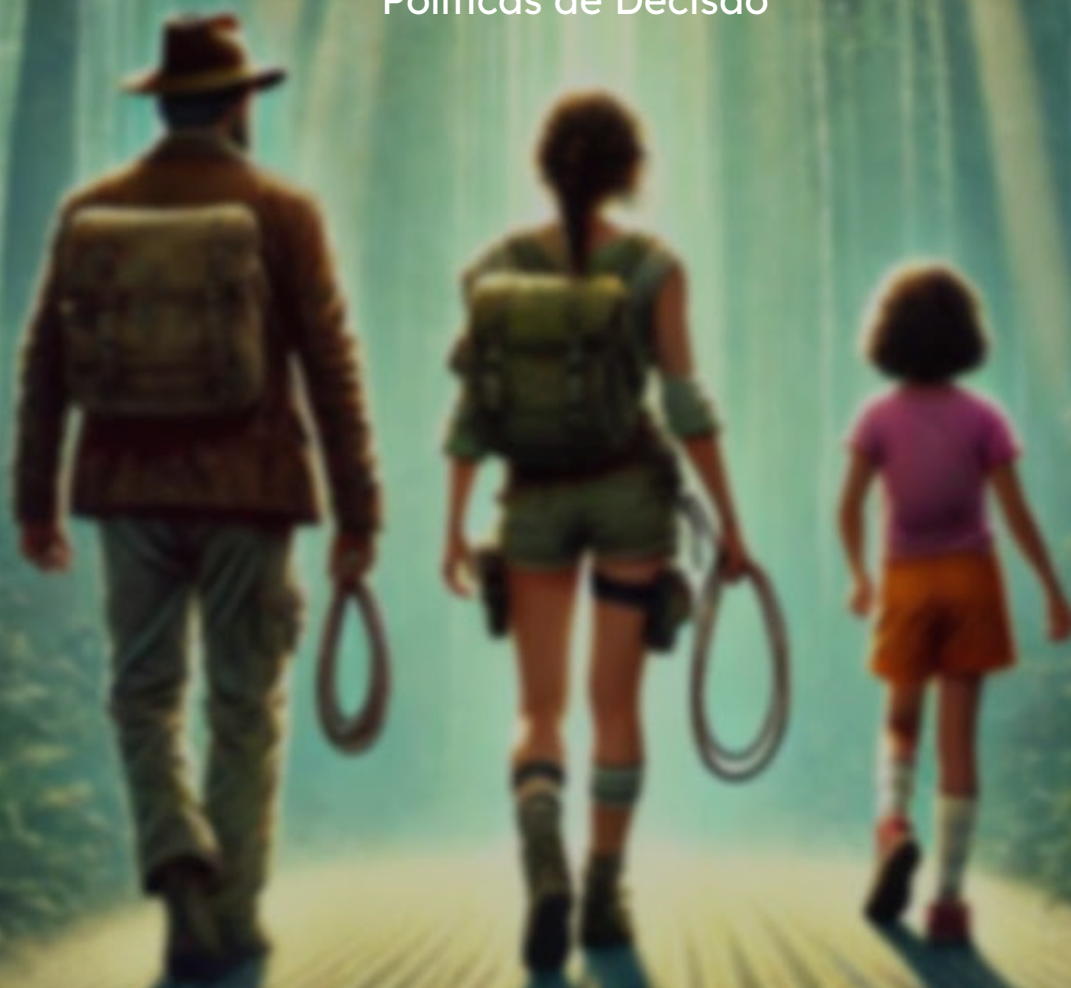


Desafios do Trade-Off Exploração-Exploração em Aprendizado por Reforço Profundo

Estudo Comparativo de Técnicas em Cenários Simulados para Otimização de
Políticas de Decisão



Fabrycio Leite Nakano Almada



UFG

UNIVERSIDADE
FEDERAL DE GOIÁS

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

FABRYCIO LEITE NAKANO ALMADA

**Desafios do Trade-Off Exploração-Exploração em Aprendizado por
Reforço Profundo**

Estudo Comparativo de Técnicas em Cenários Simulados para Otimização de
Políticas de Decisão

Goiânia
2025



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): FABRYCIO LEITE NAKANO ALMADA

Título do trabalho: Desafios do Trade-Off Exploração-Exploração em Aprendizado por Reforço Profundo

Estudo Comparativo de Técnicas em Cenários Simulados para Otimização de Políticas de Decisão

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [X] SIM [] NÃO¹

[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(a)(s) autor(a)(es)(as) e ao(a) 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 **Fabrycio Leite Nakano Almada, Discente**, em 12/01/2025, às 21:29, 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 15/01/2025, às 16:13, 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 **5089562** e o código CRC **1CDC8623**.

