UNIVERSIDADE FEDERAL DE GOIÁS / INSTITUTO DE INFORMÁTICA

APRENDIZADO POR REFORÇO PARA OTIMIZAÇÃO DE ESTOQUE

Gustavo Barbosa

UFG
UNIVERSIDADE FEDERAL DE GOIÁS

UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)

GUSTAVO BARBOSA

APRENDIZADO POR REFORÇO PARA OTIMIZAÇÃO DE ESTOQUE

Goiânia
2024

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): **GUSTAVO BARBOSA**

Título do trabalho: **APRENDIZADO POR REFORÇO PARA OTIMIZAÇÃO DE ESTOQUE**

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [**X**] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Gustavo Barbosa**, **Discente**, em 16/02/2024, às 22:01, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Marques Federson**, **Professor do Magistério Superior**, em 12/09/2024, às 11:00, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4383377** e o código CRC **F9EB8000**.

Referência: Processo nº 23070.008382/2024-77 SEI nº 4383377

GUSTAVO BARBOSA

APRENDIZADO POR REFORÇO PARA OTIMIZAÇÃO DE ESTOQUE

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.
Orientador: Prof. Dr. Fernando Marques Federson

Goiânia
2024

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

BARBOSA, GUSTAVO
APRENDIZADO POR REFORÇO PARA OTIMIZAÇÃO DE ESTOQUE [manuscrito] / GUSTAVO BARBOSA. - 2024.
88 f.

Orientador: Prof. Dr. FERNANDO MARQUES FEDERSON.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Informática (INF), Inteligência Artificial, Goiânia, 2024.

1. inteligência artificial. 2. aprendizado por reforço. 3. estoque. I. FEDERSON, FERNANDO MARQUES, orient. II. Título.

CDU 004

GUSTAVO BARBOSA

APRENDIZADO POR REFORÇO PARA OTIMIZAÇÃO DE ESTOQUE

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Data da Aprovação: 08 de fevereiro de 2024.

Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)

Prof. Dr. Aldo André Díaz Salazar
Coordenador de TCC do BIA (INF-UFG)

Prof. Dr. Vinícius Sebba Patto
Coordenador do BIA (INF-UFG)

Prof. Me. Leonardo Antônio Alves
(INF-UFG)

GUSTAVO BARBOSA

APRENDIZADO POR REFORÇO PARA OTIMIZAÇÃO DE ESTOQUE

RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **Aprendizado por Reforço**. Uma ilustração e sua narrativa descrevem os períodos de trabalho. Os Apêndices contêm os Termos de Aceite de Entrega e os resultados obtidos durante cada período de trabalho.

Palavras-chave: inteligência artificial, aprendizado por reforço, estoque.

ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **Reinforcement Learning**. An illustration and its narrative describe the work periods. The Appendices contain the Delivery Acceptance Terms and the results obtained during each work period.

Keywords: artificial intelligence, reinforcement learning, stock.

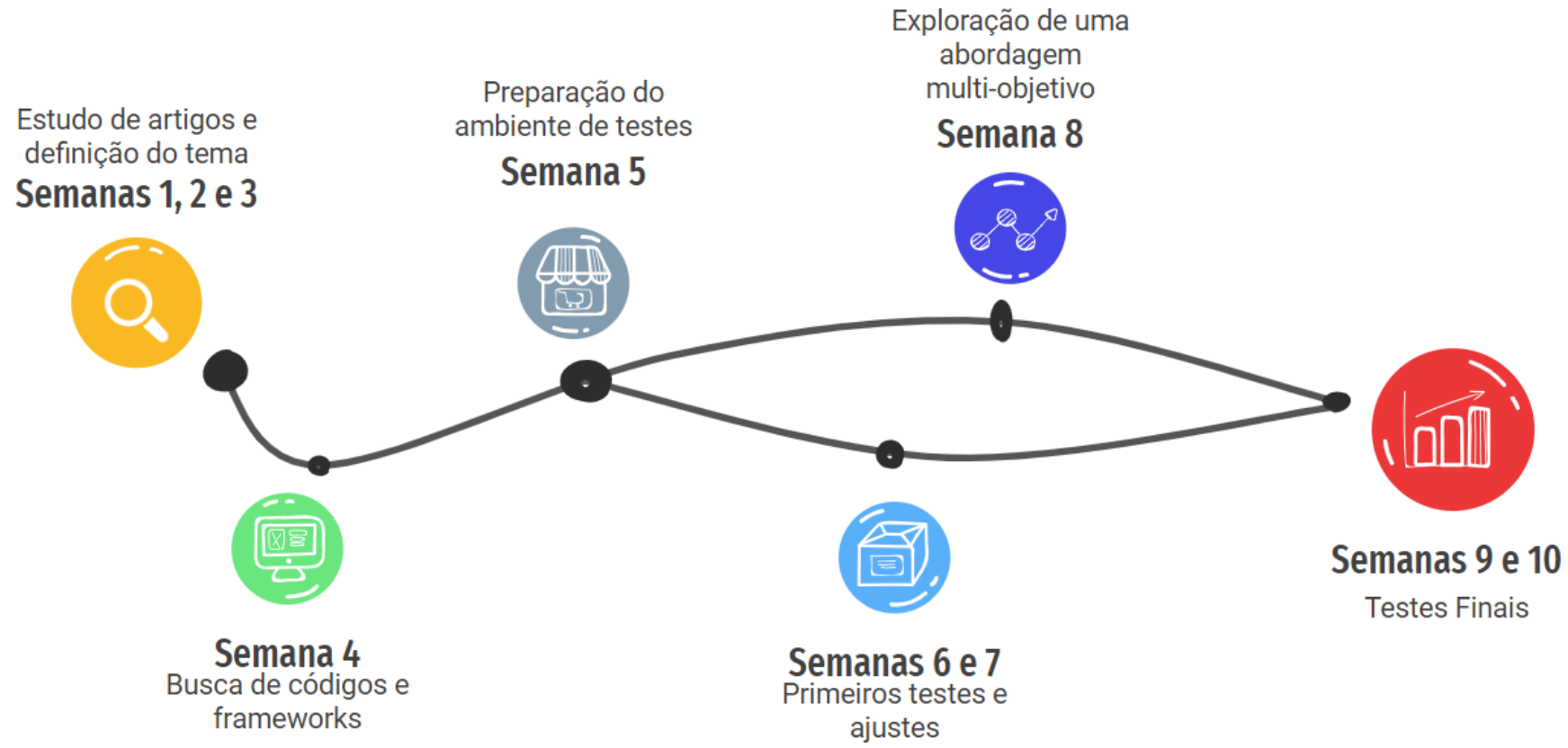
Goiânia

2024

Minha Jornada

Especialista em: Aprendizado por Reforço

Gustavo Barbosa



Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

MINHA JORNADA

Nome: Gustavo Barbosa

Especialidade: Aprendizado Por Reforço

Objetivo deste documento

Durante o processo da disciplina Residência em IA[1], foram gerados diversos resultados na construção da minha especialização. A cada semana, um conjunto de resultados foi formalizado por um Termo de Aceite de Entrega e avaliado por uma banca, considerando o planejado e o realizado para o período. Este documento tem como objetivo descrever esses resultados obtidos, fazendo referência aos Termos de Aceite de Entrega e seus documentos associados.

Minha Jornada

Minha jornada de pesquisa começou na **Semana 1** com a definição da área de especialização: Aprendizado por Reforço (AR) ou *Reinforcement Learning* (RL). Inicialmente, estava incerto quanto ao foco exato do meu trabalho nessa área, mas tinha interesse em explorar a intersecção entre aprendizado por reforço e séries temporais. Na **Semana 1**, analisei termos relacionados à Inteligência Artificial presentes no site do evento *Conference on Computational Science and Computational Intelligence* (CSCI'23). Identifiquei termos relevantes ao campo de Aprendizado por Reforço, tais como Agentes Inteligentes, Aprendizado por Reforço e Redes Neurais. A partir desses termos, realizei uma pesquisa de artigos relacionados. Muitos artigos apresentavam ideias sobre como o AR pode ser aplicado em diferentes campos, como jogos, energia, finanças, entre outros. Consequentemente, classifiquei os artigos por área e os categorizei com base em conceitos, metodologias ou aplicações. A classificação e o contexto dos artigos, resumidos brevemente, estão presentes no **Apêndice 1**.

[1] Dez semanas, entre setembro de 2023 e janeiro de 2024.

Prosseguindo em direção à área de interesse, na **Semana 2,** pesquisei artigos relacionados ao Aprendizado por Reforço aplicado a séries temporais. Dos artigos lidos, encontrei métodos com propostas bastante variadas. Por exemplo, alguns aplicam o AR diretamente nas séries temporais para prever valores futuros, outros utilizam essa técnica para escolher hiperparâmetros de modelos de previsão, há aqueles que empregam o AR para combinar previsões de vários modelos distintos, além dos que o utilizam para otimização de estoque. Uma visão geral desses artigos lidos é apresentada no **Apêndice 2**, trazendo uma introdução ao problema e à metodologia utilizada pelos autores.

Ao estudar esses artigos, percebi que, apesar de ter participado da Disciplina de Aprendizado por Reforço durante o Bacharelado em Inteligência Artificial, havia algoritmos desconhecidos para mim e outros que precisava revisar. Por isso, na **Semana 3**, decidi que, enquanto estudava mais artigos, também faria o estudo dos algoritmos. Pesquisamos, então, os algoritmos mencionados nos artigos - *Deep Q-Network* (DQN), *Deep Deterministic Policy Gradient* (DDPG) e *Augmented Random Search* (ARS). Um resumo sobre eles está disponível no **Apêndice 3**.

Nesse mesmo período, encontrei artigos relacionados às áreas de varejo e mercado financeiro. Foi então que decidi me especializar em AR no contexto do varejo, atuando no uso para otimização de estoque. Esta aplicação é importante pois permite às empresas reduzirem custos, aumentarem a eficiência e a satisfação do cliente e portanto, maximizar a lucratividade. Isso é alcançado por meio do equilíbrio entre a disponibilidade de produtos e o capital líquido necessário para manter o estoque. Nesse contexto, o AR é uma ferramenta para auxiliar na tomada de decisão de quanto pedir a cada dia, a fim de encontrar esse equilíbrio. Artigos que abordam este tema podem ser encontrados no **Apêndice 3**.

Com o propósito de avançar para uma etapa mais “prática”, na **Semana 4**, dediquei-me ao estudo de *frameworks* que facilitam a implementação dos algoritmos. Inicialmente, encontrei três que pareciam promissores. No entanto, foi apenas nas semanas seguintes que descobri outro, com o qual me adaptei melhor e que simplificou o estudo dos algoritmos implementados pela biblioteca a “*stable-baselines3*”.

Além disso, procurei por códigos que para auxiliar nos testes, seja através de implementações do ambiente de teste ou da execução de algum algoritmo. Os *frameworks* e códigos encontrados estão listados no **Apêndice 4**. Realizei também uma busca por dados de séries temporais de vendas no varejo e compilei esses bancos de dados em uma tabela para facilitar a visualização das informações (**Apêndice 4**). Por fim, busquei por *baselines* para comparar os resultados obtidos. A explicação dos *baselines* encontrados encontra-se também no **Apêndice 4**.

Na **Semana 5**, dediquei-me à preparação do ambiente de teste, baseando-me em um dos códigos descobertos na semana anterior. Isto incluiu ajustes sugeridos pelo artigo *"Deep Inventory Management"*, de Madeka et al. (2020). A criação de um ambiente próprio foi necessária, pois não encontrei nenhum totalmente pronto que satisfizesse as minhas necessidades. O resultado foi um ambiente que simula interações com o estoque, levando em conta os dados de demanda de um único produto (**Apêndice 5**). Se houver demanda em determinado dia, a quantidade correspondente é deduzida do estoque, no caso de demanda sem estoque suficiente, registra-se apenas o volume disponível. Além disso, integrei a dinâmica de recebimento aleatório dos produtos, já que os produtos nem sempre chegam nas datas previstas. A recompensa deste ambiente foi estipulada com base nos lucros obtidos com as vendas, após deduzir os custos de manutenção do estoque e os gastos com pedidos. Este ambiente foi desenvolvido em um *Colab Notebook*, e seus detalhes podem ser consultados no **Apêndice 5**.

Na **Semana 6**, testei os algoritmos *Deep Q Network* (DQN) e *Deep Deterministic Policy Gradient* (DDPG), obtendo resultados apenas com o DQN. Nesse período, percebi a necessidade de desencorajar a realização de pedidos diários, já que meu objetivo era que os pedidos ocorressem de maneira mais espaçada. Para isso, implementei um custo fixo por pedido, o que levou o agente DQN a realizar os pedidos de forma mais controlada, embora não de maneira “perfeita”, pois houve ocasiões em que o estoque se esgotou. Organizei os códigos e os agentes treinados em um github (**Apêndice 6**).

Na **Semana 7**, testei novos modelos e aperfeiçoei os já experimentados para melhorar o desempenho. Comparando com o *baseline*, obtive um aumento de 54%

no lucro utilizando o algoritmo ARS. Entretanto, mais tarde, nas entregas seguintes, percebi que essa comparação não havia sido justa, pois o *baseline* também necessitava de otimização. No entanto, isso serviu como indício de que estava na direção certa. Os métodos, resultados e algoritmos testados estão detalhados no **Apêndice 7**.

Na **Semana 8**, decidi explorar o Aprendizado por Reforço Multi-objetivo, já que, até aquele momento, havia focado apenas no controle do estoque de um único produto. O objetivo era permitir que meus agentes de Aprendizado por Reforço gerenciassem múltiplos produtos simultaneamente. O desafio consistia em como otimizar a recompensa, considerando que cada produto teria sua própria função de recompensa. Uma das abordagens mais simples, sugerida por Madeka et al. em *"Deep Inventory Management"*, era a combinação linear das recompensas para criar uma função unificada. Outra alternativa seria adentrar no campo do multi-objetivo propriamente dito, o qual também foi examinado durante a semana. No entanto, devido à sua complexidade e ao fato de estarmos nas últimas semanas, decidi manter o foco na otimização de um único produto. Os resultados desta semana podem ser encontrados no **Apêndice 8**.

Nas **Semanas 9 e 10**, concentrei-me em aprimorar meus agentes por meio da otimização dos hiperparâmetros dos algoritmos. Elaborei um catálogo detalhando os hiperparâmetros e como estes influenciam o desempenho de cada algoritmo. Além disso, compilei um resumo explicativo do funcionamento desses algoritmos. Este resumo se encontra disponível no **Apêndice 9**. Com o catálogo em mãos, a busca por hiperparâmetros tornou-se mais eficiente, permitindo-me alterar apenas aqueles com potencial para gerar bons resultados em meu estudo.

Ao todo, cinco algoritmos foram otimizados. Dentre eles, o *Soft Actor-Critic* (SAC) se destacou, apresentando resultados 12% superiores ao novo *baseline*, agora já otimizado. Este algoritmo foi capaz de manter um estoque reduzido e, ao mesmo tempo, aumentar os lucros. O detalhamento dos resultados, assim como o código-fonte, podem ser consultados no Apêndice 10.

Ao longo da minha jornada, consegui avançar consideravelmente na minha especialização em Aprendizado por Reforço. As últimas semanas foram repletas de

aprendizados, onde tive contato com diferentes conceitos cruciais do RL, desde a elaboração do ambiente até o domínio de frameworks e algoritmos da área. Essa experiência foi fundamental para aprofundar meu entendimento em cada fase de desenvolvimento de um projeto de RL. Mesmo com progressos na otimização de estoque, percebo que ainda há espaço para aperfeiçoamentos, especialmente na função de recompensa e na experimentação com diversos algoritmos. Ao concluir esta etapa de formação, sinto que avancei de maneira significativa em direção ao meu objetivo de me consolidar como um especialista em Aprendizado por Reforço e estou entusiasmado para explorar a aplicação do RL em novos desafios.

APÊNDICE 1

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: **19 de out. de 2023**

Participantes da Entrega **[matriculados em Residência em IA]:**

Gustavo Barbosa e Luís Augusto

Entrega: **[descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

O foco do trabalho do nosso time é o campo de aprendizado por reforço. Nesse sentido, para realização da atividade proposta para o Gate dessa semana, foram analisados os diversos termos relacionados à área de IA presentes no site do evento Conference on Computational Science and Computational Intelligence (CSCI'23). Dentre os termos presentes, foi identificado aqueles mais relacionadas ao campo de aprendizado por reforço:

- Intelligent agents
- Reinforcement learning
- Neural networks

Onde o termo principal é Reinforcement learning (RL). Já Intelligent agents também se encaixa visto que o foco é desenvolver agentes capazes de aprender e tomar decisões cada vez melhores através das experiências obtidas da interação com o ambiente. Quanto ao termo Neural networks, a pretensão é estudar os algoritmos de RL que utilizam redes neurais no aprendizado visto que hoje são as estratégias que atingem os melhores resultados.

Dentre os três termos escolhidos, foi selecionado o "Reinforcement Learning" para ser feita uma pesquisa pelos trabalhos presentes nos anais do CSCI dentro das diversas edições realizadas. Nesta pesquisa foram encontrados artigos de 2016 até 2019 aos quais foi obtido o acesso. Um breve resumo contextualizando o cenário em que cada um deles aplica aprendizado por reforço foi escrito no documento:

https://drive.google.com/file/d/1rT7SOV_aWJ1FZCN6lfrlSKUg9WLek4gH/view?usp=share_link

Vários desses artigos trazem ideias de como o RL pode ser usado em diversas áreas diferentes como, por exemplo, em jogos, setor de energia, mercado financeiro, entre outros. Dessa forma, com uma estruturação e organização de processos e ideias aprendido durante a disciplina de empreendedorismo do BIA, foi feita uma classificação dos trabalhos em três categorias diferentes. Uma visualização dessa classificação se encontra no link abaixo, além dos artigos do CSCI foram incluídos outros artigos lidos fora do escopo do evento relacionados à RL.

https://drive.google.com/file/d/1wJJJR-Bug5G7ZFLy0ijJF_6lDUlleYzI/view?usp=share_link

A primeira categoria se trata do “conceito” onde se encaixam os artigos lidos cujo propósito é apresentar a ideia geral de como o aprendizado por reforço pode ser utilizado na área em questão. Já na segunda categoria, se encontram as “ferramentas (metodologia)”, onde foram incluídos os artigos que focam em apresentar uma estratégia ou algoritmo para o aprendizado do agente e realizam testes em ambientes mais simples. Já na última categoria de “aplicação (Materialização)” são enquadrados os artigos que focam em fazer testes e que trazem resultados dos métodos de RL aplicados em situações reais.

Em grande parte dos trabalhos lidos é apresentado uma introdução a conceitos básicos de RL, por isso foi uma excelente oportunidade para revisar os fundamentos da área que já haviam sido estudados na disciplina de RL no sexto período do bacharelado em IA. Dentre os conceitos revisados estão:

- Processo de decisão de Markov
- Ambiente, Agente, Estado, Ação, Recompensa
- Retorno esperado
- Função de valor de estado e de estado-ação
- Políticas

Alguns conceitos relacionados às estratégias de aprendizado também foram revisadas, como:

- Off-policy (Ex: Q-learning)
- On-policy (Ex:Sarsa)
- Model-Based
- Model-free

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Para a próxima semana planejamos continuar a busca e leitura de artigos na área de RL, porém agora com foco em trabalhos voltados para o contexto de séries temporais. O principal objetivo é dar os primeiros passos para entender as metodologias utilizadas e como as técnicas de RL são adaptadas para esse cenário de séries temporais.

