modelo(DDPG + Bi-LSTM) [%]***
PETR3	1.267	2.445
ITSA4	8.317	5.737
VALE3	25.417	45.983

Fonte: Autor (2021)

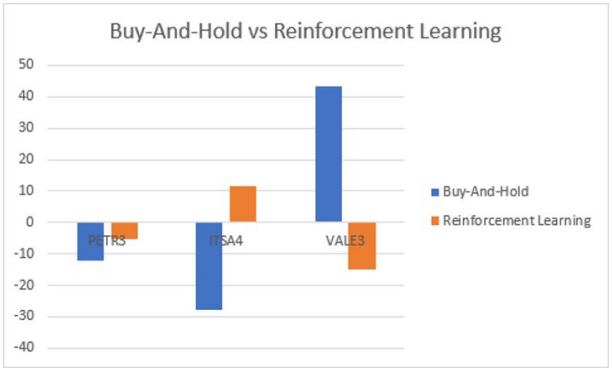
Os resultados apresentados na tabela 19 são positivos para o modelo proposto, indicando a viabilidade da modelagem do ator-crítico utilizando a Bi-LSTM. Comparando os resultados apresentados do “ibovespa 1 20”, com o “ibovespa lstm 1 20”, em que tem-se modelos identidos, diferenciando-se em na modelagem do ator-crítico, teve uma alta relevante e uma consistencia maior nas operações, ou seja, em quedas o agente perde pouco e altas o agente ganha pouco. a longo prazo ele é mais consistente superando o *buy-and-hold*.

7 ANÁLISE DOS RESULTADOS

No capítulo 6 foi apresentado os resultados dos experimentos e algumas análises. A pergunta principal a ser respondida é: O modelo proposto utilizando Reinforcement Learning (RL) foi superior ao buy-and-hold?

O gráfico da figura 31 é uma síntese de todas as abordagens de RL propostas por este trabalho, ele agrupa todos os experimentos executados. Tem-se que o Reinforcement Learning foi superior ao buy-and-hold nos experimentos feitos na PETR3 e ITSA4, porém na VALE3 o buy-and-hold foi superior. Acredita-se que isto está relacionado a tentativa do modelo de executar um *short* mal sucedido (tal medida será analisada em mais detalhes neste capítulo) e a capacidade do modelo de aproveitar dos movimentos de alta das séries temporais.

Figura 31 – Análise dos resultados (buy-and-hold vs reinforcement learning)



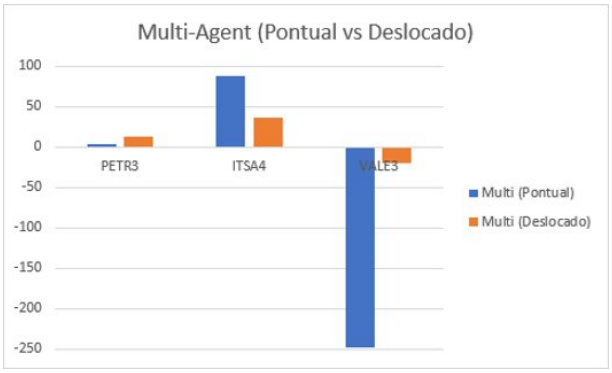
Fonte: Autor (2021)

Fazendo um estudo mais minucioso dos experimentos têm-se que a abordagem *multi-agent* deslocada foi superior à abordagem *multi-agent* pontual (Figura 32). Quando observam-se os resultados da abordagem *single-agent*, a análise pontual foi superior a deslocada (Figura 33). Acredita-se que este motivo está relacionado a complexidade do estado quando se utiliza deslocamento temporal, com isso há uma dificuldade de interpretação dos dados por um único agente e o modelo *multi-agent* é capaz de abstrair melhor esses estados devido à segmentação de responsabilidade proveniente da utilização de dois agentes.

Outra questão a ser respondida no trabalho é: O modelo *multi-agent* é superior ao *single-agent*?