Referência: Processo nº 23070.001557/2025-04

SEI nº 5089562

FABRYCIO LEITE NAKANO ALMADA

**Desafios do Trade-Off Exploração-Exploração em Aprendizado por
Reforço Profundo**

Estudo Comparativo de Técnicas em Cenários Simulados para Otimização de
Políticas de Decisão

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

2025

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

ALMADA, FABRYCIO LEITE NAKANO

Desafios do Trade-Off Exploração-Exploração em Aprendizado por
Reforço Profundo [manuscrito] : Estudo Comparativo de Técnicas em
Cenários Simulados para Otimização de Políticas de Decisão /
FABRYCIO LEITE NAKANO ALMADA. - 2025.

142 f.

Orientador: Prof. Dr. Fernando Marques Federson.

Trabalho de Conclusão de Curso (Graduação) - Universidade
Federal de Goiás, Instituto de Informática (INF), Inteligência
Artificial, Goiânia, 2025.

1. inteligência artificial. 2. aprendizado por reforço profundo. 3.
exploração. I. Federson, Fernando Marques , orient. II. Título.

CDU 004

FABRYCIO LEITE NAKANO ALMADA

**Desafios do Trade-Off Exploração-Exploração em Aprendizado por
Reforço Profundo**

Estudo Comparativo de Técnicas em Cenários Simulados para Otimização de
Políticas de Decisão

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: 17 de dezembro de 2024.



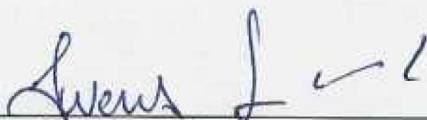
Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)



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



Prof. Dr. Anderson da Silva Soares
Coordenador do BIA (INF-UFG)



Prof. Dr. Iwens Gervasio Sene Junior
(INF-UFG)

FABRYCIO LEITE NAKANO ALMADA

Desafios do Trade-Off Exploração-Exploração em Aprendizado por Reforço Profundo

Estudo Comparativo de Técnicas em Cenários Simulados para Otimização de Políticas de Decisão

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 Profundo (Exploration)**. 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, modelos grandes de linguagem, geração automática de datasets.

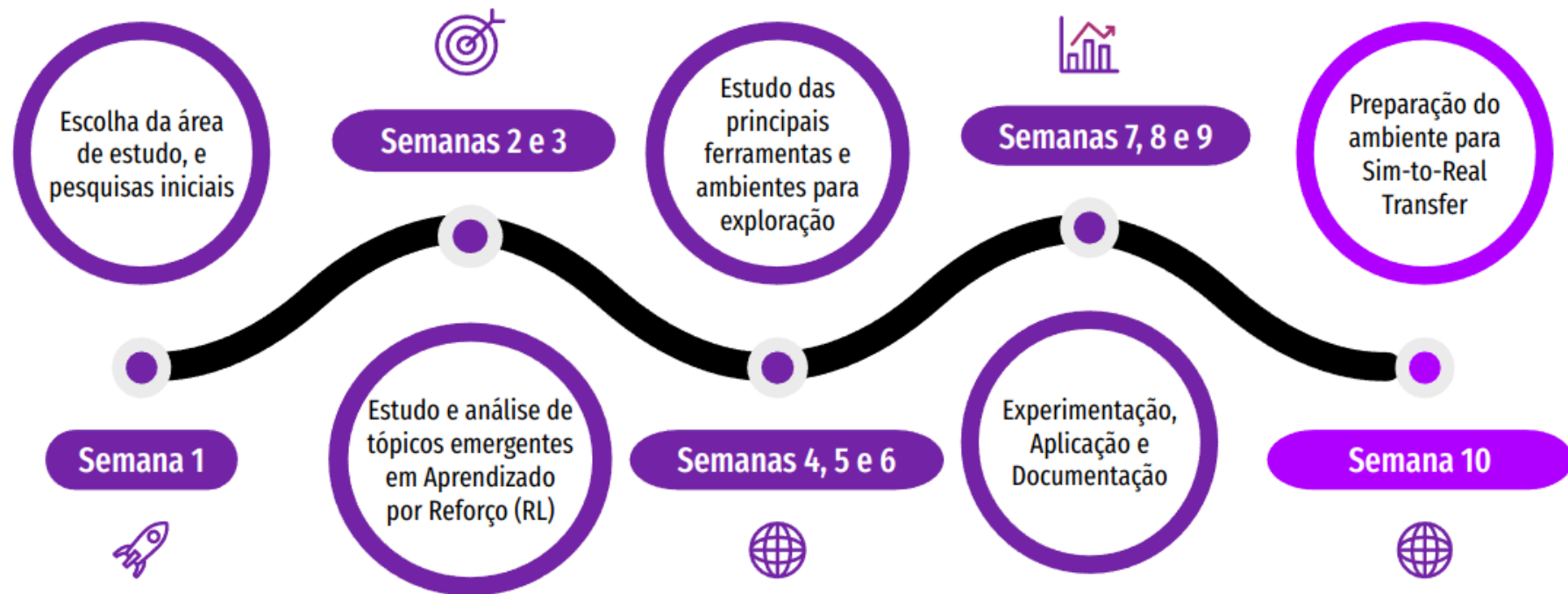
ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **Deep Reinforcement Learning (Exploration)**. 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, large language models, automatic dataset generation.