Observação: [caso precise fazer alguma observação, de qualquer “natureza”]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

LUANA GUEDES BARROS MARTINS: Go!

Revisão dos artigos

Neste texto é apresentado uma breve contextualização dos trabalhos entre 2016 à 2019 presentes no evento Conference on Computational Science and Computational Intelligence (CSCI) relacionadas a Reinforcement Learning (RL). Os artigos lidos apresentam várias ideias e aplicações do uso RL em diferentes áreas como setor elétrico, mercado financeiro, jogos, robótica e outros.

Artigos de 2016

Sahba, em [1], propõe a utilização de RL para segmentação de uma sequência de imagens de um vídeo, o que permite um ajuste dinâmico da segmentação e correções manuais dos usuários no meio do processo. Nesse contexto, as imagens do vídeo são quebradas em sub-imagens do qual são extraídas features pré-estabelecidas para representar um estado. Em relação às ações, elas são definidas como instruções que mudam a cor de cada pixel para ajustar a máscara de segmentação. Já as recompensas são calculadas ao comparar a segmentação feita pelo agente com versões manualmente segmentadas das imagens de treino. O algoritmo utilizado é o Deep Q-learning para aproximar os valores de estado-ação.

Artigos de 2017

Ensinar robôs autônomos a planejar e seguir uma rota sozinho desviando de obstáculos é uma tarefa complicada. Contudo, em [2], kwak, Yoon e Roh propõem a utilização de RL, com Deep learning, para ensinar múltiplos robôs a aprenderem a seguir uma rota desviando de obstáculos pelo caminho. O algoritmo utilizado foi Deep Q-learning (DQN) cuja a entrada da rede neural consiste em dados de um scanner com lasers obtendo distâncias e ângulos dos obstáculos, o ponto da rota mais próximo, velocidade, orientação e ações anteriores. No espaço das ações é composto por 5 movimentos possíveis, dentre eles estão mover para frente ou para trás, rotacionar no sentido horário ou anti-horário e parar. No artigo, como ponto positivo para reprodutibilidade, é apresentado todos os hiperparâmetros necessários para construir a DQN.

Em relação a [3], é proposto por Lu et al. um cenário relacionado ao setor elétrico. Nesse ambiente temos o operador da rede elétrica que fornece a energia ao provedor de serviços e este, por sua vez, repassa a energia elétrica para os clientes. Dessa forma, um agente é o provedor de serviço que vai ajustar o preço cobrado de cada cliente de acordo com o perfil de consumo e o preço da energia comprada do operador, a fim de maximizar os lucros obtidos. Além disso, um segundo agente é proposto para representar os clientes cujo objetivo é minimizar os gastos com energia elétrica, contudo outro objetivo também modelado é o desconforto que precisa ser minimizado, onde ele aumenta conforme o cliente deixa de usar equipamentos elétricos para economizar energia. Assim, um dos agentes deve aumentar o lucro do provedor e outro diminuir o gasto do cliente. O algoritmo proposto para aprendizado se trata do Q-Learning.

Em [4], o Boyd e Barbosa apresentam uma maneira de aplicar RL para criação da inteligência artificial de inimigos em videogames. Como ferramenta é utilizado a Unreal Engine para criação do ambiente cujos agentes interagem, bem como o sistema blueprint

que permite a implementação do código de maneira visual. A estratégia adotada para o aprendizado foi Q-learning, onde o espaço de estados é discreto composto por 5 indicadores: Visão do inimigo, sofrendo dano, causando dano e vida em estado crítico. Enquanto as ações que o inimigo (agente) pode realizar se resumem a apenas 3: correr, atacar e explorar. Incluindo algumas restrições de combinações, a tabela de valor estado-ação é pequena, com apenas 80 possibilidades.

O trabalho [5] de Rioual et al. voltado para parte de energia elétrica, assim como em [3], apresenta uma estratégia para o gerenciamento da energia elétrica em grandes sistemas de IoT. Os sensores de um sistema IoT possuem diferentes restrições quanto a necessidade de energia elétrica, alguns possuem bateria e suportam ficar ligado algum tempo sem energia, enquanto outros dependem de um acesso constante à energia para funcionar. Dessa forma, RL é aplicado para gerenciar o uso de energia do sistema a fim de minimizar os custos com energia elétrica, mas manter todos componentes da rede funcionando da maneira esperada.

Artigos de 2018

No artigo [6], Sabharwal e Anjum fazem uma revisão sobre os grandes impactos gerados pelos avanços dos consultores robóticos no setor financeiro. Graças ao machine learning, tais consultores são capazes de analisar milhares de dados e oferecer serviços como detecção de fraude, construção de portfólios de investimento, análise de crédito, entre outros. Em especial, no artigo é mencionado o potencial do RL nessa área e cita outros trabalhos que aplicaram essa estratégia para construção automática de portfólio de investimentos.

Artigos de 2019

Estratégias de RL são aplicadas para um agente jogar o “jogo da cobrinha” dos celulares antigos da Nokia, em [7]. Almalki e Wocjam aplicam o algoritmo SARSA em uma versão personalizada do jogo onde são adicionados novos desafios para o agente lidar.

Asher et al. em [8] apresentam uma forma de visualizar o trabalho em equipe em um ambiente onde há vários agentes em busca de um mesmo objetivo. No artigo é utilizado um ambiente presa-predador a título de exemplo, onde existem 3 agentes predadores cujo alvo é um quanto agente que representa a presa. É utilizada a estratégia multi-agent deep deterministic policy gradient (MADDPG), onde há uma única rede central como crítico que tem acesso às experiências coletadas por todos os agentes. Para mensurar essa cooperação entre os predadores, o autor propõe um método de perturbação nas experiências de um predador e analisa o impacto disso nas ações dos demais agentes.

Referências:

[1] F. Sahba, "Deep Reinforcement Learning for Object Segmentation in Video Sequences," 2016 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2016, pp. 857-860, doi: 10.1109/CSCI.2016.0166.

[2] N. Kwak, S. Yoon and K. Roh, "Learning Motion Policy for Mobile Robots Using Deep

Q-Learning," 2017 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2017, pp. 805-810, doi: 10.1109/CSCI.2017.139.

[3] R. Lu, S. H. Hong, X. Zhang, X. Ye and W. S. Song, "A Perspective on Reinforcement Learning in Price-Based Demand Response for Smart Grid," 2017 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2017, pp. 1822-1823, doi: 10.1109/CSCI.2017.327.

[4] R. A. Boyd and S. E. Barbosa, "Reinforcement Learning for All: An Implementation Using Unreal Engine Blueprint," 2017 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2017, pp. 787-792, doi: 10.1109/CSCI.2017.136.

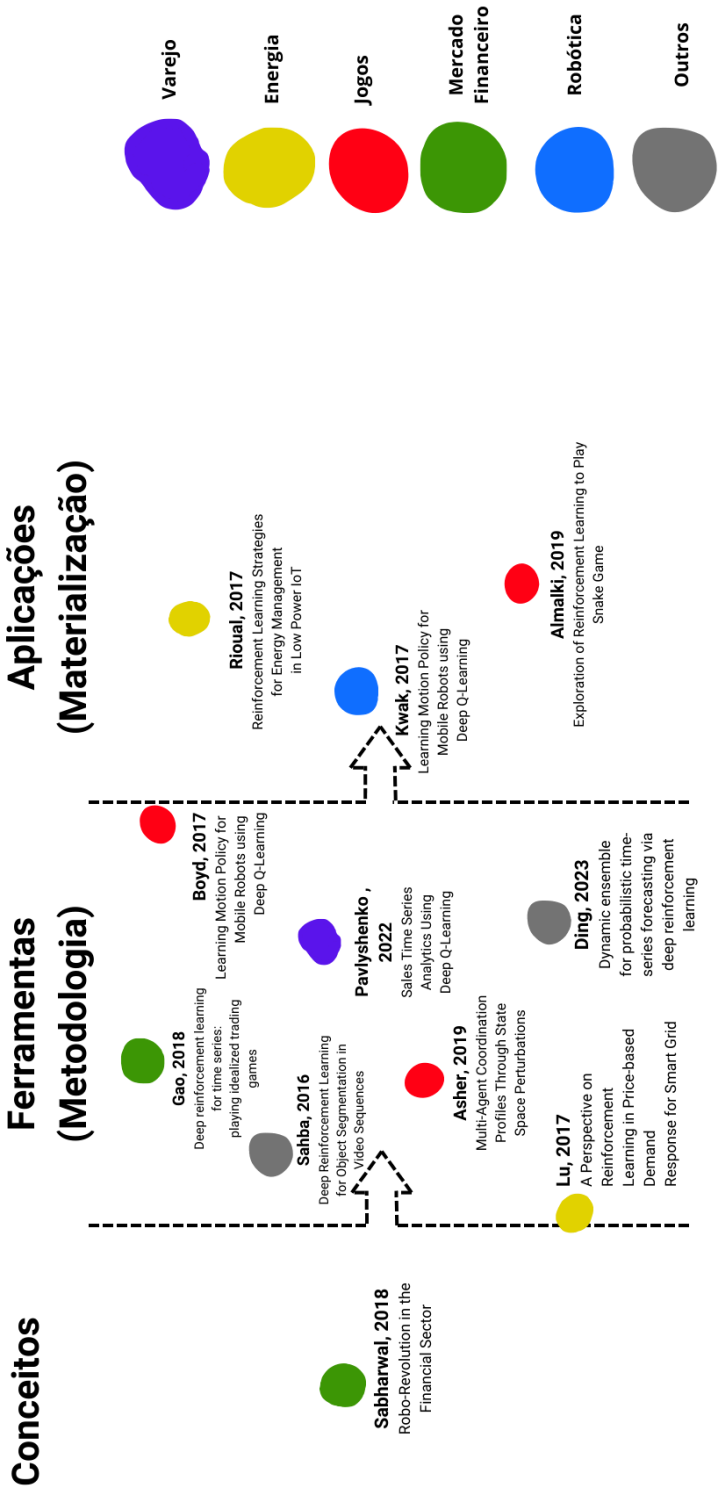
[5] Y. Rioual, J. Laurent, E. Senn and J. -P. Diguet, "Reinforcement Learning Strategies for Energy Management in Low Power IoT," 2017 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2017, pp. 1377-1382, doi: 10.1109/CSCI.2017.240.

[6] C. L. Sabharwal and B. Anjum, "Robo-Revolution in the Financial Sector," 2018 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2018, pp. 1289-1292, doi: 10.1109/CSCI46756.2018.00249.

[7] A. J. Almalki and P. Wocjan, "Exploration of Reinforcement Learning to Play Snake Game," 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2019, pp. 377-381, doi: 10.1109/CSCI49370.2019.00073.

[8] D. Asher, M. Garber-Barron, S. Rodriguez, E. Zaroukian and N. Waytowich, "Multi-Agent Coordination Profiles through State Space Perturbations," 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2019, pp. 249-252, doi: 10.1109/CSCI49370.2019.00051.

Classificação dos artigos


Varejo
Energia
Jogos
Mercado Financeiro
Robótica
Outros
Aplicações (Materialização)
Rioual, 2017
Reinforcement Learning Strategies for Energy Management in Low Power IoT
Kwak, 2017
Learning Motion Policy for Mobile Robots using Deep Q-Learning
Almalki, 2019
Exploration of Reinforcement Learning to Play Snake Game
Ferramentas (Metodologia)
Boyd, 2017
Learning Motion Policy for Mobile Robots using Deep Q-Learning
Gao, 2018
Deep reinforcement learning for time series: playing idealized trading games
Sahba, 2016
Deep Reinforcement Learning for Object Segmentation in Video Sequences
Pavlyshenko , 2022
Sales Time Series Analytics Using Deep Q-Learning
Asher, 2019
Multi-Agent Coordination Profiles Through State Space Perturbations
Ding, 2023
Dynamic ensemble for probabilistic time-series forecasting via deep reinforcement learning
Lu, 2017
A Perspective on Reinforcement Learning in Price-based Demand Response for Smart Grid
Conceitos
Sabharwal, 2018
Robo-Revolution in the Financial Sector

APÊNDICE 2

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: **26 de out. de 2023**

Participantes da Entrega **[matriculados em Residência em IA]:**

Gustavo Barbosa
Luís Augusto

Entrega: **[descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Para o *stage* 2, o time de Reinforcement Learning (RL) focou em estudar as formas de aplicação de RL em séries temporais por meio de trabalhos já publicados. Para encontrar os artigos científicos na área, foi feita uma busca dos trabalhos citados nas referências de alguns trabalhos lidos durante o *stage* 1 relacionados ao tema de séries temporais, bem como a pesquisa em sites que fazem o catálogo desses artigos.

Além disso, em um primeiro contato com a ferramenta Parsifal, foi estudado um pouco sobre construção de strings de busca para encontrar artigos nas bibliotecas disponíveis. Dessa forma, foi realizada uma busca inicial por surveys que juntassem as áreas de RL e séries temporais para obter visão mais ampla dessa área, mas até o momento nenhum foi encontrado. Contudo, com o Parsifal foram obtidos artigos que propõem alguns métodos para uso RL e séries.

Dos artigos lidos até o momento, há métodos com propostas bem diferentes, por exemplo, alguns aplicam RL diretamente na série temporal, outros usam RL para escolher hiperparâmetros de modelos de forecasting, há aqueles que usam RL para combinar as previsões de vários modelos distintos, além dos que usam para otimização de estoque. No documento abaixo é apresentado uma visão geral desses artigos lidos, trazendo uma introdução ao problema e metodologia utilizada:

https://docs.google.com/document/d/1j0RlVg-zmyfADe4NR8WsVKRnhbms-FvT4K4L68YHFP0/edit?usp=sharing

Já os artigos completos podem ser encontrados na pasta do drive:

https://drive.google.com/drive/folders/1bSw9WaRtUBQ_W94DWSSMA248D0zrT9qO?usp=sharing

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Como atividades para próximo *stage*, foi planejado:

- Estudar sobre os algoritmos RL utilizados nos artigos vistos, assim como outros que possamos achar relevantes durante o processo.

- Continuar procurando e estudando artigos sobre RL e séries temporais
- Começar a procurar por algum dataset para fazer testes

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

LUANA GUEDES BARROS MARTINS: Go!

Revisão dos artigos

Este texto fornece uma visão geral dos artigos lidos pelos membros do time de Reinforcement Learning (RL) durante o *Stage 2*. O objetivo deste estudo é compreender melhor as metodologias utilizadas para aplicar RL no contexto de séries temporais. Dentre os métodos encontrados, há a aplicação de RL diretamente na série, uso de RL para selecionar os hiperparâmetros dos modelos e também abordagens em que RL é usado para combinar/selecionar os resultados de vários modelos de forecasting.

Analisar os padrões de dados temporais e tomar uma ação a partir disso é útil em diversos contextos. Por exemplo, no varejo, decisões como alterar o preço dos produtos e repor o estoque de acordo com a demanda de cada produto são medidas essenciais para aumentar o lucro obtido. É nesse cenário que Pavlyshenko, em [1], propõe um método utilizando Deep Q-Learning (DQN) para tomar tais decisões de acordo com o comportamento de vendas ao longo do tempo.

O primeiro problema abordado no trabalho [1], se trata justamente da determinação de uma estratégia de precificação para um produto a fim de maximizar as vendas. O ambiente nessa abordagem é representado por um modelo paramétrico que relaciona a quantidade de vendas com o aumento de preço, além disso uma série sintética que simula demanda pelo produto é utilizada para representar o estado. Enquanto que as ações são discretizadas em incrementos fixos no preço. Assim, o agente analisa a demanda pelo produto ao longo do tempo e decide se vai aumentar ou diminuir o preço em questão. Os resultados mostram que o Agente treinado com DQN, acaba por encontrar um incremento ideal que aumenta o lucro obtido.

Ainda em [1], a segunda aplicação abordada é a de reposição de produtos por demanda, onde os testes foram realizados com o dataset do kaggle 'Rossmann Store Sales'. Nesse problema, o objetivo é decidir o quanto comprar de um produto para repor o estoque baseado na expectativa de vendas no futuro. Para modelar o estado, o autor utilizou features como a quantidade de vendas no dia anterior, se há ou não promoção ativa e uma representação one-hot do dia da semana. Já o espaço das ações foi limitado a 7 possíveis

escolhas referente a quantidade que deverá ser comprada para suprir a demanda. Nesse caso, Pavlyshenko traz uma visão interessante dos resultados ao analisar o comportamento do agente em diferentes dias da semana.

Outra abordagem para o varejo, é tratada no artigo [6], o problema de otimização de estoque, o qual tem como objetivo aprimorar as operações de gerenciamento de estoque da empresa, assegurando o suprimento de produtos conforme a demanda, evitando desperdícios e os riscos associados ao excesso de estoque, ao mesmo tempo em que melhora o fluxo de caixa. Neste artigo, a abordagem Deep Q Network (DQN) é empregada para criar uma política que determine quando e quanto encomendar determinado produto.

Seguindo os princípios do Processo de Decisão de Markov (MDP), o estado é definido como a combinação da posição de estoque (estoque atual mais o pedido futuro) e o dia da semana. O espaço de ações (a_t) refere-se à quantidade de produtos que devem ser encomendados até o final do dia. A recompensa é calculada através da função r_t, a qual é definida como o mínimo entre a demanda do dia (d_t) e a quantidade de produtos pedidos (i_t), multiplicado pelo preço unitário de venda do produto (p), subtraindo o custo de manutenção por unidade por noite (h) multiplicado pelo indicador I(a_t > 0) (que assume o valor de 1 se a_t > 0 e 0 caso contrário), e subtraindo o custo fixo do pedido (f) acrescido do custo variável do pedido por unidade (v).

Assim, utilizando um conjunto de dados sintéticos, utilizando a DQN, conseguiu obter um lucro de 18,09% a mais que uma política clássica que diz que se o estoque ficar abaixo de um valor s, devemos fazer um pedido S unidades.

Uma dificuldade na modelagem de séries temporais é saber quais pontos do passado são mais úteis para realizar a predição de um ponto futuro, no artigo [2], os autores utilizam RL para tentar resolver esse problema. No trabalho, RL não é aplicado diretamente nas séries temporais, mas sim utilizado para realizar uma busca por hiperparâmetros de uma rede neural que irá melhor se adequar aos dados. O Reinforcement Learning based Dimension and Delay Estimator (RLDDE) é proposto para determinar dois hiperparâmetros: Dimension e Delay. O primeiro está relacionado ao tamanho da entrada da rede neural, enquanto o segundo dita quais pontos da série temporal serão escolhidos como features para realizar a previsão. O algoritmo de RL utilizado é Q-learning, o estado que o agente recebe é determinado pelo desvio padrão que é discretizado em diferentes intervalos e a ação escolhida é formada por dois números representando o Dimension e o Delay. O autor chega em resultados muito bons se comparado com os métodos disponíveis na época da publicação do artigo.