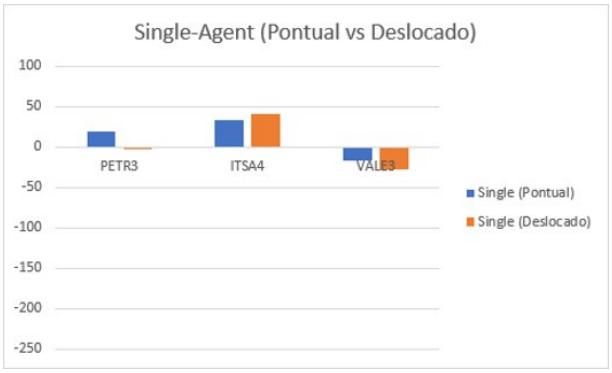
Analisando o gráfico da figura 34 tem-se que para PETR3 os modelos desempenharam de forma similar, porém o *single-agent* foi superior. Nos experimentos feitos em ITSA4 o *multi-agent* foi superior e na VALE3 o *single-agent* foi superior. De forma geral os modelos

Figura 32 – Análise dos resultados (pontual vs deslocado *multi-agent*)



Fonte: Autor (2021)

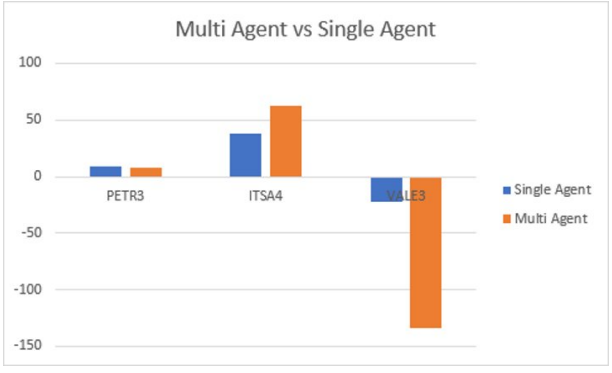
Figura 33 – Análise dos resultados (pontual vs deslocado *single-agent*)



Fonte: Autor (2021)

são muito similares e acabam por ter mais aderência em uma situação ou outra, ambos são capazes de serem utilizados para criar sistemas de compra e venda de ações e, existem questões que vão além de serem *multi-agent* ou *single-agent* para serem realmente eficientes em suas tomadas de decisão.

Figura 34 – Análise dos resultados (*single-agent* vs *multi-agent*)

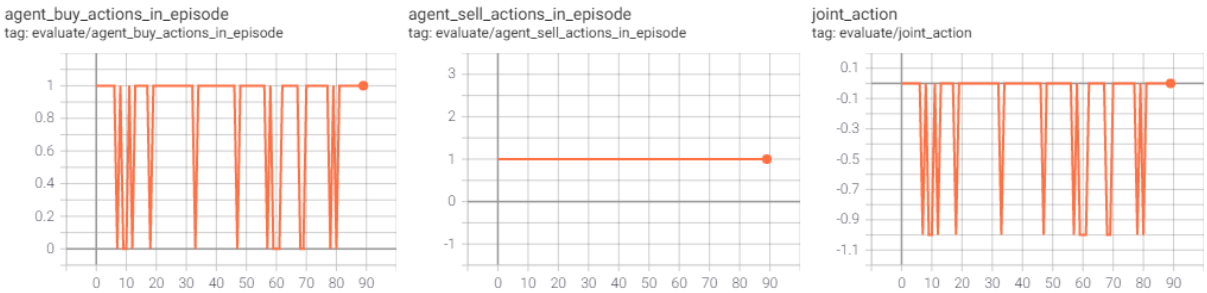


Fonte: Autor (2021)

O modelo proposto, tanto *single-agent* ou *multi-agent*, possibilitam o agente operar em *long* ou *short* (tendencias de alta ou baixa, respectivamente), porém quando analisa-se as operações executadas pelos agentes, tem-se uma predominância maior a executar *long* do que *short*, mesmo em situações que o preço está em queda. Acredita-se que isso está relacionado à capacidade do agente de explorar corretamente o ambiente, ou seja, ele não opera em *short* porque não tentou essa possibilidade, ou a médio prazo ele poderá ser recompensado, porém ele não aprende isso corretamente porque nos primeiros estados em que tenta uma operação *short* ele acaba sendo reforçado negativamente. Analisando as tendências de baixa, tem-se que muitas vezes o agente prefere não operar do que executar um *short*, são casos raros que o agente opera em tendências de baixa para obter lucro. Quando analisado o "*ponto fora da curva*" apresentado nos experimentos da tabela 15 e dos gráficos das figuras 31, 32 e 34, o valor de -178% ocorreu devido à tentativa, não bem sucedida, do agente executar o *short*.

Nos experimentos analisados não houve grandes cooperações entre os agentes. Existem cenários na qual apenas um agente apropriou-se da tomada de decisão (único valor que varia em função do tempo), enquanto o outro agente tem uma posição fixa durante todo o experimento. A figura 35 e 36 apresenta este cenário, em que o primeiro gráfico a esquerda refere-se a ação do agente de compra, o segundo refere-se ao agente de venda e o terceiro diz respeito a ação conjunta entre os dois agentes.