Há uma ampla variedade de métodos disponíveis para lidar com análise de séries temporais, incluindo abordagens baseadas em modelos estatísticos, técnicas de aprendizado de máquina e algoritmos de deep learning. No entanto, a escolha do modelo mais adequado que funcione bem em diferentes horizontes de previsão, ou seja, no curto, médio e longo prazo, pode ser um problema complexo. Assim, os artigos [4] e [5] entram para abordar essa questão.

Diferentes modelos tendem a apresentar melhor desempenho em períodos específicos da série temporal, o que torna a seleção do modelo apropriado uma tarefa complexa. Nesse contexto, a estratégia de Dynamic Ensemble surge como uma abordagem promissora. Essa estratégia envolve a seleção dinâmica dos melhores pesos para a previsão.

Os métodos de Dynamic Ensemble buscam propor políticas adequadas para a previsão de séries temporais probabilísticas, considerando as características de ponderação dos modelos para diferentes horizontes de previsão. Essa abordagem permite uma adaptação flexível, permitindo que os modelos mais apropriados sejam escolhidos de forma dinâmica, dependendo do contexto e das necessidades da previsão. Assim, o objetivo é fazer uma rápida adaptação dos pesos dependendo das características da série temporal, fazendo a pergunta “Qual o melhor modelo para prever o próximo valor da série temporal?”.

Nesse contexto, segundo os artigos, o Reinforcement Learning é utilizado porque consegue explorar melhor o espaço de ações para otimizar a política, além disso abordagens de Model-Free nos permitem aprender uma política puramente a partir dos dados registrados, sem conhecer a complexa dinâmica subjacente do sistema. O modelo utilizado nos artigos [4] e [5] foi um off-policy actor-critic, o DDPG. No artigo [4] ele foi utilizado baseado no Processo de Decisão de Markov, já em [5] ele foi utilizado pelo agente criado que é o RL Based Model Combination (RLMC).

Yu et al., em [3], segue uma linha de pesquisa parecida com os artigos [4] e [5], contudo o objetivo não é selecionar o melhor modelo e sim combinar as previsões feitas por eles. O método proposto é aplicado no campo financeiro para prever o preço das ações da bolsa de valores, um cenário bem complicado devido à alta volatilidade. Este trabalho consiste em estabelecer um pipeline para forecasting robusto, com pré-processamento das séries e integração de modelos de deep learning através do Q-learning. Na parte de pré-processamento, os autores decompõem a séries em vários componentes mais simples através de ferramentas estatísticas a fim de facilitar o aprendizado do modelo. Enquanto que na parte de modelagem, é utilizado 3 modelos de deep learning para fazer a previsão em cada um dos componentes: Deep belief Network (DBN), gated recurrent unit (GRU), long short-term memory network (LSTM).

Nesse cenário, RL é utilizado por meio do Q-learning para melhorar o resultado ao aprender a combinar as previsões dos três modelos de forma eficiente. Assim, o agente recebe como estado os parâmetros ajustados de todos modelos e como ação escolhe os pesos para ponderar cada uma das três previsões feitas em um único valor. Como resultado, o modelo proposto no artigo supera todos os outros métodos testados e atinge bons resultados mesmo nesse cenário desafiador onde há uma grande volatilidade no preço das ações.

Referências

[1] Pavlyshenko, B. M. (2022). Sales time series analytics using deep Q-learning. *arXiv preprint arXiv:2201.02058*.

[2] Liu, F., Quek, C., & Ng, G. S. (2005, July). Neural network model for time series prediction by reinforcement learning. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* (Vol. 2, pp. 809-814). IEEE.

[3] Yu, Y., Lin, Y., Hou, X., & Zhang, X. (2023). Novel optimization approach for realized volatility forecast of stock price index based on deep reinforcement learning model. *Expert Systems with Applications*, 120880

[4]FU, Y.; WU, D.; BOULET, B. Reinforcement Learning Based Dynamic Model Combination for Time Series Forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, *[S. l.]*, v. 36, n. 6, p. 6639-6647, 2022. DOI: 10.1609/aaai.v36i6.20618. Disponível em: https://ojs.aaai.org/index.php/AAAI/article/view/20618. Acesso em: 25 oct. 2023.

[5]Yuhao Ding and Youngsuk Park and Karthick Gopalswamy and Hilaf Hasson and Bernie Wang and Luke. Dynamic ensemble for probabilistic time-series forecasting via deep reinforcement learning, 2023Disponível em: https://www.amazon.science/publications/dynamic-ensemble-for-probabilistic-time-series-forecasting-via-deep-reinforcement-learning. Acesso em: 25 oct. 2023.

[6]XIE, G. Reinforcement Learning for Inventory Optimization Series I: An RL Model for Single Retailers. Disponível em: <https://towardsdatascience.com/a-reinforcement-learning-based-inventory-control-policy-for-retailers-ac35bc592278>.

Organização dos artigos no Drive

Nome	Proprietário	Última modifica...	Tamanho do
Madeka-2022.pdf	eu	8 de nov. de 2023	805 KB
Xie-2022.pdf	eu	26 de out. de 2023	5,8 MB
Yuhao Ding-2023.pdf	eu	26 de out. de 2023	1,6 MB
Yuwei-2022.pdf	eu	26 de out. de 2023	1.008 KB
Pavlyshenko-2022.pdf	luisassuncao@discente.u...		782 KB
Yu-2023.pdf	luisassuncao@discente.u...		4,4 MB
Liu-2005.pdf	luisassuncao@discente.u...		2 MB

APÊNDICE 3

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: **9 de nov. de 2023 12:00**

Participantes da Entrega **[matriculados em Residência em IA]:**

Luís Augusto
Gustavo Barbosa

Entrega: **[descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

No *stage* 3, o time de Reinforcement Learning (RL) decidiu continuar a pesquisa por cenários de aplicação de RL em séries temporais. Depois de mais busca por artigos, foi encontrado aplicações dessas técnicas principalmente nas áreas do varejo e financeira. Na primeira, o contexto de aplicação foi para o gerenciamento de estoque, já no segundo foi aplicado para gerenciamento de portfólio de ações da bolsa de valores. Uma visão geral desses trabalhos podem ser vistos no documento:

https://docs.google.com/document/d/1JmlZR9NR5saIusiZrdv_DkMD1fZsqhfrrbK36W8ca98/edit?usp=sharing

Além disso, como descrito na parte de planejamento do *gate* 2, foi revisado alguns dos algoritmos usados nos artigos lidos até o momento, um breve resumo das principais características deles foi colocado no documento:

https://docs.google.com/document/d/1oa5gWbkHzEq18Cbe2h3_kkjkSQ_5w_6x7zeW0zCrKEE/edit?usp=sharing

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Para próxima semana, foi planejado:

- Estudar mais a fundo as métricas e funções de recompensa comumente utilizadas no cenário do mercado financeiro.
- Buscar por frameworks e repositórios/códigos, e datasets para o varejo e o financeiro

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

LUANA GUEDES BARROS MARTINS: Go!

Revisão dos artigos

Uma das aplicações de *Reinforcement Learning* (RL) com dados temporais é encontrado no mercado financeiro, mais especificamente no trading. Nesse cenário, especialistas precisam observar os padrões passados da cotação dos ativos para tomar decisões assertivas sobre a compra ou venda a fim de gerar lucro. Dessa forma, com RL, uma abordagem para automatização deste processo é treinamento de um agente que aprenda sobre esse padrões do mercado interagindo com o ambiente capaz de tomar tais decisões.

Millea, em [1], realiza uma revisão bibliográfica sobre o uso de *deep Reinforcement Learning* (DRL) para o trading. Neste trabalho, é feito um apanhado geral sobre como DRL permitiu que os agentes conseguissem aprender mesmo de estados complexos e também em ambientes com um espaço de ação grandes. Millea deixa em evidência duas grandes vertentes em que DRL pode ser utilizado, a primeira é o uso de um único ativo da bolsa de valores e com base na cotação passada, o agente toma a decisão de vender, comprar ou segurar. Já na segunda opção, se trata do gerenciamento de um portfólio de ações, onde o agente decide qual será a proporção de cada ativo na carteira do investidor, ou seja, é um espaço de ações contínuo.

Na linha da segunda opção, o artigo [2] propõe uma forma de lidar com problema de gerenciamento de portfólio a partir do histórico de cotação do ativo bem como outras características como liquidez e dividendos pagos. O autor escolheu para compor o ambiente de aprendizagem 415 ações das 500 presentes no índice S&P 500 cotado na bolsa americana. Cada estado era representado por um tensor de dimensão (d, n, f) onde d é quantidade de dias (pontos das séries), n representa o número de ações e f a quantidade de features (cotação, liquidez, dividendo, etc…). O autor decidiu limitar o agente a escolher no máximo 20 ativos para colocar na carteira, sendo assim, o espaço de ações são pesos contínuos de cada ativo cuja soma resulta em 1. O algoritmo utilizado não é especificado, porém é dito que se trata de um *model-free* e *policy gradient*.

O agente foi treinado nos dados de 1998 à 2011, realizou validação de 2012 a 2013 e os testes ficaram para o período de 2014 até 2018. O autor obteve resultados promissores se comparado com outras estratégias de gestão de carteira, contudo, apesar do lucro obtido, é ressaltado que de todas as estratégias, a do agente foi aquela mais arriscada passado por altos e baixos expressivos. Além disso, como nenhuma empresa que faliu foi escolhida para compor a base de treino, é bem possível que o comportamento mais arriscado do agente seja devido a esse viés. Portanto, os resultados de [2] são animadores, mas podem não servir para um investidor mais conservador.

Por outro lado, Yu, P. et al, em [3], foca também na otimização de portfólio, mas a abordagem nesse caso foi *model-based*. O autor utiliza uma modelagem bem mais

elaborada, onde há um modelo de forecasting que é responsável entender a dinâmica do ambiente e realizar a predição do próximo valor da ação. Além disso, no trabalho é ressaltado a insuficiência na quantidade dos dados para treinar o agente, por isso o autor também propõe usar redes neurais generativas adversárias (GAN) para aprender a gerar amostras sintéticas. Por fim, Yu et al também sugere o treinamento e uso de um modelo que imitasse o comportamento de um especialista para gerar experiências adicionais ao agente. No estado do algoritmo é representado pela saída de um rede LSTM aplicada às 10 horas anteriores de todas as 8 ações escolhidas. Nesta representação também é concatenada com a ação anterior juntamente com a previsão do modelo de forecasting. Assim, como em [2], nesse trabalho o espaço de ação é formado pelos pesos que a gente decide dar a cada ativo da carteira. Os resultados pelo agente bateu o baseline estabelecido, além de que o desempenho dele não foi tão volátil como em [2].

Fora do contexto financeiro, há também aplicações de RL e séries temporais no cenário de varejo também como o gerenciamento de estoque. É nesse sentido que o artigo [4] propõe um sistema de controle de inventário, que visa determinar o nível ideal de estoque para diferentes produtos, equilibrando o custo de atender à demanda do cliente com o custo de manter muito estoque. O sistema ajusta o nível de estoque de forma periódica, adquirindo mais estoque ou removendo o estoque existente conforme necessário. Além disso, lida com alguns problemas como a demanda que é uma variável aleatória com dinâmicas desconhecidas(apresentando sazonalidade, tendência, picos), atrasos na chegada de produtos após um pedido e preocupações com a correspondência de preços por concorrentes.

As features utilizadas para representar o estado foram o nível atual de estoque, as ações anteriores tomadas até o tempo t, características relacionadas à série temporal de demanda histórica ajustada, informações sobre a proximidade de feriados públicos, dados históricos de visualizações da página do produto, características estáticas do produto, como grupo de produtos, recursos baseados em texto da descrição do produto, marca, além de informações econômicas do produto, como preço e custo, lags, entre outros. E para ações é utilizado um espaço unidimensional de ações contínuas para cada produto para representar a demanda.

No artigo, é introduzido o algoritmo *DirectBackpropagation*, que é detalhadamente explicado ao longo do texto. Para avaliar seu desempenho em comparação com outros métodos de RL, foram selecionados os métodos *model-free*, A3C, SAC e ARS. Os resultados obtidos revelam que o algoritmo *DirectBackpropagation* mantém níveis mais elevados de recompensa, ao mesmo tempo em que emite um número menor de pedidos em comparação com os outros métodos. Em seguida, o ARS também apresenta um desempenho notável.Além disso, o artigo destaca a capacidade dos algoritmos de RL em gerar curvas de inventário que se assemelham de maneira razoável aos inventários reais.

Referências

[1] Millea, Adrian. "Deep reinforcement learning for trading—A critical survey." *Data* 6.11 (2021): 119.

[2] Huotari, Tommi, Jyrki Savolainen, and Mikael Collan. "Deep reinforcement learning agent for S&P 500 stock selection." *Axioms* 9.4 (2020): 130.

[3] Yu, Pengqian, et al. "Model-based deep reinforcement learning for dynamic portfolio optimization." *arXiv preprint arXiv:1901.08740* (2019).

[4] Madeka, Dhruv, et al. 'Deep Inventory Management'. arXiv [Cs.LG], 2022, http://arxiv.org/abs/2210.03137. arXiv.

Artigo	Dados	Espaço de estados	Espaço de ações	Recompensa	Desempenho
Huotari T., Savolainen J., Collan M. - 2020	S&P 500	Dimensão: (d, n, f). Índices (f) de várias ações (n) a longo do tempo (d).	Contínuo. Peso de cada ação na carteira	Sharpe ratio	328% de retorno total um período do 1255 dias
Yu P. et al - 2019	8 ativos selecionados	saída do LSTM para 10 horas anteriores de 8 ativos+ ação anterior + forecasting do próximo valor da cotação + índice S&P 500	Contínuo. Peso de cada ação na carteira	logarítimo da taxa de retorno	Em 2 anos, obteve 16% de lucro totalizando $ 80,899 dolars
Madeka-2022	Dados semanais de um único Marketeplace, 104 semanas de treinamento, 19 validação	Nível atual de estoque, as ações anteriores tomadas até o tempo t, características relacionadas à série temporal de demanda, informações sobre a proximidade de feriados, dados históricos de visualizações da página do produto, características estáticas do produto, além de informações econômicas do produto, como preço e custo, além de lags	Contínuo. Quantidade de pedido para cada produto	Fluxo de caixa, receita menos temos de custo	Diminuição do estoque em 12% sem perder receita

Estudo dos algoritmos vistos nos artigos

DQN

O Deep Q Learning(DQN) é um método que combina as redes neurais profundas com o algoritmo de Q-Learning, por isso devemos primeiro entender o que é o Q-Learning. Ele é um método baseado em valor, ou seja, é um algoritmo que tenta aproximar a função de valor Q, a qual atribui um valor para cada par de ação e estado. O objetivo é aprender uma função Q ótima, para que assim possa tomar ações que irão maximizar a recompensa acumulativa ao longo do tempo.

Explicação funcionamento Q-Learning:

Inicialização: os valores da função Q são inicializados com zeros ou com valores aleatórios para os pares de estado-ação.

Iteração: O agente interage com o ambiente por tentativa e erro. A cada iteração ele escolhe uma ação com base na política epsilon-greedy, e assim recebe uma recompensa e o próximo estado resultante.

Atualização da função Q: Após cada iteração, o agente atualiza a função Q com a equação de Bellman.

$$Q(s,a) = r(s,a) + \gamma \max_{a} Q(s',a)$$

Onde:

- -"s'" é o próximo estado resultante da ação.
- -"r(s,a)" é a recompensa imediata obtida.
- - "γ" é o fator de desconto que representa a importância das recompensas futuras.

A ideia-chave aqui é que o agente ajusta a estimativa do valor Q para o par de estado-ação com base na recompensa imediata e na estimativa dos valores futuros.

Exploração e Convergência: o agente continua a interagir com o ambiente, utilizando a função Q repetidamente. Com o tempo, a função Q converge para a Q ótima, que atribui valor máximo à melhor ação em cada estado.

Política ótima: A política ótima é então derivada da função Q, escolhendo a ação com o valor Q mais alto em cada estado.

Agora voltando para a DQN, temos que a principal diferença é a rede neural para para aproximar a função de valor, ela recebe o estado e retorna os valores Q para cada ação. Para o treinamento da rede busca-se minimizar a diferença entre os valores Q estimados pela rede online e os valores Q alvo, calculados com base na recompensa imediata e nos valores Q máximos do próximo estado (equação de Bellman). Outra funcionalidade que temos é o replay buffer, porque não iremos fazer o treinamento após cada ação ser tomada, o treinamento será feito em batch, pois ajuda a evitar que o agente fique preso em mínimos locais durante o treinamento.

Links úteis:

- https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html
- https://medium.com/data-science-in-your-pocket/deep-q-networks-dqn-explained-with-examples-and-codes-in-reinforcement-learning-928b97efa792

DDPG

Outro algoritmo que apareceu bastante nos artigos foi o Deep Deterministic Policy Gradient (DDPG), o qual é uma extensão da DQN para espaço de ações que podem assumir valores contínuos. O DDPG pertence à família dos métodos Actor-Critic, que são métodos que utilizam dois principais componentes: o ator(actor) e o crítico(critic). Assim, temos o Ator que

é responsável por aprender a política de forma a maximizar a recompensa esperada ao interagir com o ambiente. E temos o Crítico que estima o desempenho da política no futuro, ao mesmo tempo que ele também aprende as experiências coletadas por tal política, por isso, ele é responsável por avaliar o Ator, ajudando a melhorar sua política.
Assim podemos destacar as seguintes características que o DDPG incorpora:
Ator(política): DDPG usa uma rede neural profunda para representar o ator. A rede aceita o estado como entrada e gera ação como saída. A principal característica é que a política é determinística, ou seja, dada uma entrada de estado, ela mapeia diretamente para uma ação, em vez de produzir uma distribuição de probabilidade sobre as ações.

Crítico: Além da rede do ator, DDPG também utiliza uma rede neural profunda para estimar o valor da ação tomada em um determinado estado (ou crítico). Essa rede recebe o estado e ação realizada pela política, e é treinada para prever o valor esperado (retorno) de estar em um determinado estado e seguir a política determinística do ator. Para a loss da rede crítica, utiliza-se a função de erro Bellman de mínimos quadrados (MSBE), que estima o quão próximo Q* chega de satisfazer a equação de Bellman, conforme mostrado na equação:

$$L(\phi, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}} \left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d) \max_{a'} Q_\phi(s',a') \right) \right)^2 \right]$$

Target Networks: Para tornar o treinamento mais estável, o DDPG utiliza duas redes-alvo, uma para o ator e outra para o crítico. Essas redes alvo são cópias suavizadas das redes principais, assim todos os pesos da rede são suavizados através de uma variável de trade-off.

Exploração com ruído: Para explorar o espaço de ação de maneira eficaz, o DDPG adiciona ruído à ação escolhida pelo ator. Isso ajuda a evitar que o agente fique preso em ótimos locais.

Para problemas com séries temporais o DDPG é interessante porque utiliza um ações contínuas, assim, por exemplo pode ser que em um problema de otimização de estoque podemos querer tomar decisões como a quantidade produtos que precisaremos pedir para poder atender a demanda.

Links úteis:

- https://medium.com/data-science-in-your-pocket/deep-deterministic-policy-gradient-ddpg-explained-with-codes-in-reinforcement-learning-5825fbdc77b2
- https://spinningup.openai.com/en/latest/algorithms/ddpg.html

ARS

ARS (Augmented Random Search) é um algoritmo de aprendizado por reforço (RL) projetado para a otimização de políticas em tarefas de controle contínuo. Ele é conhecido por sua eficácia em ambientes com espaços de ação contínuos e é uma abordagem de gradiente de política baseada em busca aleatória. Ele é uma melhoria do Basic Random Search(BRS), que consiste em escolher uma política parametrizada $\pi\theta$, perturbar os parâmetros θ aplicando $+\nu\delta$ e $-\nu\delta$ (onde $\nu < 1$ é um ruído constante e δ é um número aleatório gerado a partir de uma distribuição normal). Em seguida, aplicar as ações com base em $\pi(\theta+\nu\delta)$ e $\pi(\theta-\nu\delta)$ e coletar as recompensas $r(\theta+\nu\delta)$ e $r(\theta-\nu\delta)$ resultantes dessas ações.Agora que temos as recompensas dos θ perturbados, calculamos a média $\Delta = 1/N * \Sigma[r(\theta+\nu\delta) - r(\theta-\nu\delta)]\delta$ para todos os δ e atualizamos os parâmetros θ usando Δ e uma taxa de aprendizado $\alpha.\theta^{j+1} = \theta^{j} + \alpha.\Delta$.

As diferenças que o ARS traz é que para amenizar as variações do $\alpha.\Delta$ e suavizar mais as atualizações dos parâmetros da política, é feita uma divisão desse valor pelo σ_r(desvio padrão das recompensas coletadas. Outro processo que precisa ser feito é a normalização das features para que todas tenham o mesmo peso.Outro ponto é que como o objetivo é maximizar a recompensa, e para isso a cada iteração fazemos a média das recompensas, e tendo valores pequenos, nossa média cairá muito, por isso utiliza-se a abordagem de utilizar as top-N, ordenadas em ordem decrescente.

Vale citar também que, ao contrário de muitos algoritmos de RL modernos que utilizam redes neurais profundas, o ARS não depende de redes neurais. Em vez disso, ele se concentra em otimizar diretamente os parâmetros da política.

Assim, percebemos que o ARS é um algoritmo bastante eficiente computacionalmente, e que consegue lidar bem com ambientes complexos, o qual é o caso de problemas com séries temporais.

Links úteis:

- https://towardsdatascience.com/introduction-to-augmented-random-search-d8d7b55309bd

APÊNDICE 4

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: **16 de nov. de 2023**

Participantes da Entrega **[matriculados em Residência em IA]:**

Gustavo Barbosa

Entrega: **[descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Para esta entrega, o objetivo principal foi fazer um estudo sobre frameworks de aprendizado por reforço, explorar códigos e repositórios voltados para a otimização de estoques no contexto do varejo. E também fazer uma pesquisa sobre conjuntos de dados relevantes para o meu problema. Dentre os conjuntos de dados considerados, destaco o Brazilian E-Commerce Public Dataset by Olist, que me chamou atenção por ser um conjunto de dados relacionado ao Brasil e pelas features relevantes que ele possui e que foram mencionadas em um dos artigos revisados. Todo esse estudo pode ser visto no documento:
https://docs.google.com/document/d/1c_g_0OwUpCV_Hb5UKZkgaTonSCWs4lNP_P77pl515Ok/edit

Para essa entrega, como um complemento, trouxe também uma visão geral dos baselines, que é uma parte importante, pois que tenho que ter uma forma de comparar a performance do meu agente RL, por isso o tema será melhor aprofundado em uma futura entrega:

Baselines:
https://docs.google.com/document/d/1RIAFFPO92ZDgWFC7yXPbx_gB5rOYsZhkeP3Pxm-j24E/edit?usp=sharing

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Para o próximo stage é planejado:

- Começar a preparação dos dados para o modelo.
- E começar a preparar o ambiente de teste (open-ai gym).

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

CEDRIC LUIZ DE CARVALHO: Go!

LUANA GUEDES BARROS MARTINS: Go!

Frameworks, códigos e datasets

Frameworks:

Gym: utilizado para desenvolvimento e teste de algoritmos de aprendizado por reforço, o gym traz uma padronização dos ambientes, permitindo a criação de ambientes personalizados. Além disso facilita a interação com os ambientes trazendo uma abstração dos detalhes do ambiente, permitindo se concentrar mais na lógica do agente após o ambiente ter sido criado.

Spinning Up: oferece implementações claras e compreensíveis de algoritmos de RL. Embora pareça ter limitações em termos de personalização avançada, é uma escolha sólida, especialmente para aprendizado. Tem a facilidade de se integrar a ambientes Gym. Os tutoriais interativos e a documentação trazem um ambiente propício para o aprendizado prático de RL, tornando o Spinning Up uma opção acessível e informativa.

Rllib: outra biblioteca que traz a implementação de algoritmos de RL, ela destaca-se pela facilidade de ajuste de hiperparâmetros dos algoritmos, além disso, abstrai detalhes de implementação, deixando as tarefas simples e permitindo focar no que realmente importa. Sua capacidade de se integrar aos ambientes do Gym amplia ainda mais sua utilidade, tornando o Rllib uma boa escolha para se obter eficiência e versatilidade em suas implementações de RL.

Implementações dos algoritmos também podem ser encontradas nos links do documento de estudo dos algoritmos que foi entregue no gate 4.

Repositórios/Códigos | Otimização de estoque:

Códigos para ajudar na parte de resolução do problema:

> Link: Implementação DQN para otimização estoque, considerando a demanda do produto.
>
> https://github.com/mariebelle1996/RL_for_inventory_management: RL para otimização de estoque.

https://github.com/Raldoso/Warehouse_RL_Project : O objetivo do programa é gerenciar o estoque de mercadorias no armazém, para prever a quantidade do pedido para t+4 dias, isso considerando lojas dependentes desse estoque.

Datasets

Foi feita uma pesquisa por dados para poder aplicar RL em problemas de séries temporais, levando em consideração o varejo, os dados estão listados a seguir.

1. Store Item Demand Forecasting Challenge:
    - Features: Date, Store, Itens, Sales
    - Quantidade: 913 mil dados de treino, 45 mil dados de teste
    - Link: Store Item Demand Forecasting Challenge

2. Brazilian E-Commerce Public Dataset by Olist:
    - Features: Order_id, product_id, seller_id, order_item_id (quantidade), price, features estatísticas do produto
    - Quantidade: 113 mil dados
    - Link: Brazilian E-Commerce Public Dataset

3. M4-daily-subset:
    - Features: Somente série temporal
    - Quantidade: 4227 dados de treino, 4227 dados de teste
    - Link: M4-daily-subset

4. Retail Data Analytics:
    - Features: Date, store, department, weekly_sales, holiday, temperature, Fuel_Price, Unemployment
    - Quantidade: 422 mil dados
    - Link: Retail Data Analytics

5. Online Retail:
    - Features: Product, quantity, date, price
    - Quantidade: 541,909 dados
    - Link: Online Retail

Destaque para o dataset, Brazilian E-Commerce Public Dataset by Olist, um dataset com vários produtos e que possuem features semelhantes às usadas no artigo [1] para a otimização de estoque.

[1] Madeka, Dhruv, et al. ‘Deep Inventory Management’. arXiv [Cs.LG], 2022, http://arxiv.org/abs/2210.03137. arXiv.

Nome	Features	Quantidade	Link
Store Item Demand Forecasting Challenge	Date, Store, Itens, Sales	913K treino, 45K teste	Link : https://www.kaggle.com/competitions/demand-forecasting-kernels-only/overview
Brazilian E-Commerce Public Dataset by Olist	Order_id, product_id, seller_id, order_item_id(quantidade), price, features estatísticas do produto	113K	www.kaggle.com/datasets/olistbr/brazilian-ecommerce/data
M4-daily-subset	Somente série temporal	4227 treino, 4227 teste	https://www.kaggle.com/datasets/yogesh94/m4-forecasting-competition-dataset/data?select=Daily-test.csv
Retail Data Analytics	Date, store, department, weekly_sales, holiday, temperature, Fuel_Price, Unemployment	422k	https://www.kaggle.com/datasets/manjeetsingh/retaildataset
Online Retail	Product, quantity. date, price	541909	https://archive.ics.uci.edu/dataset/352/online+retail

Baselines

Otimização de Estoque: Baselines

Conforme destacado no artigo [1], propõe-se o uso dos seguintes baselines:

- **Política (R,Q):** Quando o nível do estoque cai abaixo de R unidades, é efetuado um pedido fixo de Q unidades de produtos. Aqui, R é o ponto de reordenação e Q é a quantidade do pedido. Geralmente, o estoque é verificado no início ou no final de cada dia.

- **Política (T,S):** Este método implica em fazer um pedido para repor o estoque até S unidades a cada T dias. T representa o período de revisão, determinando a frequência de revisão do nível de estoque, enquanto S é o nível de pedido.

- **Política (s,S):** Quando o nível do estoque cai abaixo de s unidades, é necessário fazer um pedido para repor o estoque até S unidades. Neste caso, s é considerado como o ponto de reordenação, e S como o nível de pedido.

- **Política de Estoque Base:** Equiparada à política (S-1,S), indicando que um pedido é feito para repor o estoque até S unidades imediatamente se houver qualquer demanda consumindo o estoque em um dia específico.

Outro artigo [2] introduz os seguintes baselines:

- Oracle: é uma política idealizada, o agente irá otimizar a quantidade de pedido sabendo dos futuros valores das variáveis, a ideia é que essa política obtenha os melhores servido de base para ver quão próximo o nosso agente chegará do melhor cenário.
- Diferentes Algoritmos de RL para Otimização de Estoque: Exploração de diversos algoritmos de Reinforcement Learning para aprimorar a gestão de estoque.

O artigo também apresenta outras políticas base, como Newsvendor, Myopic Approximation e Planning Horizon Normalized Policy, porém, essas exigem um estudo mais aprofundado.

Problema de Ensemble em Séries Temporais: Baselines

- **Média/Mediana**: combinação simples de previsões usando a média ou a mediana dos modelos.
- **Pesos Ótimos Encontrados Sem Ensemble Dinâmico**: Determinação de pesos ideais para modelos individuais sem utilizar um ensemble dinâmico.
- **Melhor dos Modelos Base**: Seleção do melhor desempenho entre os modelos individuais.

[1]XIE, G. Reinforcement Learning for Inventory Optimization Series I: An RL Model for Single Retailers. Disponível em: <https://towardsdatascience.com/a-reinforcement-learning-based-inventory-control-policy-for-retailers-ac35bc592278>.

[2] Madeka, Dhruv, et al. 'Deep Inventory Management'. arXiv [Cs.LG], 2022, http://arxiv.org/abs/2210.03137. arXiv.

APÊNDICE 5

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: **23 de nov. de 2023**

Participantes da Entrega **[matriculados em Residência em IA]**:

Gustavo Barbosa

Entrega: **[descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Objetivos do gate: preparar os dados e criar o ambiente de simulação com o OpenAI Gym.

Para a realização do objetivo foi feita toda a preparação dos dados e análise para identificar se os dados eram adequados. Também criei o ambiente onde os agentes irão atuar. O ambiente foi criado com a base de um ambiente já pronto mas com modificações inspiradas na implementação de Madeka et al. (2020, "Deep Inventory Management"). No momento o ambiente funciona com somente um produto para fins de validação, mas a ideia é que ele funcione com múltiplos produtos. A ideia é que a cada época o agente decida quando do produto deve se fazer pedido, e assim otimizar as ações para se obter melhores recompensas.

A preparação dos dados e o ambiente podem ser encontrados em: OtimizaçãoEstoque.ipynb

Uma descrição mais detalhada do ambiente pode ser encontrado no documento: Gate 5

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Como próximos passos para o Stage 6, é planejado:

- Testar algum algoritmo de aprendizado por reforço
- Escolher melhores parâmetros para a função de recompensa
- Melhorar o código da implementação do ambiente

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

LUANA GUEDES BARROS MARTINS: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Ambiente

```
import gym
from gym import spaces
from gym.utils import seeding
import random

class InvOptEnv_unico_produto(gym.Env):
    def __init__(self, max_estoque, estoque_inicial,
custo_pedido,custo_manter, penalidade, custo_fixo, time_serie,
horizonte):

      self.estoque_inicial = estoque_inicial

      self.Co = custo_pedido   # Custo Pedido
      self.Ch = custo_manter    # Custo de manter estoque
      self.Cp = penalidade        # penalidade por ter demanda e não
ter estoque
      self.custo_fixo = custo_fixo

      self.action_space = spaces.Box(low=0, high=max_estoque,
shape=(1,), dtype=int)    # -> ações discretas
      self.observation_space = spaces.Tuple((
          spaces.Discrete(max_estoque+1),    # -> Nível do estoque
          spaces.Discrete(max_estoque+1),    # -> Transição de estoque
          spaces.Discrete(max_estoque+1),    # -> Pedido
          spaces.Discrete(max_estoque+1),     # -> Quantidade para
chegar
          spaces.Discrete(60),                # -> Distância do feriado;
          spaces.Discrete(7),                # -> Dia_semana;
          spaces.Box(low=0, high=max_estoque, shape=(7,),
dtype=np.int32), # -> Histórico de demanda
          ))

      # Initialize the time-series parameters

      self.demand_ts = time_serie

      self.max_estoque = max_estoque    # -> Maximum inventory level
```

```
      self.reset()    # reset the state vector
      self.seed()

      self.horizonte = horizonte

    def seed(self, seed=None):
      self.np_random, seed = seeding.np_random(seed)
      return [seed]

    def reset(self):
        #reset the state vector : state = [Nivel_estoque, Transicao,
Pedido, Pedido_entrega, Distancia_feriado, historico, dia_semana]
        self.Nivel_estoque = self.estoque_inicial
        self.Transicao = 0
        self.Demanda = 0
        self.Pedido = 0
        self.Em_entrega = 0
        self.Distancia_feriado = 0
        self.Dia_semana = 0
        self.historico = [0]*7
        self.state = self._get_observation()

        #reset tempo entrega
        self.Tempo_entrega = 0

        #reset variables used
        self.backlog = 0
        self.produtos_a_mais = 0
        self.lista_em_entrega = []

        #reset period
        self.periodo = 8


        return self._get_observation()

    #Calcula o tempo de entrega
    def _get_tempo_entrega(self):
        return random.randint(5, 10)
```

```
    #obtem a demanda do dia atual
    def _get_demanda(self):
        return self.demand_ts.loc[self.periodo, 'unit_sales']

    #obtem a demanda do dia atual
    def _get_diasferiado(self):
        return self.demand_ts.loc[self.periodo,
'dias_ate_proximo_feriado']

    #obtem a demanda do dia atual
    def _get_dia_da_semana(self):
        return self.demand_ts.loc[self.periodo, 'dia_da_semana']

    #obtem a demanda do dia atual
    def _get_historico(self):
        historico = []
        for i in range(7, 0, -1):
          p = self.periodo - i
          historico.append(self.demand_ts.loc[p, 'unit_sales'])
        return historico

    def _OnOrder_update(self, tempo_entrega, action):
        if len(self.lista_em_entrega) > tempo_entrega:
            self.lista_em_entrega[tempo_entrega] += action
        else:
            self.lista_em_entrega.extend([0]*int(tempo_entrega -
len(self.lista_em_entrega) + 1))
            self.lista_em_entrega[tempo_entrega] += action

    #obtem estado atual
    def _get_observation(self):
        obs = (*(self.Nivel_estoque, self.Transicao, self.Pedido,
self.Em_entrega, self.Distancia_feriado, self.Dia_semana),
*self.historico)
        return np.array(obs).astype(np.int32)

    def _calcular_custo(self):
        if self.Pedido != 0:
            custo_fixo_pedido = 1000
```

```
        else:
            custo_fixo_pedido = 0
        return self.custo_fixo + self.Co*self.Pedido +
custo_fixo_pedido + self.Ch*self.Nivel_estoque + self.Cp*self.backlog +
self.Cp*self.produtos_a_mais

    def step(self, action):
        self.Pedido = action
        self.Tempo_entrega = self._get_tempo_entrega()
        self._OnOrder_update(self.Tempo_entrega, self.Pedido)
        self.Transicao = self.lista_em_entrega.pop(0)
        self.Em_entrega = sum(self.lista_em_entrega)
        self.Demanda = self._get_demanda()
        self.periodo += 1
        self.Distancia_feriado =  self._get_diasferiado()
        self.Dia_semana = self._get_dia_da_semana()
        self.historico =  self._get_historico()
        self.produtos_a_mais = max((self.Nivel_estoque +
self.Transicao) - self.max_estoque,0)
        self.backlog = max(self.Demanda - min(self.Nivel_estoque +
self.Transicao, self.max_estoque), 0)
        self.Nivel_estoque = max(min(self.Nivel_estoque +
self.Transicao, self.max_estoque) - self.Demanda, 0)
        reward = -1*self._calcular_custo()
        done = self.periodo == self.horizonte
        self.state = self._get_observation()

        return self.state, reward, done, {}
```

Descrição do Ambiente

Gate 5

Para se adequar ao meu problema, adaptei códigos já existentes de ambientes em um ambiente OpenAI Gym customizado. Os códigos de base para a criação podem ser encontrados no documento Frameworks, códigos e datasets - gate4 do gate 4. Além disso, fiz a preparação dos dados, unindo tabelas diferentes, formatando os dados e tratando valores nulos. O código se encontra em:

https://colab.research.google.com/drive/12Epcp48zyJagVUP_Yas8l569qefOUtQe?usp=sharing

Sobre o Ambiente

O ambiente foi montado para simular um estoque para um produto. Nesse ambiente é possível realizar uma ação de quantos produtos pedir no dia tendo como base o estado recebido da ação anterior. O pedido tem tempo de entrega aleatório e assumimos que ao chegar ele fica disponível antes que ocorra a demanda do dia para fins de cálculo de recompensa. Ao chegar o produto já é adicionado ao estoque e o novo nível de estoque é calculado, considerando também a demanda. Há também uma variável de *backlog* para calcular se houve demanda e quantos vendas foram perdidas. O ambiente é finalizado quando se atinge um horizonte específico de dias.

$$back\log \;=\; \max\left(I_t - d_t,\; 0\right)$$

Ações e estados

Ação a_t discreta assumindo valores de 0 até o nível máximo de estoque, representando quanto deve ser pedido daquele produto.

Já os estados são representados pelas seguintes variáveis:

- Nível de estoque dia anterior (I_{t-1})
- Transição de estoque (quanto produtos chegaram no dia t_t)
- Pedido (última ação tomada)
- Pedido em entrega (x_t)
- Distância do feriado (df_t)
- Dia da semana (ds_t)
- Histórico de demanda de n dias anteriores(d_t representa a demanda)

Assim temos que o estado no tempo t é:

$$s_t \;=\; \left(I_{t-1},\; t_t\,,\; a_{t-1},\; x_t\,,\; df_t,\; ds_t\,,\; \left[d_{t-n},\; \ldots\,,\; d_{t-1}\right]\right)$$

Recompensa

A função de recompensa a cada época de decisão mede os custos do estoque, há um custo por por fazer um pedido, existem custos para manter estoque e também são penalizadas perdas de vendas. Assim, a função de recompensa é:

$$R_t = custofixo \ + \ custo_{pedido} \cdot a_t \ + \ custo_{estoque} \cdot I_t + custo_{perda} \cdot back\log$$

APÊNDICE 6

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: **30 de nov. de 2023**

Participantes da Entrega **[matriculados em Residência em IA]**:

Gustavo Barbosa

Entrega: **[descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Como planejado, no *stage* 6 fiz testes com algoritmos de aprendizado por reforço no meu ambiente. Testei os algoritmos Deep Q Network (DQN) e Deep Deterministic Policy Gradient (DDPG) com implementações que são encontradas facilmente em repositórios da internet. Também comecei a tentar rodar os agentes Proximal Policy Optimization (PPO) e Augmented Random Search (ARS), mas devido a necessidade de estudo mais aprofundado das bibliotecas em que esses algoritmos estão implementados, não consegui fazer os testes nesse *stage*.