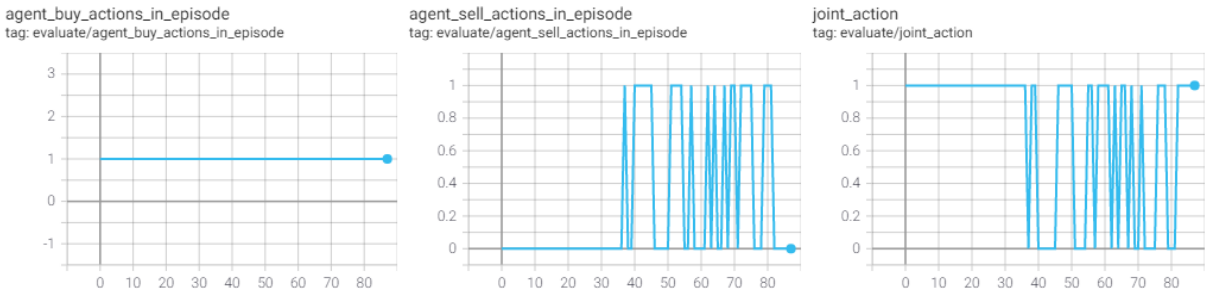
Figura 35 – Ações dos agentes no tempo (Experimento Ibovespa 2 1 - VALE3 - Stage 0)



Fonte: Autor (2021)

O motivo da implementação de um modelo capaz de executar ações contínua foi discutido no capítulo referente a metodologia do projeto, há um ganho na utilização, entretanto houve problemas vistos no desenvolvimento deste projeto. O fato de existirem infinitas possibilidades que o agente possa executar dentro de um range de -1 a 1 ou de 0 a 1 (varia em função da modelagem) traz uma dificuldade na convergência do algoritmo nos episódios propostos (50.000 episódios). Existem períodos de treinamento que o agente não foi capaz de aprender a dinâmica

Figura 36 – Ações dos agentes no tempo (Experimento Ibovespa 2 1 - PETR3 - Stage 3)



Fonte: Autor (2021)

do ambiente e fez com que comprometesse a sua tomada de decisão ou, em alguns casos, ele não executou ação alguma porque sempre que executava uma ação ela não era suficiente para ter-se um reforço positivo.

A tomada de decisão no modelo discreto (compra, venda e espera) possibilita mais facilidade da convergência do modelo em que na revisão bibliográfica a maior parte dos trabalhos aplicavam modelos com ação discretos para os agentes.

Os testes realizados utilizando a rede Bi-LSTM trouxe mais consistência a transação efetuada pelo agente, as oscilações nas transações são menores e mais estável sendo superior ao *buy-and-hold* na maior parte das operações a longo prazo, isto prova e abre uma possibilidade de estudos modelando o problema utilizando POMDP, a inércia proporcionada pela rede Bi-LSTM tem impacto positivo na decisão dos agentes, deixa o sistema equilibrado e, frequentemente, o agente executa ações.

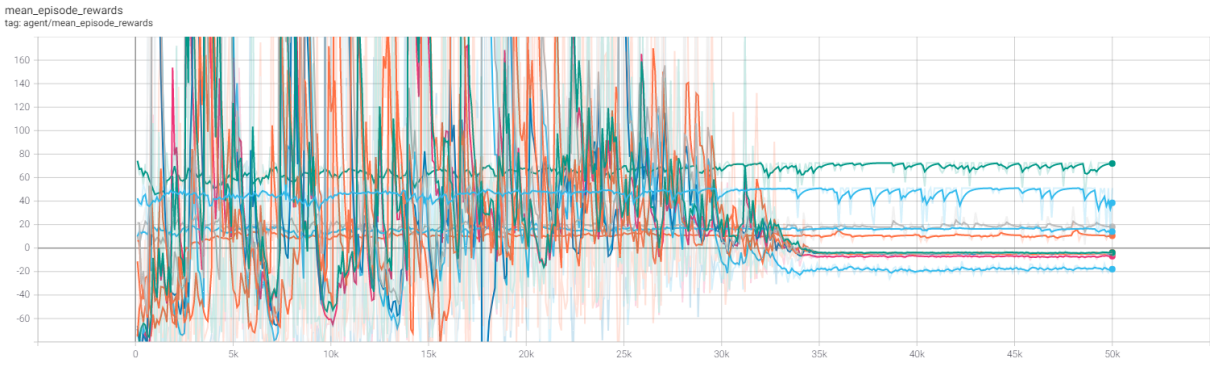
Quando avalia-se as tabelas referentes a todas as transações executadas em todos os estágios de análise (tabelas 10, 12, 14 e 16) pode-se visualizar que há períodos em que os agentes optaram por não executar ação alguma. Há algumas hipóteses para esta decisão e serão abordadas a seguir.