Resultados podem ser encontrados no documento: Stage 6

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- **Testar os algoritmo ARS e PPO**
- **Implementar baselines básicos e fazer comparação**

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

Neste gate, o Professor Aldo André Díaz Salazar esteve na banca avaliadora substituindo a Professora Luana.

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

LUANA GUEDES BARROS MARTINS: **Em análise!**

Resultados dos testes

Stage 6

Nesse *stage* me concentrei em testar agentes no meu ambiente,tendo conseguido testar os agentes Deep Q Network (DQN) e Deep Deterministic Policy Gradient (DDPG), com implementações que são encontradas facilmente em repositórios da internet. Comecei a tentar rodar os agentes Proximal Policy Optimization (PPO) e Augmented Random Search (ARS), mas devido a necessidade de estudo mais aprofundado das bibliotecas em que esses algoritmos estão implementados, não consegui fazer os testes.

Código estão no github: https://github.com/Gustavobrg/Residencia

Dados

Os dados utilizados para treinamento são de demanda diária de um único produto, assim como features extraídas da data, que são distância do feriado e dia da semana, a demanda pode ser observados na Figura 1:

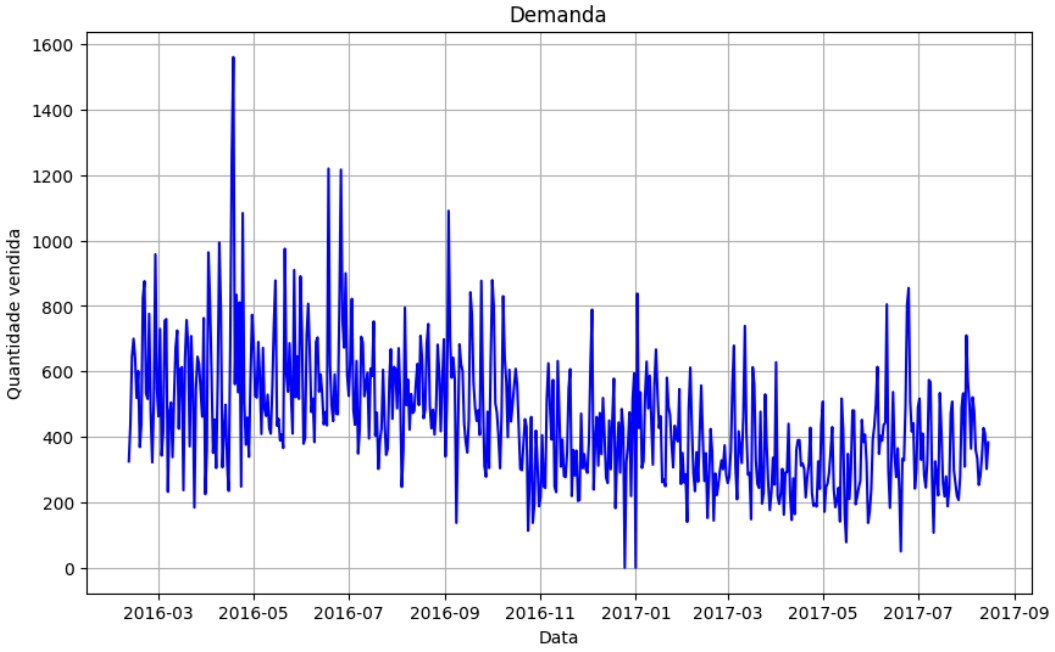


Figura 1: Demanda

Para o treinamento a cada época a gente interage com o ambiente por 400 dias.

Problema

Existem dois componentes principais em uma abordagem de aprendizado por reforço: o agente e o ambiente. A interação entre o agente e o ambiente de estoque é mostrada na Figura 2:

1. O agente observa o estado atual do ambiente.
2. Com base no estado atual, o agente toma uma ação de acordo com sua política.
3. O ambiente avança um passo com base na ação e na transição de estado, ou seja, uma transição, e então gera uma recompensa.
4. O agente recebe a recompensa e utiliza as transições para atualizar sua política.

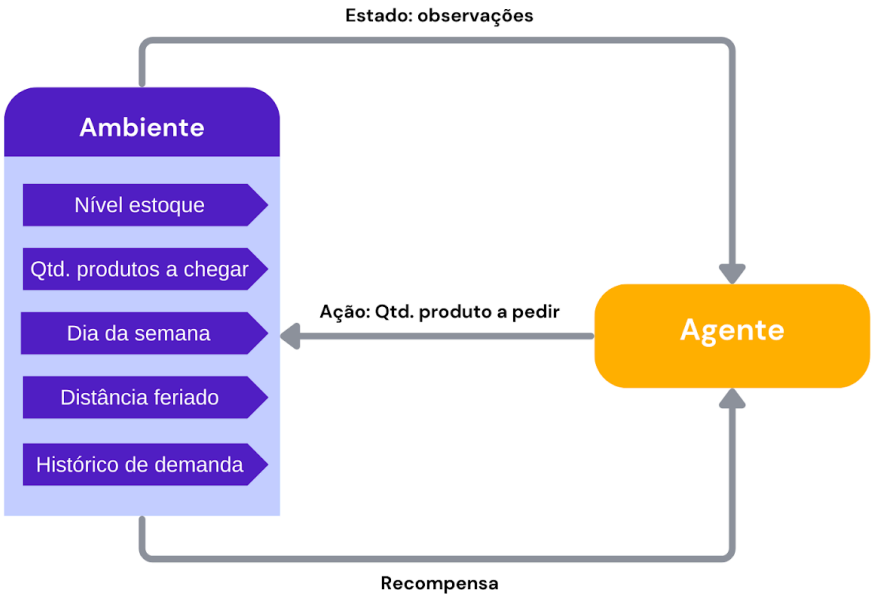


Figura 2: Problema RL

Função de recompensa

A função de recompensa a cada época de decisão mede os custos do estoque, há um custo por por fazer um pedido, existem custos para manter estoque e também são penalizadas perdas de vendas. Assim, a função de recompensa é:

$$R_t = custo fixo \; + \; custo_{pedido} \cdot a_t \; + \; custo_{estoque} \cdot I_t + custo_{perda} \cdot back\log$$

- a_t: Pedido
- I_t: Estoque

- backlog: diferença entre estoque e demanda, se o estoque for menor que a demanda

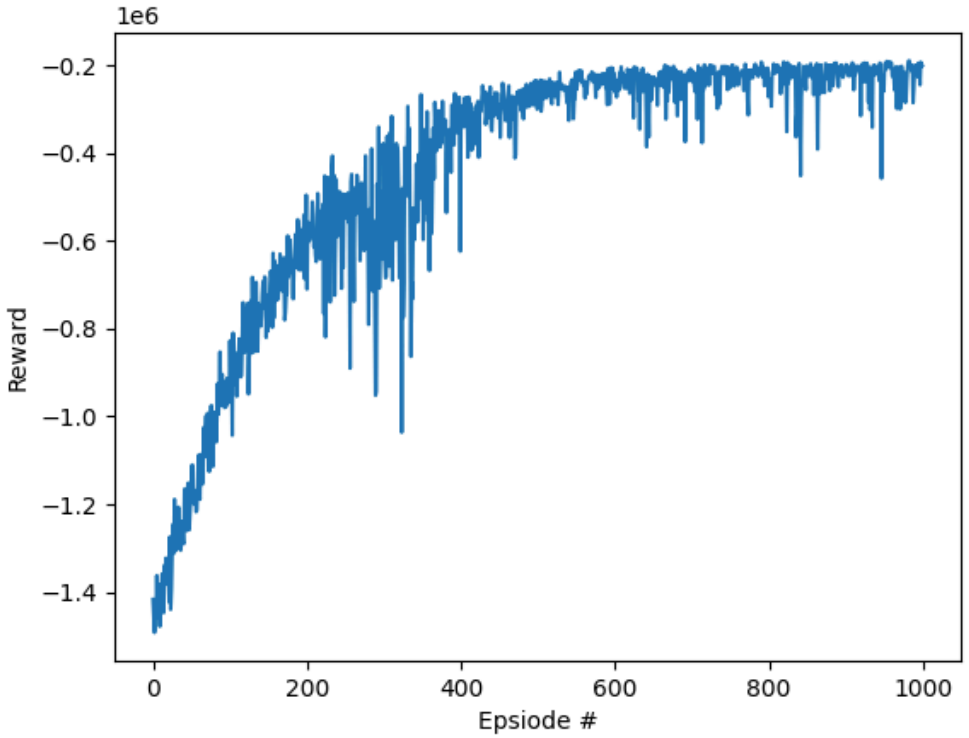
Ademais, foi adicionado um novo custo caso o pedido seja diferente de 0.

Resultado

Agente DQN

Para o treinamento usei um número determinado de ações que o agente pode tomar. Na primeira tentativa o agente convergiu para ótimas ações, entretanto, o agente fazia pedidos todos os dias que representava a média diária, e como não era algo que eu queria, fiz um novo teste. Nesse, resolvi mudar a função de recompensa e penalizar quanto é feito um pedido, o que se mostrou uma solução eficaz, já que o agente começou a zerar os pedidos e fazer pedidos maiores ocasionalmente já que pedidos individuais custam mais caro.

A imagem mostra a convergência do algoritmo:



Agente DDPG

Como o DQN só toma ações discretas limitadas, resolvi testar o DDPG, já que ele é feito para espaço de ações contínuas . Entretanto, o agente não conseguiu convergir com o ambiente, tomando somente ações 0. Testei algumas abordagens, como alterar parâmetros do DDPG, como tamanho da rede, ruído de exploração, tamanho da memória, também teste

mudar os espaço de ações para o intervalo de 0 e 1 e fazer escala, mudar a função de recompensa, mas nada com sucesso. Apesar de tudo ainda não desisti dele, visto que ele dá resultados em problemas semelhantes ao meu.

APÊNDICE 7

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: **7 de dez. de 2023**

Participantes da Entrega **[matriculados em Residência em IA]**:

Gustavo Barbosa

Entrega: **[descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Continuando o meu estudo Reinforcement Learning para otimização de estoque, trouxe para essa semana o treinamento de novos algoritmos, o PPO e o ARS, melhorias na função de recompensa e o uso de um novo dataset, e um baseline para comparação de resultados. No arquivo Stage 7 É possível encontrar a descrição das minhas atividades e os resultados obtidos com todos os meus modelos treinados. Também dentro do arquivo há o acesso ao repositório no github e a explicação de como reproduzir os experimentos.

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Agora que eu tenho resultados mais sólidos, pretendo avançar o meu projeto para o próximo estágio que é o controle de estoque de mais de um produto, para isso farei o estudo de como fazer isso tendo como base artigos já vistos.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

Neste gate, o Professor Aldo André Díaz Salazar esteve na banca avaliadora substituindo a Professora Luana.

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

LUANA GUEDES BARROS MARTINS: **Em análise!**

Resultados: Semana 7

Atualizações das semanas anteriores:

- Função de recompensa: Para considerar não somente os custos, mas também os ganhos alterei a função de recompensa para somar o dinheiro ganho com as vendas, assim a nova função de recompensa é definida como o preço de vendas vezes a demanda total atendida, menos o custo total associado ao estoque e pedidos:
  Rt = p*min(Dt, It-1) - (custoestoque*It-1 + custopedido*at + custofixo_pedido)

- Dataset: como os dados que eu estava utilizando tinha uma demanda muito uniforme, os modelos de Reinforcement Learning não traziam melhora nos resultados, por isso, resolvi utilizar o dataset da competição M5 Forecasting: https://www.kaggle.com/competitions/m5-forecasting-accuracy/data

Descrição das atividades feitas:

Fiz o treinamento de novos modelos de RL que são o ARS e o PPO, e implementei um baseline que chamei de “sS”, conforme visto no artigo Reinforcement Learning for Inventory Optimization Series I: An RL Model for Single Retailers que diz que quando o nível do estoque cai abaixo de s unidades, é necessário fazer um pedido para repor o estoque de S unidades. Neste caso, s é considerado como o ponto de reordenação, e S como o nível de pedido.

Os modelos que eu tinha antes eram implementações com várias linhas de código, mas para esse novos modelos resolvi utilizar uma biblioteca chamada Stable Baselines, já que ela unifica a estrutura dos algoritmos e permite uma codificação mais limpa. Assim, uma parte desse stage foi dedicada ao estudo, entendimento e aplicação dessa biblioteca. Também fiz algumas alterações no ambiente para se encaixar no modelo requerido pela biblioteca, mas nenhuma alteração afetou a lógica do ambiente.

Após o estudo dos algoritmos fiz o treinamento de todos e fiz teste em um período de 100 dias que não estava presente no conjunto de treinamento.

Resultados:

Receita: comparei os resultados com a soma acumulativa da recompensa durante os 100 dias (Figura 1), os resultados foram que o ARS e DDPG alcançaram os melhores resultados que o baseline, enquanto os modelos DQN e PPO não conseguiram se adaptar bem ao problema. Para que o PPO alcance melhores resultados farei um melhor ajuste dos hiperparâmetros para que ele se adapte melhor ao meu problema.

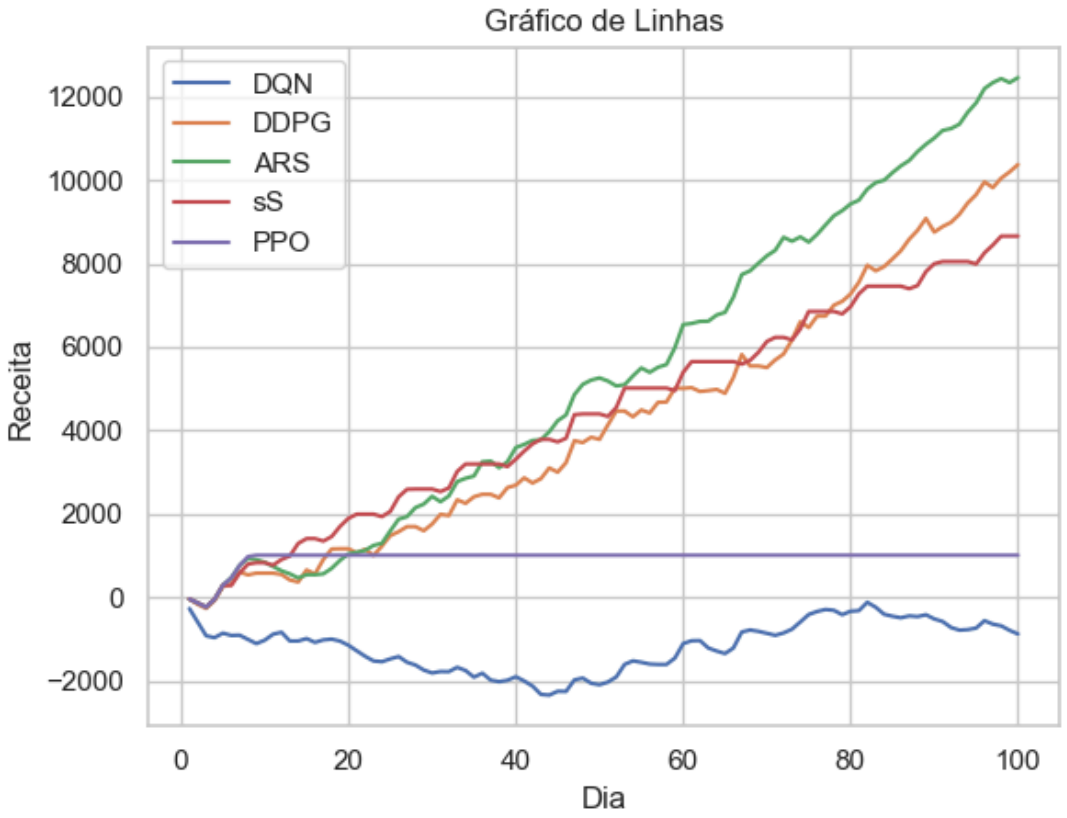


Figura 1

Comparando valores o ARS alcançou uma receita 54% maior que o baseline e o DDPG alcançou uma receita 29% maior, entretanto esse valor pode variar um pouco devido a dinâmica aleatória de chegada de produtos.

O código para treinamento dos modelos está disponível no repositório: https://github.com/Gustavobrg/Residencia. É possível treinar os modelos com seus respectivos códigos de treinamento, e acessar os modelos na pasta Agentes. A comparação entre os modelos pode ser feita com o arquivo benchmark.py.

APÊNDICE 8

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 14 de dez. de 2023

Participantes da Entrega [matriculados em Residência em IA]:

Gustavo Barbosa

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

O objetivo da *stage* era estudar como ampliar um problema de aprendizado por reforço para lidar com múltiplos produtos. Duas abordagens foram consideradas: multi-objetivo e multi-agente. A abordagem multi-objetivo foi descartada porque os objetivos do problema são complementares, não conflitantes e com objetivo único. A abordagem multi-agente também foi descartada porque os produtos são independentes uns dos outros. A abordagem mais eficaz foi a de compartilhar os parâmetros da política entre todos os produtos, retornando na função de recompensa simplesmente a soma das recompensas individuais dos produtos. O novo ambiente foi implementado no repositório https://github.com/Gustavobrg/Residencia.git com o nome env_v2.py. No entanto, ainda não foi possível realizar testes com algoritmos de RL porque a política do ambiente está retornando ações NaN, talvez sendo necessário criar uma política customizada para resolver esse problema.
A entrega está contida no documento Stage 8.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Para a próxima *gate* irei explorar os algoritmos novos, "A3C" (Asynchronous Advantage Actor-Critic) e "SAC" (Soft Actor-Critic), assim como explorar mais os que eu já tinha testado (ARS e PPO). Planejo também incorporar maior customização ao desenvolver políticas personalizadas para o meu problema.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

LUANA GUEDES BARROS MARTINS: Go!

Pesquisa: Semana 8

Para essa *stage* fiquei de fazer um estudo de como ampliar o meu problema para que eu possa lidar com múltiplos produtos, há duas abordagens: Multi-objetivo e Multi-Agente. O termo "multiobjetivo" é uma extensão do aprendizado por reforço (RL) que refere-se a situações em que um agente de RL deve otimizar simultaneamente vários objetivos. Em vez de lidar com uma única recompensa ou objetivo, o agente enfrenta uma combinação de objetivos, muitas vezes conflitantes, que precisam ser equilibrados. No meu problema, cada objetivo pode ser associado a uma função de recompensa do produto. Essas funções de recompensa indicam ao agente como ele está se saindo em relação a cada produto. E para o meu problema não é uma técnica relevante porque ele é utilizada mais frequentemente quando se tem objetivos conflitantes e no meu caso, os objetivos não seriam conflitantes, seria somente um objetivo, outro ponto a se considerar é que o meu problema pode ser resumido a um único objetivo que é maximizar o lucro vindo de todos os produtos.