A primeira possibilidade está relacionada ao período de transação, se é um período de queda os agentes tendem a não realizar operações de venda a descoberto devido as experiências passadas ou até mesmo a não exploração dessa categoria de ação.

A segunda possibilidade está diretamente relacionada com a capacidade do modelo de abstrair a dinâmica do sistema, quando observam-se os estágios de análise que obtiveram ações neutras, ou seja, zero em todos os estágios, vê-se algo em comum entre eles no gráfico de "recompensa média por episódios".

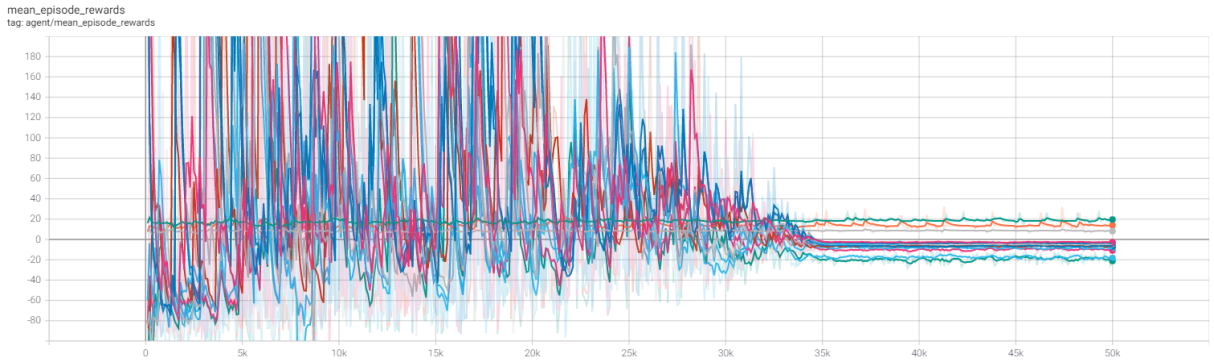
Os gráficos das figuras 37, 38, 39 e 40 apresentam, respectivamente, a “recompensa média por episódios” dos experimentos: *ibovespa_1_1*, *ibovespa_1_20*, *ibovespa_2_1* e *ibovespa_2_20*. As legendas foram retiradas para não poluir os gráficos devido a quantidade de experimentos que foram adicionados no mesmo gráfico e a observação que será feita indifere se o experimento diz respeito ao estágio de análise *x* do ativo *y*.

Figura 37 – Recompensa Média por Eposódio. Experimento *ibovespa_1_1*



Fonte: Autor (2021)

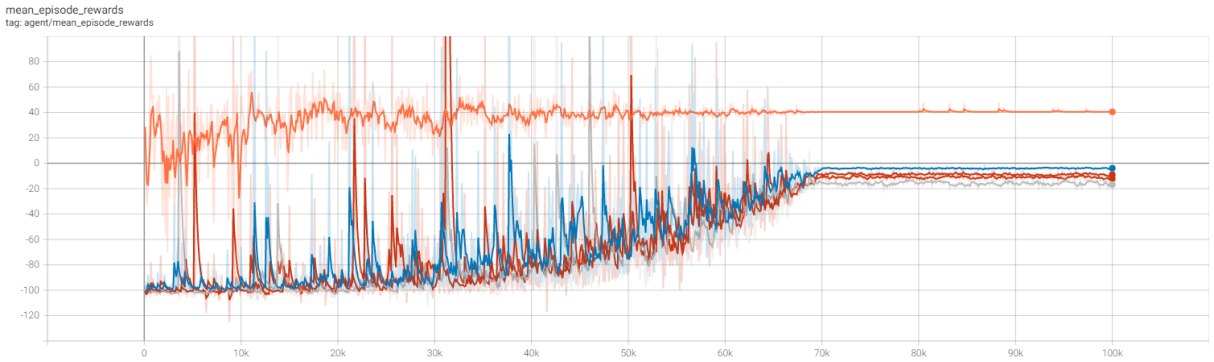
Figura 38 – Recompensa Média por Eposódio. Experimento *ibovespa_1_20*



Fonte: Autor (2021)