Outra abordagem é o aprendizado por reforço com múltiplos agentes, refere-se a um contexto em que vários agentes autônomos interagem em um ambiente comum e aprendem a melhorar seu desempenho através de recompensas .No aprendizado por reforço multi-agente, cada agente tem seus próprios objetivos, ações e observações, e eles interagem uns com os outros, influenciando o ambiente compartilhado. É para o meu problema isso a cooperação dos agentes para atingir o objetivo não seria muito relevante, porque os produtos são independentes uns dos outros, não haveria interação. Outra dificuldade dessa abordagem é ter que treinar muitos agentes.

A abordagem mais eficaz é explicada no artigo [1], nela os parâmetros da política são compartilhados entre todos os produtos. E a recompensa é simplesmente a soma da recompensa individual dos produtos, isso é uma abordagem eficaz porque eu quero maximizar o meu lucro total, e não produtos únicos.

Assim, modifiquei meu ambiente para que ele agora aceite múltiplos produtos, assim, o estado tem a forma (*num_produtos*, *num_features*), e a ação a forma (*num_produtos*). A função de recompensa retorna um valor que é a soma de da recompensa de cada produto. Como é possível encontrar mais soluções para problemas de trading de ações, me baseei no repositório: https://github.com/Akhilesh-Gogikar/MultiStockRLTrading/blob/main/multi_stock_trading_env.py. O novo ambiente está disponível no github https://github.com/Gustavobrg/Residencia.git com o nome *env_v2.py*. Ainda não consegui fazer testes com os algoritmos de RL porque a política dele está retornando ações NaN, por isso, creio que terei que fazer uma política customizada para receber o estado e retornar as ações corretamente.

Descrição Geral do Problema: O meu problema envolve tomar decisões sobre o estoque de produtos ao longo do tempo.Irei focar em um produto específico para explicar o processo, mas as decisões podem ser tomadas para todos os produtos juntos.

Processos Externos:

- Demanda: A cada passo de tempo, há uma demanda para o produto.
- Preço de Venda e Custo de Compra : Os preços de venda e custos de compra também são variáveis.
- Tempo de entrega: Descreve o tempo de entrega de estoque nos próximos períodos após fazer um pedido ao fornecedor.

Assim, o estado do produto em um momento específico é representado por um vetor que contém todas essas informações.

Política de Controle:

O processo de controle envolve tomar ações para cada produto, representadas pelas quantidades de pedido, escolhidas a partir de um conjunto de todas as possíveis ações.

Processo de Controle:

As ações são escolhidas por um processo de controle em conjunto para todos os produtos.
As políticas descrevem como as ações são escolhidas com base nas features.
Em resumo, o problema envolve tomar decisões sobre a quantidade de produtos a serem pedidos ao longo do tempo, considerando a incerteza na demanda e tempo de entrega. O objetivo é desenvolver políticas eficientes para otimizar o desempenho do sistema de estoque.

[1] Madeka, Dhruv, et al. ‘Deep Inventory Management’. arXiv [Cs.LG], 2022, http://arxiv.org/abs/2210.03137. arXiv.

APÊNDICE 9

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: **21 de dez. de 2023**

Participantes da Entrega **[matriculados em Residência em IA]**:

Gustavo Barbosa

Entrega: **[descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Para este estágio, concentrei meu estudo na análise dos algoritmos e seus parâmetros. Os algoritmos incluem o Proximal Policy Optimization (PPO), Augmented Random Search (ARS), Soft Actor-Critic (SAC) , Asynchronous Advantage Actor-Critic (A3C) e Deep Deterministic Policy Gradient (DDPG). Utilizando o framework Stable Baselines 3 como base, conduzi uma análise abrangente sobre como diferentes hiperparâmetros e configurações afetam o desempenho desses algoritmos em diversos ambientes.

Este estudo aprofundará nossa compreensão sobre o funcionamento de cada algoritmo e como os parâmetros impactam o desempenho e a eficiência, visando aprimorar os resultados em contextos diversos e maximizar os benefícios do uso do framework Stable Baselines 3.

O estudo está no documento Gate 9.

A primeira parte do trabalho aborda o estudo dos algoritmos, enquanto a segunda parte consiste em uma análise detalhada dos parâmetros dessas técnicas.

No documento há um texto destacado em amarelo que é o estudo que eu tinha feito no stage 3.

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Para a próxima *gate* irei seguir para a segunda parte do meu planejamento feito anteriormente, que é explorar os algoritmos novos, "A3C" (Asynchronous Advantage Actor-Critic) e "SAC" (Soft Actor-Critic), assim como explorar mais os que eu já tinha testado (ARS, PPO e DDPG). Para isso, irei sair da teoria e trazer resultados práticos/comparações de resultados. Planejo também incorporar maior customização ao desenvolver políticas personalizadas para o meu problema.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

LUANA GUEDES BARROS MARTINS: Go!

Estudo dos algoritmos e seus parâmetros

Introdução

Para este estágio, concentrei meu estudo na análise dos algoritmos e seus parâmetros. Os algoritmos escolhidos incluem o Proximal Policy Optimization (PPO) [1], Augmented Random Search (ARS) [2], Soft Actor-Critic (SAC) [3], Asynchronous Advantage Actor-Critic (A3C) [4] e Deep Deterministic Policy Gradient (DDPG) [5]. Utilizando o framework Stable Baselines 3, uma biblioteca conceituada que oferece implementações robustas desses algoritmos (o que facilitou entendimento da implementação dessas técnicas), conduzi uma análise abrangente sobre como diferentes hiperparâmetros e configurações afetam o desempenho desses algoritmos em diversos ambientes.

Este estudo aprofundará nossa compreensão sobre o funcionamento de cada algoritmo e como os parâmetros impactam o desempenho e a eficiência, visando aprimorar os resultados em contextos diversos e maximizar os benefícios do uso do framework Stable Baselines 3. Com esse conhecimento, nossa intenção é avançar na aplicação prática do aprendizado por reforço na área de otimização de estoque.

A primeira parte do trabalho aborda o estudo dos algoritmos, enquanto a segunda parte consiste em uma análise detalhada dos parâmetros dessas técnicas.

1. Estudo dos algoritmos

PPO

Proximal Policy Optimization (PPO) é um algoritmo de aprendizado por reforço utilizado para treinar redes neurais profundas com o objetivo de otimizar políticas. Ele aborda várias dificuldades associadas aos métodos de gradiente de política e visa melhorá-los introduzindo maneiras mais eficientes de atualizar políticas. Aqui está um resumo abrangente do PPO, seus processos de funcionamento e suas características:

Visão Geral do PPO:

O PPO é um tipo de algoritmo de gradiente de política que utiliza uma família de funções objetivas para atualizar a política de uma maneira que garanta mudanças modestas entre iterações consecutivas da política. O PPO foi projetado para ter os benefícios da Otimização de Política de Região de Confiança (TRPO), sendo ao mesmo tempo mais simples de implementar, mais geral e demonstrando uma melhor complexidade de amostragem.

Como o PPO Funciona:

O PPO funciona alternando entre a coleta de dados por meio da interação com o ambiente e a otimização de uma função objetiva substituta usando ascensão estocástica de gradiente.

1. **Coleta de Dados:** Interações com o ambiente são realizadas para coletar dados na forma de transições de estado, recompensas e ações, de maneira semelhante aos métodos típicos de gradiente de política.

2. **Função Objetiva Substituta**: Em vez de simplesmente usar os retornos observados para atualizar o gradiente de política, o PPO modifica o objetivo padrão do gradiente de política com uma função que considera a razão de probabilidade das ações sob as políticas atual e anterior.

3. **Razões de Probabilidade Clipadas:** O PPO propõe uma nova função objetiva que usa razões de probabilidade clipadas, formando uma estimativa pessimista do desempenho da política.

A Função Objetiva Clipada:

No cerne da abordagem do PPO está uma função matemática chamada função objetiva clipada. Ela é projetada para manter as atualizações da política estáveis e impedir que a política mude demais, o que pode degradar o desempenho do agente de aprendizado por reforço.

A função objetiva clipada é definida como:

$$Ł^{CLIP}(\theta) = E_t\left[\min(r_t(\theta) \cdot A_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon) \cdot A_t)\right]$$

Onde:
- $Ł^{CLIP}(\theta)$ é o objetivo clipado.
- $E_t[\cdot]$ denota o valor esperado ao longo de um lote finito de amostras. $E_t[\cdot]$ denota o valor esperado ao longo de um lote finito de amostras.
- $r_t(\theta)$ é a razão de probabilidade da ação em t sob a nova política π_θ em comparação com a antiga política $\pi_{\theta_{\text{old}}}$.
- A_t é um estimador da vantagem no tempo t.
- ε (épsilon) é um hiperparâmetro, geralmente um valor pequeno como 0,1 ou 0,2, que define a faixa de clipagem.

A função funciona tomando o mínimo entre os objetivos não clipado e clipado, tornando o objetivo final uma estimativa pessimista do objetivo não clipado. Esse mecanismo de clipagem limita as atualizações a uma faixa confiável para evitar atualizações de política grandes e potencialmente prejudiciais.

Características do PPO:

1. **Simplicidade:** O PPO visa oferecer uma alternativa mais simples ao TRPO, com um processo de otimização de primeira ordem que não envolve operações complexas, como o cálculo de gradiente conjugado ou Hessiano.
2. **Generalidade:** Pode funcionar em uma ampla gama de problemas de aprendizado por reforço.
4. **Complexidade de Amostragem:** Empiricamente, o PPO mostrou ser eficiente em termos da quantidade de interação ambiental necessária para alcançar bom desempenho, em comparação com alguns outros métodos.
5. **Flexibilidade:** O PPO pode ser adaptado para usar segmentos de trajetória de comprimento fixo e pode ser combinado com um termo de erro de função de valor e um bônus de entropia para promover a exploração.
6. **Desempenho:** Em experimentos, o PPO mostrou superar muitos outros métodos de gradiente de política em tarefas de referência, incluindo controle robótico simulado e jogar jogos Atari.

O equilíbrio entre simplicidade, robustez e eficiência fez do PPO um algoritmo popular na comunidade de aprendizado por reforço, especialmente no controle de sistemas com espaços de ação contínuos

SAC

O algoritmo Soft Actor-Critic (SAC) é um método de aprendizado profundo por reforço off-policy, model-free desenvolvido para abordar as limitações dos algoritmos anteriores de aprendizado profundo por reforço, como a ineficiência amostral já que a cada atualização de política deve se fazer uma nova amostragem, o que não é necessário nos métodos off-policy já que eles conseguem aprender com experiências passadas com o replay buffer. E outro ponto de melhoria é na alta sensibilidade a hiperparâmetros.

Características do SAC:

Aprendizado Off-Policy: O SAC é um método off-policy, o que significa que ele pode utilizar experiências passadas para aprender. Essa capacidade permite uma eficiência amostral maior, pois as experiências podem ser armazenadas em um buffer de repetição e reutilizadas para várias atualizações.

Model-Free: Sendo sem modelo, o SAC não precisa de um modelo do ambiente (como prever o próximo estado dado o estado e a ação atuais). Isso torna o SAC aplicável a uma ampla variedade de problemas, incluindo casos em que é difícil obter um modelo do ambiente.

Controle Contínuo: O SAC é projetado para ambientes com espaços de estado e ação contínuos, comuns em robótica e outras aplicações do mundo real.

Aprendizado de Reforço com Entropia Máxima: O SAC incorpora um termo de maximização de entropia na função de recompensa para incentivar a exploração no aprendizado de políticas. Isso leva a políticas mais robustas que performam bem diante de mudanças no ambiente e incertezas.

Política Estocástica: A política aprendida pelo SAC é estocástica, o que significa que a ação tomada é amostrada de uma distribuição de probabilidade (como uma Gaussiana) em vez de ser determinística.

Estabilidade e Convergência: O design do SAC promove estabilidade durante o treinamento e convergência para uma política ótima em comparação com outros métodos fora de política, que frequentemente enfrentam problemas de estabilidade e convergência em domínios de ação contínua.

Como o SAC Funciona: A principal inovação do SAC é a integração da maximização de entropia dentro do framework de aprendizado por reforço, equilibrando exploração e exploitação.

Iteração da Soft Policy:

O SAC opera com base no princípio da iteração da Soft Policy, que consiste em duas etapas principais: avaliação e melhoria da soft policy.

Avaliação de Política Suave: Estima o valor dos estados sob a política atual, considerando tanto recompensas quanto entropia (aleatoriedade das ações). Os valores Q são atualizados com base em uma forma da equação de Bellman acrescida de um termo de entropia.

Melhoria da Soft Policy: Atualiza a política para aumentar a recompensa esperada e a entropia da distribuição de ações. Isso pode ser feito por meio de uma técnica de reparametrização que permite retropropagação tanto através da função Q quanto da rede de políticas.

A entropia:

A entropia no contexto do aprendizado por reforço é uma medida da imprevisibilidade e aleatoriedade nas ações da política. Maximizar a entropia é desejável porque incentiva a política a explorar uma diversidade de ações, o que pode evitar a convergência prematura para soluções subótimas, melhorar a robustez às mudanças ou imprecisões no modelo e ajudar a descobrir múltiplas estratégias que podem levar a recompensas igualmente elevadas. Em essência, ao manter uma entropia mais alta, a política pode equilibrar de forma mais eficaz entre explorar novas possibilidades e explorar comportamentos conhecidos que oferecem recompensas.

Aprendizado com Redes Neurais:

Aproximadores de função, geralmente redes neurais profundas, representam tanto a política (ator) quanto a função Q (crítico). O SAC mantém dois conjuntos de críticos (funções Q) para reduzir o viés de superestimação na estimativa de valor.

Redes Alvo:

Para estabilizar o aprendizado, o SAC usa redes alvo, que representam versões retardadas dos críticos que são atualizadas lentamente ao longo do tempo. Essas redes alvo ajudam a mitigar mudanças rápidas nos valores que podem desestabilizar o processo de otimização.

Truque de Reparametrização:

O SAC aplica o truque de reparametrização para atualizações de política. Isso envolve amostrar ações de uma versão transformada de uma distribuição de ruído (por exemplo, ruído gaussiano por meio de uma rede neural), o qual é diferenciável e permite uma estimativa de gradiente mais estável e precisa.

Atualizações de Ator e Crítico:

Os parâmetros das redes de ator e crítico são atualizados por meio de descida de gradiente estocástica. O crítico é atualizado para minimizar o erro de Bellman, e o ator é otimizado para aumentar as recompensas esperadas, incentivando também uma entropia mais alta na distribuição de ações.

Em resumo, a integração do aprendizado off-policy com exploração baseada em entropia no SAC abordou desafios-chave no aprendizado profundo por reforço, oferecendo desempenho de ponta com eficiência amostral aprimorada e robustez para problemas de controle contínuo.

A2C

Advantage Actor-Critic (A2C) é um algoritmo popular no campo de Aprendizado por Reforço Profundo (DRL) que busca combinar as vantagens de métodos policy-based e value-based. Abaixo, há uma explicação detalhada do A2C, como ele funciona e suas características:

O que é o A2C?

A2C significa Advantage Actor Critic. Este é um tipo de método Actor-Critic em que o "Ator" refere-se ao componente responsável por selecionar ações (ou seja, a política), e o "Crítico" avalia as ações tomadas estimando seu valor.

Como o A2C Funciona

O algoritmo A2C envolve duas principais aproximações de função, geralmente representadas por redes neurais:

Ator: Esta é uma função de política parametrizada por theta (θ), que recebe o estado atual do ambiente como entrada e produz uma distribuição de probabilidade sobre as possíveis ações a serem tomadas. O Ator é responsável por decidir qual ação realizar.

Crítico: Esta é uma função de valor parametrizada por pesos (w), que estima o retorno esperado (valor) de estar em um determinado estado e tomar uma ação de acordo com a política do Ator. O Crítico avalia a qualidade da ação escolhida aproximando o valor Q (valor de estado-ação) ou o valor V (valor de estado).

Processo de Treinamento

- Em cada timestep, o estado atual (S_t) é inserido tanto no Ator quanto no Crítico.
- O Ator gera uma ação (A_t) com base na política.
- O Crítico processa tanto o estado (S_t) quanto a ação (A_t) para calcular o valor Q, o qual é uma estimativa das recompensas esperadas para a ação tomada no estado dado.
- Quando a ação é realizada no ambiente, resulta em um novo estado (S_t+1) e uma recompensa (R_t+1).
- Usando essa experiência, o Ator atualiza seus parâmetros de política, aprendendo a selecionar ações melhores.
- O Crítico também atualiza seus parâmetros de valor para melhorar as avaliações futuras da qualidade da ação.

Características do A2C

Redução de Variância: O Crítico ajuda a estabilizar o treinamento fornecendo uma linha de base que os retornos da política do Ator podem ser comparados. Isso reduz a variância que surge do uso apenas da amostragem de Monte Carlo, prevalente em métodos de gradiente de política como o REINFORCE.

Função de Vantagem: Para reduzir ainda mais a variância, o Crítico pode usar uma função de vantagem em vez da estimativa direta do valor Q. A função de vantagem mede o quão melhor uma ação é em comparação com a ação média no estado. Isso é geralmente calculado subtraindo o valor do estado (V(s)) do valor Q (Q(s,a)), resultando em A(s,a) = Q(s,a) - V(s), assim o valor A(s,a) guia a direção do gradiente. Embora o cálculo real de uma função de vantagem exigisse conhecer tanto V(s) quanto Q(s,a), na prática, ela pode ser aproximada pelo erro temporal de diferença (TD), que é a diferença entre a recompensa obtida ao realizar a ação e a recompensa esperada.

Estabilidade: Métodos Actor-Critic como o A2C tendem a ser mais estáveis do que métodos que dependem apenas de gradientes de política ou funções de valor isoladamente. Isso ocorre porque a avaliação do Crítico fornece um sinal orientador para o Ator, reduzindo a variância nas atualizações e ajudando a evitar mudanças de política grandes e potencialmente destrutivas.

Melhoria Contínua: O A2C permite o ajuste contínuo tanto da política do Ator quanto da estimativa de valor do Crítico, levando a uma aprendizagem mais robusta ao longo do tempo.

O A2C é um algoritmo robusto que melhora a estabilidade de métodos de gradiente de política convencionais incorporando um Crítico baseado em valor junto ao Ator baseado em política. Ao usar a função de vantagem e o erro TD para atualizações de política, ele alcança uma variância reduzida e treinamento mais eficiente. O A2C é particularmente útil para problemas nos quais tanto a estabilidade das atualizações de política quanto a capacidade de avaliar ações são cruciais para o sucesso, como em tarefas complexas de controle e simulações robóticas.

NOTA: Aqui está o estudo feito no stage 3

DQN

O Deep Q Learning(DQN) é um método que combina as redes neurais profundas com o algoritmo de Q-Learning, por isso devemos primeiro entender o que é o Q-Learning. Ele é um método baseado em valor, ou seja, é um algoritmo que tenta aproximar a função de valor Q, a qual atribui um valor para cada par de ação e estado. O objetivo é aprender uma função Q ótima, para que assim possa tomar ações que irão maximizar a recompensa acumulativa ao longo do tempo.