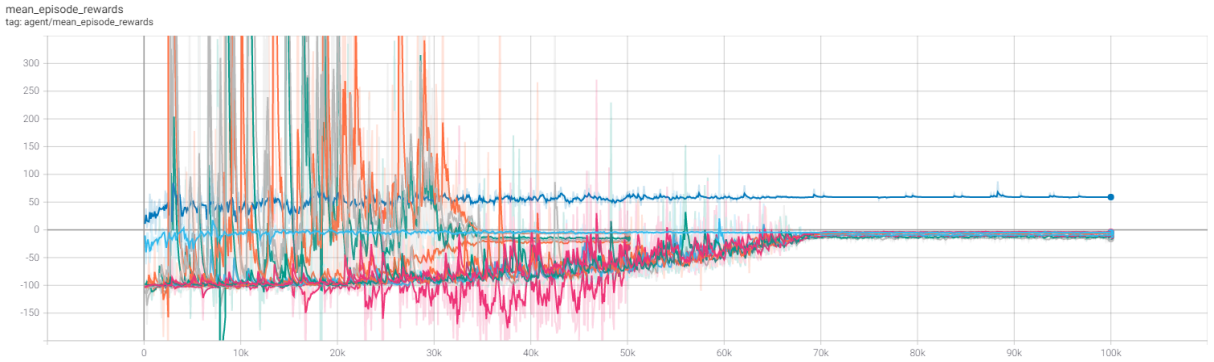
Os gráficos das figuras 37, 38, 39 e 40 contêm multiplas curvas, cada curva representa um experimento feito em um estágio de análise *x* para o ativo *y* que não executou ação alguma na base de validação. As curvas dividem-se em dois grupos, com recompensa média superiores a zero e inferiores a zero, para as curvas que tiveram convergencia superiores a zero significa que os agentes foram capazes de descobrir uma dinâmica com os dados de treinamento capaz de ter um lucro médio no tempo e, o contrário também é valido, para pontos em que os dados foram inferiores a zero os agentes não descobriram uma forma de obter lucro dentro da dinâmica da base de treinamento. Os agentes podem executar ações de *long* ou *short*, ou seja, em teoria existe *sempre* uma solução que é possível de se obter lucro no período de análise.

Figura 39 – Recompensa Média por Eposódio. Experimento *ibovespa_2_1*



Fonte: Autor (2021)

Figura 40 – Recompensa Média por Eposódio. Experimento *ibovespa_2_20*



Fonte: Autor (2021)

Observa-se que há um distúrbio maior na recompensa para os experimentos inferiores a 70% do total de episódios, dar-se isso devido ao agente executar mais *exploration* do que *exploitation* e, para os 30% finais o contrário acontece.

Utilizando os gráficos como fundamento, acredita-se em dois possíveis cenários: o primeiro, quando as curvas convergem para valores negativos não foi possível aprender uma dinâmica recompensadora na base de treinamento e quando valida-se o modelo os agentes optam por não executar ação alguma do que obter reforços negativos; a segunda, quando as linhas convergem para valores positivos o agente aprendeu a dinâmica do sistema utilizando a base de treinamento, porém está dinâmica é muito diferente da base de validação, ou seja, há estados totalmente diferentes entre treinamento e validação e o agente opta por não executar ação alguma já que não conhece aquele estado.

Uma forma de ajustar isso é obrigar o agente a sempre executar uma ação, reforçando ele negativamente para cada episódio que ele passa sem executar ação alguma, porque para alguns experimentos a solução convergiu para o agente “aprender” que executar nada é melhor

que executar algo errado, porém não aprendeu como executar de maneira correta a ação em um determinado estado (sempre existirá solução independente do estado em que ele esteja).

No que diz respeito ao segundo ponto abordado, uma forma de ajustar seria diminuir o período de análise apresentado na tabela 7. O período de treinamento e validação poderiam ser menores para que a dinâmica estudada no treinamento seja o mais próximo possível da etapa de validação. Porém sabe-se que há fatores adicionais que impactaram muito no preço dos ativos estudados, tais como: notícias, troca de gestão, efeitos climáticos, acidentes, entre outros.

8 CONCLUSÃO E TRABALHOS FUTUROS

O objetivo do trabalho é o desenvolvimento de um algoritmo de IA utilizando aprendizado por reforço multiagente aplicado à negociação de ativos na bolsa de valores, tal objetivo foi atingido quando analisa-se a metodologia proposta e os resultados. Além do objetivo principal do projeto, foram propostos alguns experimentos adicionaios para estudar o comportamento da rede do agente e do crítico composto por uma Bi-LSTM. Os resultados finais foram superiores ao *buy-and-hold* na maioria dos experimentos.

Fazendo uma análise geral do projeto, tanto o modelo *single-agent* quanto o *multi-agent* podem ser lucrativos. Os experimentos mostraram que existem casos em que um ou o outro é mais vantajoso e a utilização do modelo *multi-agent* dentro da abordagem feita por este projeto não trouxe diferenças significativas em relação ao modelo *single-agent* quando se compara o modelo deslocado no tempo com o não deslocado, porque o *multi-agent* foi melhor que o *single-agent* quando houve deslocamento no tempo, entretanto foi pior que o *single-agent* quando não houve deslocamento no tempo.

Obteve-se um impacto positivo na utilização de redes recorrentes na modelagem do ator-crítico, as operações foram mais estáveis e o lucro foi superior ao *buy-and-hold* e superior ao modelo similar, quando não utiliza-se a rede recorrente.

Presume-se que existem muitos outros estudos e testes que podem ser executados para que tenha a possibilidade de criar-se um modelo comercial, confiável e rentável.

Acredita-se que a modelagem e a simplificação de um MDP é válida (devido aos experimentos e revisão bibliográfica) dentro do escopo abordado neste trabalho, entretanto outra possibilidade seria modelar o sistema como um POMDP devido à parcialidade de informações que são transmitidas ao agente nos estados, mesmo que seja utilizado como "premissa" que uma pessoa é capaz de operar utilizando os indicadores propostos, uma pessoa de fato os utiliza e, além disso, existem outros fatores a serem levados em consideração, tais como: dados de notícias, um *dt* preferencial para realizar as operações, parcialidade na negociação do ativo, avaliação qualitativa da empresa, avaliação financeira, emoções e fatores psicológicos que podem influenciar o humano na tomada de decisão de compra ou venda do ativo. O fato de a LSTM ter um desempenho superior ao MLP no quesito estabilidade e até alguns resultados prova que o sistema é um POMDP e a inércia da LSTM contribui para solucionar o problema proposto. Isso abre possibilidade de uma linha de pesquisa na área com o intuito de modelar sistemas de RL utilizando POMDP.

Acredita-se que o algoritmo apresentado (MADDPG e DDPG) é parte do todo e há a necessidade de estudar as melhores redes para abstrair melhor os dados gerados nos experimentos de modo que o agente possa compreender o mundo da melhor forma. Comprovado pela experimentação executada no ultimo experimento apresentado na seção 6.5, há um impacto significativo na modelagem da rede proposta pelo ator-crítico.

Sugere-se estudos de redes mistas na construção do ator-crítico como uma evolução desse trabalho, criando uma rede parte convolucional (responsável por filtrar e gerar os indicadores utilizados no processo de avaliação), uma camada recorrente (responsável pela análise temporal dos dados) e por fim uma camada densa responsável pela tomada de decisão (podendo ser contínua ou discreta).

No capítulo 7 foram feitas algumas análises dos resultados do projeto, pode-se tirar pontos de melhoria em função dessas análises, tais como: implementação de diferentes modelos de *exploration*, na seção 2.1.2 dentro do capítulo da revisão bibliográfica apresenta modelos que podem ter maior aderência ao problema proposto do que a utilização do noise como técnica de *exploration*; teste comparando a implementação em um espaço de estados discreto e um contínuo, devido ao problema de convergência que surgiu no decorrer do desenvolvimento do projeto.

A função reforço apresentada na metodologia do trabalho não atribui necessariamente o reforço ao impacto da tomada de decisão do agente no momento em que ele a executa, o ganho no tempo é responsabilidade de ações passadas que não são informadas para o agente no momento em que ele executa uma nova decisão, acredita-se que com a aplicação de um modelo de reforço que de ao agente a variação de lucro que se obteve ao executar tal tomada de decisão pode fazer com que haja uma convergência melhor e que o agente aprendesse a executar ações mais lucrativas.

Acredita-se ser positivo o impacto dessa pesquisa no cenário acadêmico, comparando o trabalho desenvolvido e os projetos apresentados na revisão bibliográfica, tem-se certa similaridade aos projetos estudados na área, em sua maioria trabalhos recentes (2020) e de alta relevância. Desenvolver o problema de negociação de ativos utilizando MARL é uma abordagem diferente da comumente aplicada nessa área de pesquisa, a utilização de uma rede Bi-LSTM também não é comum nesta categoria de aplicação. Devido a estes motivos, acredita-se no impacto positivo da pesquisa no cenário acadêmico, mesmo que seus resultados comerciais não sejam tão relevantes.

REFERÊNCIAS

BANERJEE, D. Forecasting of Indian stock market using time-series ARIMA model. In: 2014 2nd International Conference on Business and Information Management (ICBIM). [S.l.: s.n.], 2014. P. 131–135.