Explicação funcionamento Q-Learning:

Inicialização: os valores da função Q são inicializados com zeros ou com valores aleatórios para os pares de estado-ação.

Iteração: O agente interage com o ambiente por tentativa e erro. A cada iteração ele escolhe uma ação com base na política epsilon-greedy, e assim recebe uma recompensa e o próximo estado resultante.

Atualização da função Q: Após cada iteração, o agente atualiza a função Q com a equação de Bellman.

$$Q(s,a) = r(s,a) + \gamma \max_{a} Q(s',a)$$

Onde:

- -”s'” é o próximo estado resultante da ação.
- -”r(s,a)” é a recompensa imediata obtida.
- - “γ” é o fator de desconto que representa a importância das recompensas futuras.

A ideia-chave aqui é que o agente ajusta a estimativa do valor Q para o par de estado-ação com base na recompensa imediata e na estimativa dos valores futuros.

Exploração e Convergência: o agente continua a interagir com o ambiente, utilizando a função Q repetidamente. Com o tempo, a função Q converge para a Q ótima, que atribui valor máximo à melhor ação em cada estado.

Política ótima: A política ótima é então derivada da função Q, escolhendo a ação com o valor Q mais alto em cada estado.

Agora voltando para a DQN, temos que a principal diferença é a rede neural para para aproximar a função de valor, ela recebe o estado e retorna os valores Q para cada ação. Para o treinamento da rede busca-se minimizar a diferença entre os valores Q estimados pela rede online e os valores Q alvo, calculados com base na recompensa imediata e nos valores Q máximos do próximo estado (equação de Bellman). Outra funcionalidade que temos é o replay buffer, porque não iremos fazer o treinamento após cada ação ser tomada, o treinamento será feito em batch, pois ajuda a evitar que o agente fique preso em mínimos locais durante o treinamento.

DDPG

Outro algoritmo que apareceu bastante nos artigos foi o Deep Deterministic Policy Gradient (DDPG), o qual é uma extensão da DQN para espaço de ações que podem assumir valores contínuos. O DDPG pertence à família dos métodos Actor-Critic, que são métodos que utilizam dois principais componentes: o ator(actor) e o crítico(critic). Assim, temos o Ator que é responsável por aprender a política de forma a maximizar a recompensa esperada ao interagir com o ambiente. E temos o Crítico que estima o desempenho da política no futuro, ao mesmo tempo que ele também aprende as experiências coletadas por tal política, por isso, ele é responsável por avaliar o Ator, ajudando a melhorar sua política.
Assim podemos destacar as seguintes características que o DDPG incorpora:
Ator(política): DDPG usa uma rede neural profunda para representar o ator. A rede aceita o estado como entrada e gera ação como saída. A principal característica é que a política é determinística, ou seja, dada uma entrada de estado, ela mapeia diretamente para uma ação, em vez de produzir uma distribuição de probabilidade sobre as ações.

Crítico: Além da rede do ator, DDPG também utiliza uma rede neural profunda para estimar o valor da ação tomada em um determinado estado (ou crítico). Essa rede recebe o estado e ação realizada pela política, e é treinada para prever o valor esperado (retorno) de estar em um determinado estado e seguir a política determinística do ator. Para a loss da rede crítica,

utiliza-se a função de erro Bellman de mínimos quadrados (MSBE), que estima o quão próximo Q* chega de satisfazer a equação de Bellman, conforme mostrado na equação:

$$L(\phi, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}} \left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d) \max_{a'} Q_\phi(s',a') \right) \right)^2 \right]$$

Target Networks: Para tornar o treinamento mais estável, o DDPG utiliza duas redes-alvo, uma para o ator e outra para o crítico. Essas redes alvo são cópias suavizadas das redes principais, assim todos os pesos da rede são suavizados através de uma variável de trade-off.

Exploração com ruído: Para explorar o espaço de ação de maneira eficaz, o DDPG adiciona ruído à ação escolhida pelo ator. Isso ajuda a evitar que o agente fique preso em ótimos locais.

Para problemas com séries temporais o DDPG é interessante porque utiliza um ações contínuas, assim, por exemplo pode ser que em um problema de otimização de estoque podemos querer tomar decisões como a quantidade produtos que precisaremos pedir para poder atender a demanda.

ARS

ARS (Augmented Random Search) é um algoritmo de aprendizado por reforço (RL) projetado para a otimização de políticas em tarefas de controle contínuo. Ele é conhecido por sua eficácia em ambientes com espaços de ação contínuos e é uma abordagem de gradiente de política baseada em busca aleatória. Ele é uma melhoria do Basic Random Search(BRS), que consiste em escolher uma política parametrizada $\pi\theta$, perturbar os parâmetros θ aplicando $+\nu\delta$ e $-\nu\delta$ (onde $\nu < 1$ é um ruído constante e δ é um número aleatório gerado a partir de uma distribuição normal). Em seguida, aplicar as ações com base em $\pi(\theta+\nu\delta)$ e $\pi(\theta-\nu\delta)$ e coletar as recompensas $r(\theta+\nu\delta)$ e $r(\theta-\nu\delta)$ resultantes dessas ações.Agora que temos as recompensas dos θ perturbados, calculamos a média $\Delta = 1/N * \Sigma[r(\theta+\nu\delta) - r(\theta-\nu\delta)]\delta$ para todos os δ e atualizamos os parâmetros θ usando Δ e uma taxa de aprendizado $\alpha.\theta^{j+1} = \theta^j + \alpha.\Delta$.

As diferenças que o ARS traz é que para amenizar as variações do $\alpha.\Delta$ e suavizar mais as atualizações dos parâmetros da política, é feita uma divisão desse valor pelo σ_r(desvio padrão das recompensas coletadas. Outro processo que precisa ser feito é a normalização das features para que todas tenham o mesmo peso.Outro ponto é que como o objetivo é maximizar a recompensa, e para isso a cada iteração fazemos a média das recompensas, e tendo valores pequenos, nossa média cairá muito, por isso utiliza-se a abordagem de utilizar as top-N, ordenadas em ordem decrescente.

Vale citar também que, ao contrário de muitos algoritmos de RL modernos que utilizam redes neurais profundas, o ARS não depende de redes neurais. Em vez disso, ele se concentra em otimizar diretamente os parâmetros da política.

Assim, percebemos que o ARS é um algoritmo bastante eficiente computacionalmente, e que consegue lidar bem com ambientes complexos, o qual é o caso de problemas com séries temporais.

2. Estudo dos parâmetros dos algoritmos

PPO

Taxa de Aprendizagem: Se for muito alta, o treinamento pode divergir; se for muito baixa, o treinamento será muito lento. Um valor de partida comum é 3e-4, mas isso deve ser adaptado com base em resultados empíricos. Taxas de aprendizagem adaptativas, como as do otimizador Adam, podem ajudar a gerenciar isso automaticamente. No *stablebaselines3* altera-se o parâmetro *learning_rate*.

Fator de Desconto (γ): Geralmente, está próximo a 1 (por exemplo, 0.99). Determina o peso das recompensas futuras. Quanto mais próximo esse valor estiver de 1, mais o agente considerará as recompensas futuras. No *stablebaselines3* altera-se o parâmetro *gamma.*

Lambda (λ) da Estimativa de Vantagem Generalizada (GAE): Usado na Estimativa de Vantagem Generalizada. Ele equilibra o viés e a variância na função de vantagem. Valores são tipicamente entre 0.9 e 1.0. No *stablebaselines3* altera-se o parâmetro *gae_lambda.*

Clip Range: Parâmetro que determina quanto as atualizações de política podem diferir do comportamento da política antiga. Ajuda a evitar grandes atualizações e manter a estabilidade do treinamento. Valores iniciais comuns estão entre 0.1 e 0.3. O intervalo de corte pode ser constante ou pode decair durante o treinamento. No *stablebaselines3* pode se definir o *clip range* para as atualizações na política e na função de valor alterando respectivamente os valores *clip_range* e *clip_range_vf.*

Número de passos por Atualização: Isso dita quantos passos de interação com o ambiente devem acontecer antes de uma atualização. Números menores levam a atualizações mais frequentes, o que pode melhorar a estabilidade da aprendizagem, mas reduz a eficiência da amostra. No *stablebaselines3* altera-se o parâmetro *n_steps.*

Tamanho do Mini Batch: Isso influencia tanto a eficiência da amostra do algoritmo quanto a estabilidade das atualizações. Muito pequeno, e a variância das estimativas pode ser muito alta; muito grande, e as atualizações podem ser muito infrequentes. No *stablebaselines3* altera-se o parâmetro *batch_size.*

Número de Épocas: O número de passagens sobre os dados coletados a serem realizados por atualização. Muitas épocas podem levar ao excesso de ajuste aos dados amostrados, enquanto poucas podem resultar em sub ajuste. No *stablebaselines3* altera-se o parâmetro *n_epochs.*

Coeficiente de Entropia: Isso controla o quanto a política é incentivada a explorar versus 'exploitar'. Um valor mais alto promove a exploração, enquanto um valor mais baixo encoraja a exploração. No *stablebaselines3* altera-se o parâmetro *ent_coef.*

Coeficiente da Função de Valor: Coeficiente da função de valor na perda total, equilibrando o quanto o erro de predição de valor contribui para a atualização da rede. No *stablebaselines3* altera-se o parâmetro *vf_coef.*

Norma Máxima do Gradiente: Geralmente, um valor de corte do gradiente é usado para prevenir atualizações excessivamente grandes que poderiam desestabilizar a aprendizagem. Valores comuns variam de 0.5 a 1.0. No *stablebaselines3* altera-se o parâmetro *max_grad_norm.*

ARS

Taxa de Aprendizado (η): Se for muito alta, o treinamento pode divergir; se for muito baixa, o treinamento será muito lento. Um valor de partida comum é 2e-2 para esse problema, mas isso deve ser adaptado com base em resultados empíricos. Taxas de aprendizagem adaptativas, como as do otimizador Adam, podem ajudar a gerenciar isso automaticamente. No *stablebaselines3* altera-se o parâmetro *learning_rate*.

Amostras de Direção (N): É o número de perturbações dos parâmetros da política a serem avaliadas em cada iteração. Mais amostras fornecerão uma melhor estimativa do gradiente, mas implicam um aumento da carga computacional. É necessário balancear o número de amostras com os recursos computacionais disponíveis e a complexidade da tarefa. No *stablebaselines3* altera-se o parâmetro *n_delta*.

Melhores Direções (b): Das amostras de direção N, o ARS normalmente considera apenas as b direções de melhor desempenho quando atualiza os parâmetros da política. Isso funciona como uma forma de redução seletiva de ruído e foca o aprendizado nas direções mais promissoras. Geralmente, é definido como uma fração de N. No *stablebaselines3* altera-se o parâmetro *n_top*.

Ruído (σ): O desvio padrão do ruído adicionado para criar as perturbações. O nível de ruído deve ser suficientemente alto para explorar o espaço de parâmetros, mas não tão alto que ofusque o sinal subjacente do que constitui um bom desempenho. No *stablebaselines3* altera-se o parâmetro *delta_std*.

Número de episódios de avaliação: Número de episódios para avaliar cada candidato durante o processo de atualização da política. No *stablebaselines3* altera-se o parâmetro *n_eval_episodes*.

SAC

Fator de Desconto (γ): Geralmente, está próximo a 1 (por exemplo, 0.99). Determina o peso das recompensas futuras. Quanto mais próximo esse valor estiver de 1, mais o agente considerará as recompensas futuras. No *stablebaselines3* altera-se o parâmetro *gamma.*

Taxa de Aprendizado: Taxa de aprendizagem para o otimizador adam, no *stablebaseline3* a mesma taxa de aprendizagem será usada para todas as redes (Q-Values, Ator e função Value). Altera-se no Stable Baselines 3 com o argumento *learning_rate*.

Tamanho do Buffer: Define a capacidade do buffer de replay, que armazena experiências passadas do agente. No Stable Baselines 3, ajusta-se com o parâmetro *buffer_size*.

Batch Size: Determina o tamanho do mini batch utilizado para cada atualização de gradiente. No Stable Basiles 3, modifica-se com o parâmetro *batch_size*.

Coeficiente de Atualização Suave (tau): O parâmetro tau é utilizado no SAC para a atualização suave das redes alvo (target networks), em particular as redes que estimam os valores Q-alvo (Q-target). Em algoritmos de aprendizado por reforço que utilizam redes alvo, estas redes fornecem estimativas estáveis para os Q-values ao longo do tempo, ajudando a estabilizar o aprendizado. Valores menores para tau vão resultar em atualizações mais lentas das redes alvo e podem promover um treinamento mais estável, especialmente em ambientes com recompensas e dinâmicas mais complexas ou ruidosas. No entanto, valores muito baixos podem fazer com que a rede alvo demore demais para incorporar melhorias da rede online, o que pode retardar o progresso no aprendizado. Por outro lado, um valor maior para TAU incentiva que as redes alvo se ajustem mais rapidamente, o que pode ser útil em ambientes mais simples ou quando se deseja acelerar o processo inicial de aprendizado. No entanto, isto pode potencialmente levar a uma maior instabilidade no treinamento, já que as estimativas de Q-target estarão mudando mais rapidamente. Portanto, ao selecionar o valor para tau, é essencial considerar o balanço entre a estabilidade do processo de aprendizado e a capacidade do modelo em adaptar-se rapidamente a melhorias na política. Configura-se no Stable Baselines 3 com o argumento *tau*.

Coeficiente de Entropia: Promove a diversidade nas ações escolhidas pelo agente, incentivando a exploração. No Stable Baselines 3, isso pode ser automático ou definido manualmente com o argumento *ent_coef*.

Frequência de Treinamento: Específica com que frequência o modelo é atualizado com relação ao número de etapas de ambiente percorridas. No Stable Baselines 3, ajusta-se com o parâmetro *train_freq.*

A2C

A2C (Advantage Actor Critic) foi implementado como uma alternativa ao A3C (Asynchronous Advantage Actor Critic) para fornecer uma implementação síncrona e determinística que espera cada ator terminar seu segmento de experiência antes de realizar uma atualização, fazendo a média sobre todos os atores [4]. Esse método permite o uso mais eficaz de GPUs, que funcionam melhor com tamanhos maiores de lotes, em contraste com a otimização do A3C para threads de CPU e atualizações assíncronas

Taxa de aprendizado: Se for muito alta, o treinamento pode divergir; se for muito baixa, o treinamento será muito lento. Um valor de partida comum é 7e-4 para esse algoritmo, mas isso deve ser adaptado com base em resultados empíricos. Taxas de aprendizagem adaptativas, como as do otimizador Adam, podem ajudar a gerenciar isso automaticamente. No *stablebaselines3* altera-se o parâmetro *learning_rate*.

Fator de desconto (γ): Geralmente, está próximo a 1 (por exemplo, 0.99). Determina o peso das recompensas futuras. Quanto mais próximo esse valor estiver de 1, mais o agente considerará as recompensas futuras. No *stablebaselines3* altera-se o parâmetro *gamma.*

Coeficiente de Entropia: Este parâmetro incentiva a exploração ao impedir a convergência prematura para políticas subótimas. Ajustar a entropia pode ajudar a garantir a exploração suficiente do espaço de estados. No Stable Baselines 3, isso pode ser automático ou definido manualmente com o argumento *ent_coef*.

Coeficiente de perda do valor: Isso determina a importância da perda da função de valor na função de perda total. Equilibrar a magnitude da perda da política e da perda do valor pode melhorar a estabilidade de aprendizado. No *stablebaselines3* altera-se o parâmetro *vf_coef.*

Norma Máxima do Gradiente: Geralmente, um valor de corte do gradiente é usado para prevenir atualizações excessivamente grandes que poderiam desestabilizar a aprendizagem. Valores comuns variam de 0.5 a 1.0. No *stablebaselines3* altera-se o parâmetro *max_grad_norm.*

DDPG

Taxa de Aprendizado: Taxa de aprendizagem para o otimizador adam, no *stablebaseline3* a mesma taxa de aprendizagem será usada para todas as redes (Q-Values, Ator e função Value). Altera-se no Stable Baselines 3 com o argumento *learning_rate*.

Tamanho do Buffer: Define a capacidade do buffer de replay, que armazena experiências passadas do agente. No Stable Baselines 3, ajusta-se com o parâmetro *buffer_size*.

Batch Size: Determina o tamanho do mini batch utilizado para cada atualização de gradiente. No Stable Basiles 3, modifica-se com o parâmetro *batch_size*.

Coeficiente de Atualização Suave (tau): Este parâmetro controla a taxa na qual as redes alvo são atualizadas em direção às redes locais, garantindo a estabilidade da aprendizagem. Configura-se no Stable Baselines 3 com o argumento *tau*.

Fator de Desconto (γ): Geralmente, está próximo a 1 (por exemplo, 0.99). Determina o peso das recompensas futuras. Quanto mais próximo esse valor estiver de 1, mais o agente considerará as recompensas futuras. No *stablebaselines3* altera-se o parâmetro *gamma.*

Ruído de ação: O DDPG incorpora ruído à saída da ação para exploração; escolher o tipo e o nível corretos de ruído é importante para uma exploração eficiente. No *stablebaselines3* altera-se o parâmetro *action_noise.*

Referências:

[1] SCHULMAN, J. et al. Proximal Policy Optimization Algorithms. 2017.

[2] MANIA, H.; GUY, A.; RECHT, B. Simple random search provides a competitive approach to reinforcement learning. 2018.

[3] HAARNOJA, T. et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. 2018.

[4] OpenAI Baselines: ACKTR & A2C. Disponível em: <https://openai.com/research/openai-baselines-acktr-a2c>. Acesso em: 20 dez. 2023.

[5] LILLICRAP, T. P. et al. Continuous control with deep reinforcement learning. 2015.

APÊNDICE 10

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 11 de jan. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Gustavo Barbosa

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Para essa semana o objetivo era trazer os testes com os algoritmos Proximal Policy Optimization (PPO), Augmented Random Search (ARS), Soft Actor-Critic (SAC) e Deep Deterministic Policy Gradient (DDPG), os resultados e métodos utilizados podem ser encontrados no arquivo Stage 10.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

LUANA GUEDES BARROS MARTINS: Go!

Resultados: Semana 10

Resultados

Foram conduzidos testes com 100 demandas distintas, selecionadas para avaliar o desempenho dos agentes de aprendizado por reforço em diversos contextos. O objetivo principal é maximizar a recompensa acumulada ao longo de todos os passos.