BITTENCOURT, J. A. et al. Análise da relação entre o perfil de investidor, a realidade do mercado de renda fixa e variável e a teoria da aversão à perda, 2018.

BOWLING, Michael. Approved for Public Release. **Disp**, v. 1, n. 1, p. 1, 1998.

CARTA, Salvatore et al. A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning. **Applied Intelligence**, Applied Intelligence, n. 1, 2020. ISSN 15737497. DOI: 10.1007/s10489-020-01839-5.

DONG, H.; DING, Z.; ZHANG, S. **Deep Reinforcement Learning: Fundamentals, Research and Applications**. [S.l.]: Springer Singapore, 2020. ISBN 9789811540950.

ELDER, A. **Aprenda a Operar No Mercado de Ações**. [S.l.]: ELSEVIER/ALTA BOOKS, 2005. ISBN 9788535218985.

FRANSES, Philip Hans; VAN DIJK, Dick. Forecasting stock market volatility using (non-linear) Garch models. **Journal of Forecasting**, v. 15, n. 3, p. 229–235, 1996. ISSN 02776693. DOI: 10.1002/(sici)1099-131x(199604)15:3<229::aid-for620>3.3.co;2-v.

GRAESSER, L.; KENG, W.L. **Foundations of Deep Reinforcement Learning: Theory and Practice in Python**. [S.l.]: Pearson Education, 2019. (Addison-Wesley Data & Analytics Series). ISBN 9780135172483.

GRAN, Petter K.; HOLM, A. J. K.; SØGÅRD, S. G. A Deep Reinforcement Learning Approach to Stock Trading, 2019.

HSIEH, Tsung Jung; HSIAO, Hsiao Fen; YEH, Wei Chang. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. **Applied Soft Computing Journal**, Elsevier B.V., v. 11, n. 2, p. 2510–2525, 2011. ISSN 15684946. DOI: 10.1016/j.asoc.2010.09.007.

HU, Junling; WELLMAN, Michael P. Multiagent Reinforcement Learning in Stochastic Games. **Intelligence**, p. 1–36, 1999.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement Learning: A Survey. **Journal of Artificial Intelligence Research**, p. 237–285, 1996.

KIM, M. **Half & Half Donation Series Step by Step Beginner to Advanced Stock Trading: Fundamental Analysis, Basic Technical Analysis**. [S.l.]: MINJUN KIM, 2015. (Half & Half Donation Series).

KNUTH, Donald E. **CMT Level I 2020: An Introduction to Technical Analysis**. [S.l.]: Wiley, 2020. ISBN 9781119674375.

LAPAN, M. **Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more**. [S.l.]: Packt Publishing, 2018. ISBN 9781788839303.

LEE, J. W. Stock price prediction using reinforcement learning. **IEEE International Symposium on Industrial Electronics**, v. 1, p. 690–695, 2001. DOI: 10.1109/isie.2001.931880.

LEE, Jae Won; O, Jangmin. A multi-agent q-learning framework for optimizing stock trading systems. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 2453, p. 153–162, 2002. ISSN 16113349. DOI: 10.1007/3-540-46146-9_16.

LEE, Jae Won et al. A Multiagent Approach to Q-Learning for Daily Stock Trading. **IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans**, v. 37, n. 6, p. 864–877, 2007. ISSN 1083-4427. DOI: 10.1109/TSMCA.2007.904825.

LELE, Shreyas. Stock Market Trading Agent Using On-policy Reinforcement Learning Algorithms, p. 1–6, 2020.

LI, Yuming; NI, Pin; CHANG, Victor. Application of deep reinforcement learning in stock trading strategies and stock forecasting. **Computing**, Springer Vienna, v. 102, n. 6, p. 1305–1322, 2020. ISSN 14365057. DOI: 10.1007/s00607-019-00773-w. Disponível em: <https://doi.org/10.1007/s00607-019-00773-w>.

LITTMAN, Michael L. **Markov games as a framework for multi-agent reinforcement learning**. [S.l.]: Morgan Kaufmann Publishers, Inc., 1994. P. 157–163. DOI: 10.1016/b978-1-55860-335-6.50027-1. Disponível em: <http://dx.doi.org/10.1016/B978-1-55860-335-6.50027-1>.

LIU, Xiao Yang et al. FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance. **arXiv**, NeurIPS, p. 1–11, 2020. ISSN 23318422. arXiv: 2011.09607.

LOWE, Ryan et al. Multi-agent actor-critic for mixed cooperative-competitive environments. **Advances in Neural Information Processing Systems**, 2017-December, p. 6380–6391, 2017. ISSN 10495258. arXiv: 1706.02275.

NOWE, Ann; VRANCX, Peter; DE HAUWERE, Yann-Michaël. Game Theory and Multi-agent Reinforcement Learning. In: [s.l.: s.n.], jan. 2012. P. 30. ISBN 978-3-642-27645-3. DOI: 10.1007/978-3-642-27645-3_14.

PAI, Ping Feng; LIN, Chih Sheng. A hybrid ARIMA and support vector machines model in stock price forecasting. **Omega**, v. 33, 2005. ISSN 03050483. DOI: 10.1016/j.omega.2004.07.024.

PALMAS, A. et al. **The The Reinforcement Learning Workshop: Learn how to apply cutting-edge reinforcement learning algorithms to a wide range of control problems**. [S.l.]: Packt Publishing, 2020. ISBN 9781800209961.

PUGA, R.; RODRIGUES, M. **Formação de traders: Faça dinheiro na bolsa com a análise técnica**. [S.l.]: Elsevier Brasil, 2010.

QIE, Han et al. Joint Optimization of Multi-UAV Target Assignment and Path Planning Based on Multi-Agent Reinforcement Learning. **IEEE Access**, IEEE, v. 7, p. 146264–146272, 2019. ISSN 21693536. DOI: 10.1109/ACCESS.2019.2943253.

RANGANATHAM, M. **Investment Analysis and Portfolio Management**. [S.l.]: Pearson Education/Dorling Kindersley (India), 2006. ISBN 9788177582291.

RATNAYAKA, R. M.Kapila Taranga et al. A hybrid statistical approach for stock market forecasting based on Artificial Neural Network and ARIMA time series models. **2015 International Conference on Behavioral, Economic and Socio-Cultural Computing, BESC 2015**, Besc, p. 54–60, 2015. DOI: 10.1109/BESC.2015.7365958.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. [S.l.]: Pearson, 2016. (Always learning). ISBN 9781292153964.

SATTAROV, Otabek et al. Recommending cryptocurrency trading points with deep reinforcement learning approach. **Applied Sciences (Switzerland)**, v. 10, n. 4, 2020. ISSN 20763417. DOI: 10.3390/app10041506.

SCHIAVONE, D. **Technical Analysis: Trading Indicators and Charting**. [S.l.]: Independently Published, 2019. (Trading Series). ISBN 9781695611436.

SEWAK, M. **Deep Reinforcement Learning: Frontiers of Artificial Intelligence**. [S.l.]: Springer Singapore, 2019. ISBN 9789811382857.

SEZER, Omer Berat; OZBAYOGLU, Ahmet Murat. Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. **Applied Soft Computing Journal**, Elsevier B.V., v. 70, p. 525–538, 2018. ISSN 15684946. DOI: 10.1016/j.asoc.2018.04.024.

SUTTON, R.S.; BARTO, A.G. **Reinforcement Learning, second edition: An Introduction**. [S.l.]: MIT Press, 2018. (Adaptive Computation and Machine Learning series). ISBN 9780262352703.

THIBAUT THÉATE, Damien Ernst. An Application of Deep Reinforcement Learning to Algorithmic Trading, 2020.

TOKIC, Michel. Adaptive ϵ-greedy exploration in reinforcement learning based on value differences. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, 6359 LNAI, September 2010, p. 203–210, 2010. ISSN 03029743. DOI: 10.1007/978-3-642-16111-7_23.

VAZIRANI, S.D.C.P.U. **Algoritmos**. [S.l.]: McGraw Hill Brasil, 2009. ISBN 9788563308535.

WILEY, John. **CMT Level II 2016: Theory and Analysis**. [S.l.]: Wiley, 2015. ISBN 9781119222705.

WILEY, John. Sign- and Volatility-Switching Arch Models : Theory and Applications To International. v. 12, n. 1, p. 49–65, 2011.

WU, Xing et al. Adaptive stock trading strategies with deep reinforcement learning methods. **Information Sciences**, Elsevier Inc., v. 538, p. 142–158, 2020. ISSN 00200255. DOI: 10.1016/j.ins.2020.05.066.

XIONG, Zhuoran et al. Practical deep reinforcement learning approach for stock trading. **arXiv**, n. 1, 2018. eprint: 1811.07522.

YANG, Hongyang et al. Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy. **SSRN Electronic Journal**, 2020. DOI: 10.2139/ssrn.3690996.

ZHANG, Zihao; ZOHREN, Stefan; ROBERTS, Stephen. Deep reinforcement learning for trading. **arXiv**, v. i, Chan 2009, 2019. ISSN 23318422.