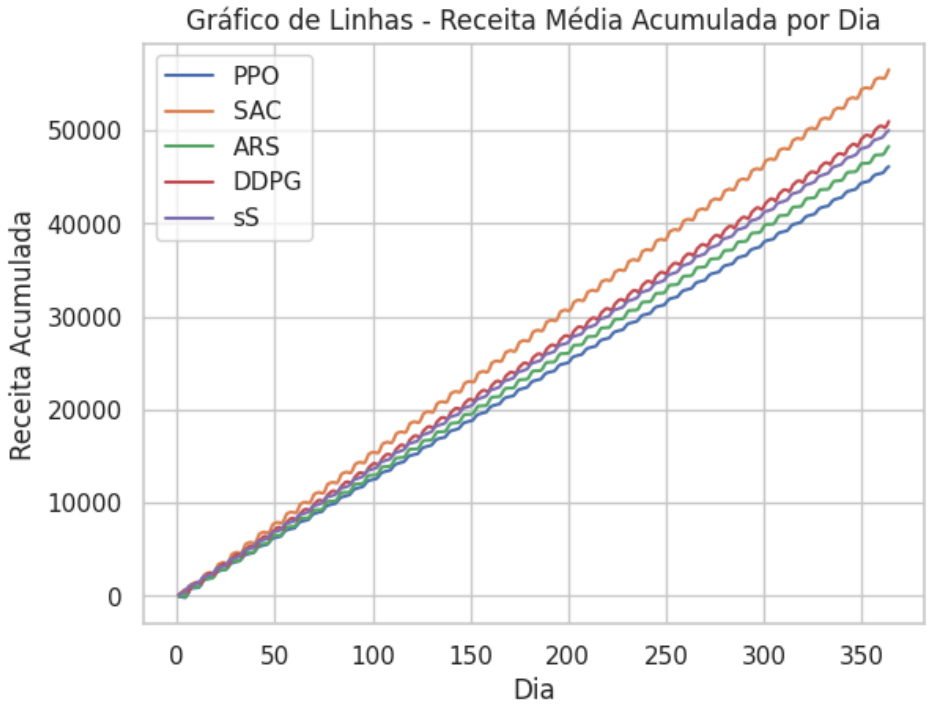


Figura 1
Receita média calculada dia a dia durante 356 dias.

Na Figura 1, são comparados os algoritmos Proximal Policy Optimization (PPO), Augmented Random Search (ARS), Soft Actor-Critic (SAC) e Deep Deterministic Policy Gradient (DDPG), juntamente com um Baseline denominado 'sS'. O Algoritmo Asynchronous Advantage Actor-Critic (A3C) não apresentou resultados satisfatórios e foi excluído da análise. O SAC obteve a maior recompensa acumulada, apesar de possuir uma média de recompensa inferior aos demais algoritmos (conforme mostrado na Figura 2), atingindo um lucro 12% maior que o baseline.

Ao analisar as ações executadas pelos algoritmos (Figura 3), observa-se que SAC, DDPG e a política sS adotam uma ampla variedade de ações. Em contraste, ARS e PPO escolhem uma ação e a mantêm ao longo do intervalo de tempo completo.

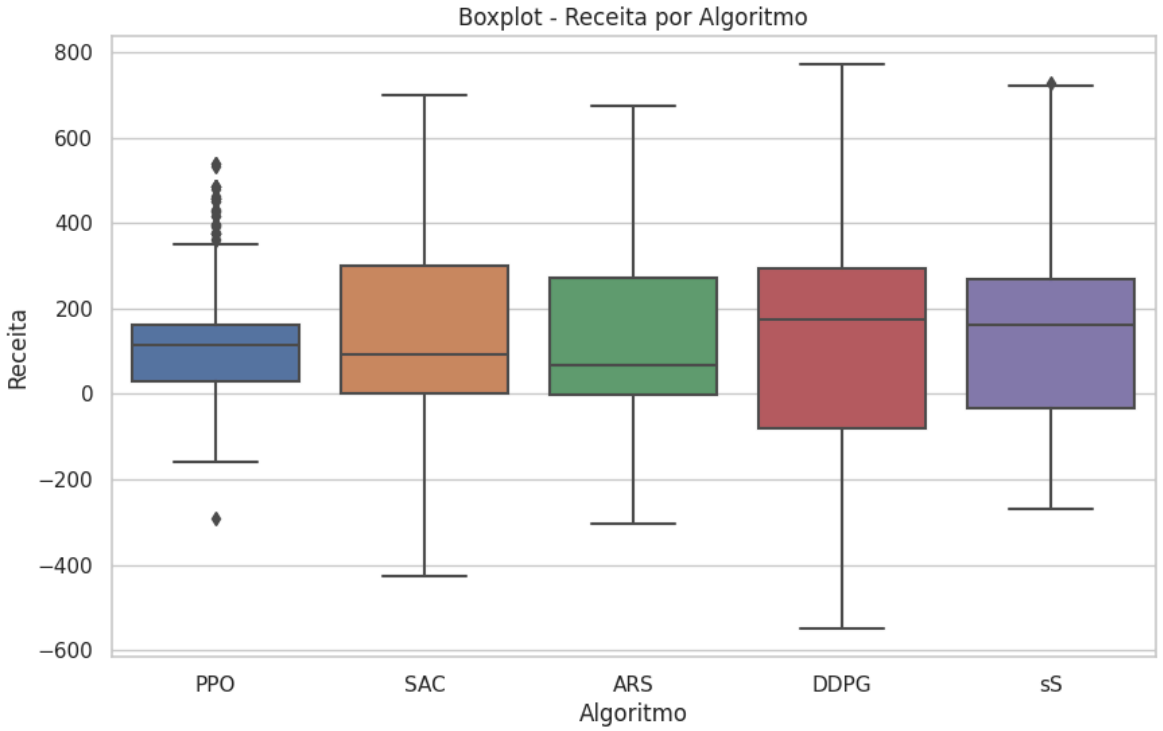


Figura 2
Boxplot receita por algoritmo

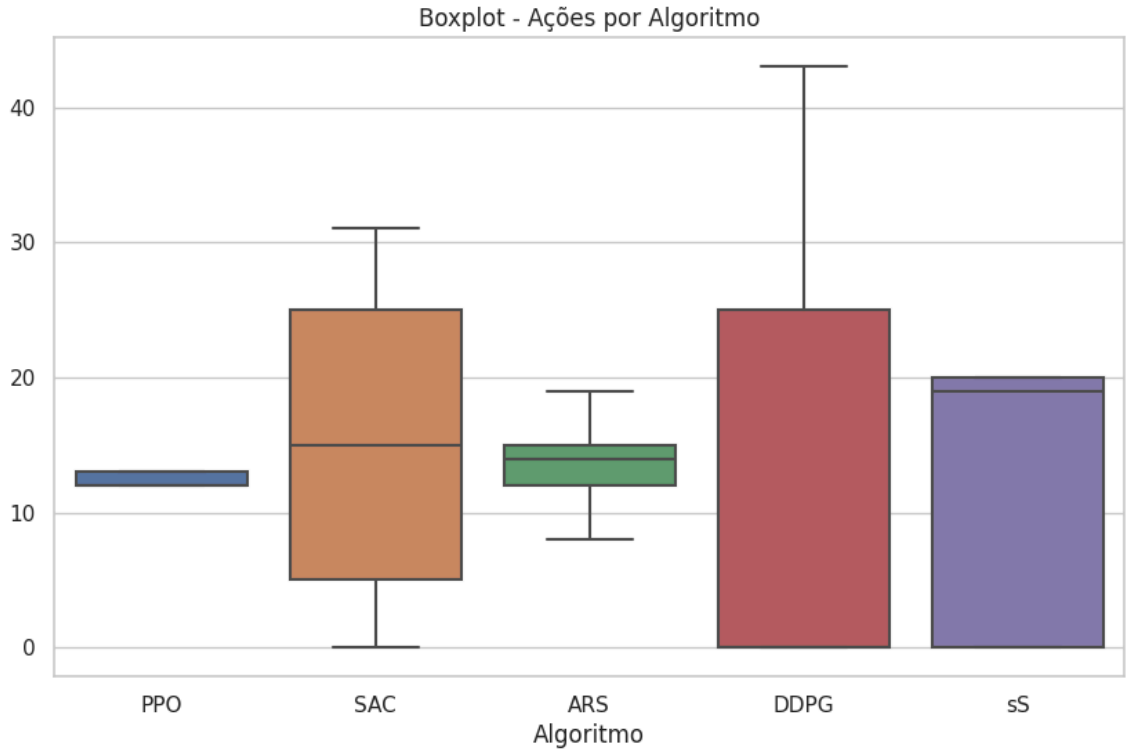


Figura 3
Boxplot Ações por algoritmo

A análise do estoque ao longo dos dias (Figura 4) revela que os algoritmos menos eficazes mantêm um estoque consistentemente baixo. Em contraste, o SAC consegue manter um nível de estoque equilibrado, nem muito alto nem muito baixo.

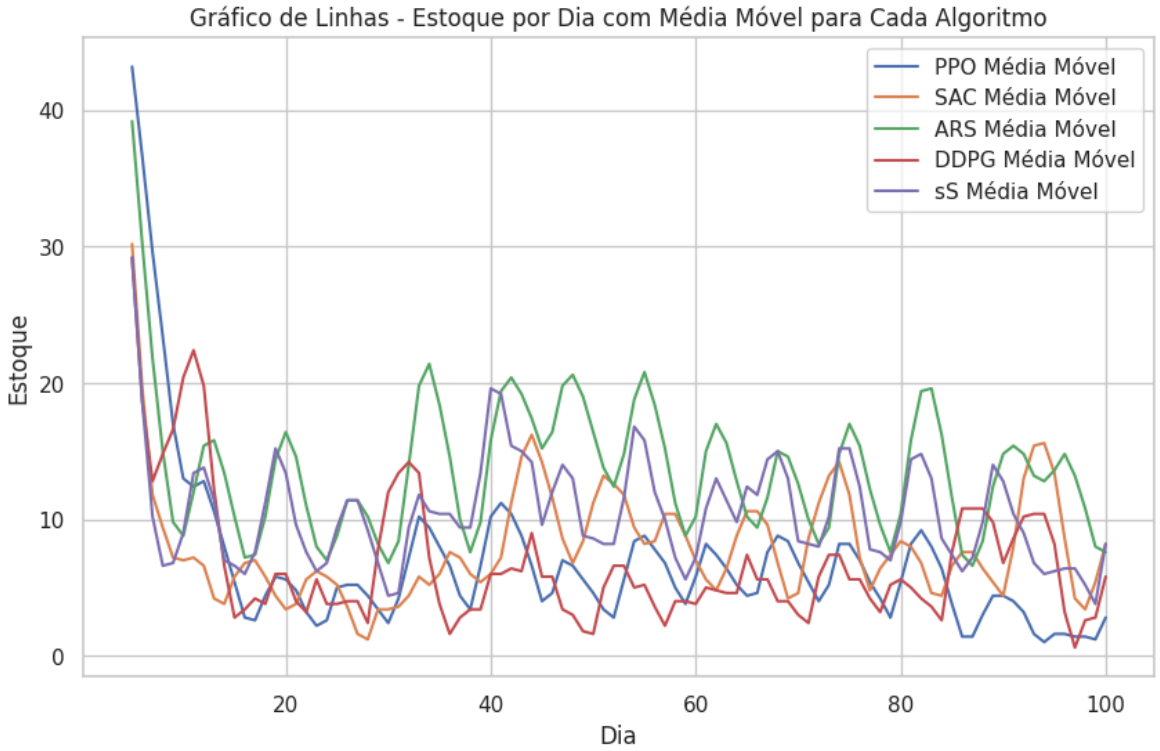


Figura 4
Gráfico de Linhas - Estoque por Dia com Média Móvel para Cada Algoritmo

Método

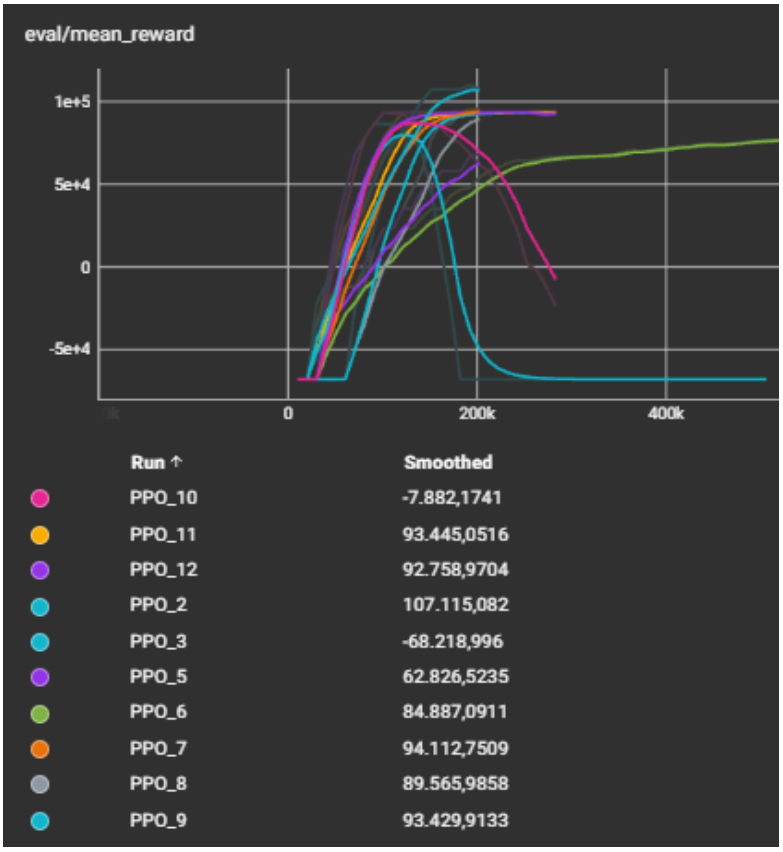
Para a realização dos experimentos, adotei uma variedade de hiperparâmetros para os diferentes algoritmos de aprendizado por reforço testados. Essa abordagem permitiu explorar diversas configurações e entender como cada algoritmo se comporta em diferentes contextos.

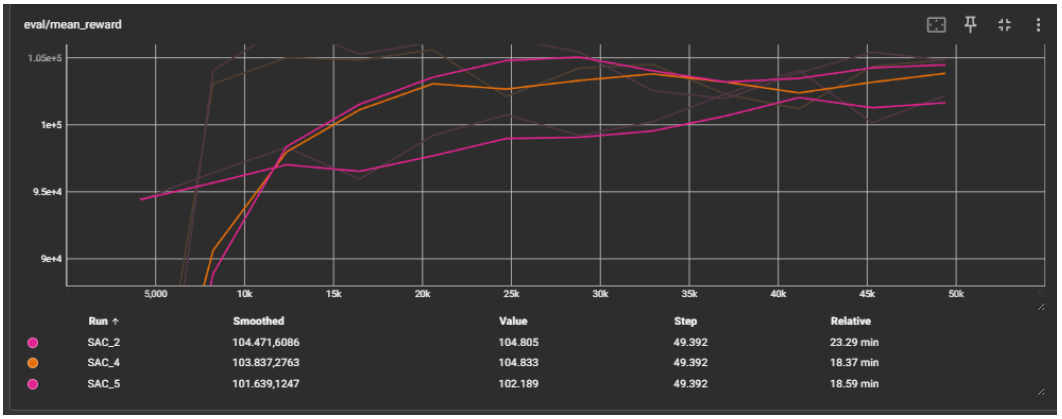
O Proximal Policy Optimization (PPO) , um algoritmo que antes não estava dando resultados agora apresentou um desempenho razoável, embora não tenha alcançado a melhor performance entre os algoritmos testados. O Soft Actor-Critic (SAC) se destacou ao conseguir obter resultados positivos e exibir um desempenho robusto em diferentes

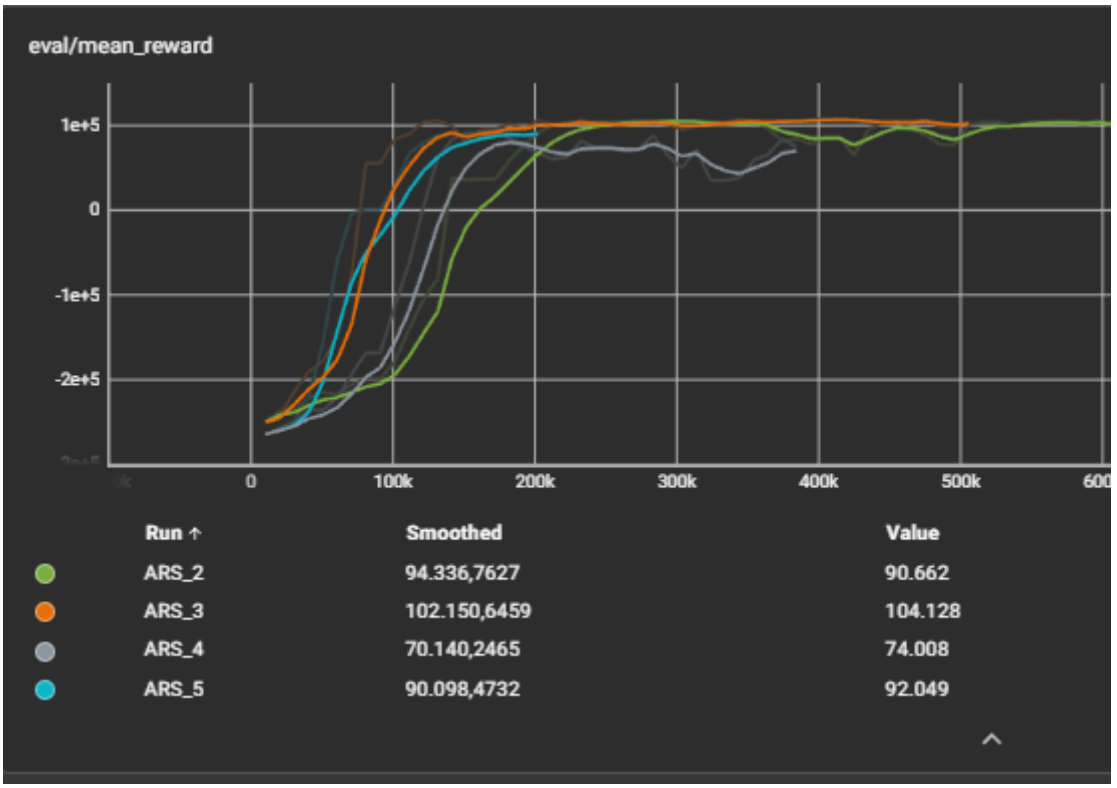
demandas, indicando uma boa capacidade de aprendizado. No entanto, vale ressaltar que o algoritmo Asynchronous Advantage Actor-Critic (A2C) não obteve resultados satisfatórios durante os testes, não conseguindo convergir durante o treinamento.

Para fornecer uma visão mais aprofundada desses resultados, irei apresentar os logs de recompensa obtidos durante as avaliações feitas durante o treinamento. Assim facilitará a compreensão das nuances e padrões específicos de cada algoritmo durante o treinamento. Além de mostrar poder mostrar os testes feitos para se encontrar os melhores hiperparâmetros.

O notebook para ver todo o código está disponível em: [TreinamentoRL-2.ipynb]




eval/mean_reward
1.05e+5
1e+5
9.5e+4
9e+4
5,000
10k
15k
20k
25k
30k
35k
40k
45k
50k
Run
Smoothed
Value
Step
Relative
SAC_2
104.471,6086
104.805
49.392
23.29 min
SAC_4
103.837,2763
104.833
49.392
18.37 min
SAC_5
101.639,1247
102.189
49.392
18.59 min


eval/mean_reward
1e+5
0
-1e+5
-2e+5
0
100k
200k
300k
400k
500k
Run
Smoothed
Value
ARS_2
94.336,7627
90.662
ARS_3
102.150,6459
104.128
ARS_4
70.140,2465
74.008
ARS_5
90.098,4732
92.049