Goiânia
2025

Minha Jornada



Fabrycio Leite Nakano Almada

Especialista em: Aprendizado por Reforço Profundo (Exploration)

MINHA JORNADA

Nome: Fabrycio Leite Nakano Almada

Especialidade: Aprendizado por Reforço Profundo (Exploration)

Objetivo deste documento

Durante o processo da disciplina Residência em IA¹, 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 começou na **Semana 1**, com o desafio inicial de definir a área de estudo que orientaria minha especialização. A exploração dos congressos acadêmicos, em especial do **2024 Word Congress in Computer Science, Computer Engineering, & Applied Computing**, sobretudo na **26th Int'l Conf on Artificial Intelligence**, revelou-me a diversidade e profundidade dos grandes temas, proporcionando insights importantes. Em um primeiro momento, minha escolha se deu a um tema diferente do escolhido para o processo. Isso se deu devido a experiências anteriores que me levaram a inicialmente escolher um tema de maior domínio da minha parte. Entretanto, após conversa com colegas de curso e alguns professores, decidi me desafiar e escolher um tema que, embora possuísse um interesse latente no mesmo, meu domínio do assunto era limitado, tornando assim a minha residência em uma jornada do (quase) zero à especialidade. A área de estudo escolhida então foi a de Aprendizado por Reforço, mais particularmente a área de aprendizado por reforço profundo. Com o objetivo determinado, fui então me imergir na área, com um esforço em resgatar o desenvolvimento histórico da área de Aprendizado por Reforço. Essas

¹ Dez semanas, entre setembro de 2024 e dezembro de 2024.

primeiras impressões estão detalhadas no **Apêndice 1**. Dois artigos particularmente marcantes foram o de Watkins, 1969, uma tese de Doutorado que apresenta o *Q-learning*, algoritmo basilar da área de reforço, e o Mnih et Al., 2013, que inaugura a sub-área que integra Aprendizado por reforço com Redes Neurais Artificiais.

Nas **Semanas 2 e 3**, concentrei-me em aprofundar meu conhecimento por meio da leitura e análise de artigos científicos, com foco em tópicos emergentes e de relevância dentro do Aprendizado por Reforço e áreas correlatas. Este processo envolveu a análise crítica de artigos científicos publicados na primeira edição da *RL Conference* (<https://rl-conference.cc>), proporcionando um panorama das tendências atuais e desafios enfrentados pela área. Esses artigos, devidamente listados no **Apêndice 2**, foram essenciais para organizar meu conhecimento e guiar o direcionamento dos estudos. Entre os artigos mais marcantes, destacaram-se *Building Machines That Learn and Think Like People* (2016, Lake et Al.), que explora paradigmas cognitivos no aprendizado de máquina, e *Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning* (2021, Fuchs et Al.), que demonstram avanços na aplicação de IA em simulações. Esses estudos foram sistematicamente organizados em tabelas, categorizando Nome, Descrição e Proposta, conforme detalhado no **Apêndice 2**. Além disso, aprofundei-me em abordagens específicas, incluindo adversariais e NLP, como Adversarial Reinforcement Learning under Partial Observability e Double Deep Q-Networks (DDQN), que são descritas no **Apêndice 3**. Essas análises não apenas ampliaram meu entendimento teórico, mas também ajudaram a consolidar as bases para futuras aplicações práticas.

Nas **Semanas 4, 5 e 6**, o foco esteve no estudo e compreensão das principais ferramentas e ambientes utilizados no Aprendizado por Reforço, bem como na realização de experimentos iniciais. Essa etapa começou com a elaboração de um resumo preliminar sobre *Exploration in Deep Reinforcement Learning: A Survey*, registrado no **Apêndice 4**, onde foram listados benchmarks relevantes, como Atari Games, VizDoom, Malmo e Mujoco. Esse resumo destacou desafios futuros, como o dilema exploração-exploração (que terminou por ser o tema que me gerou o maior interesse), problemas de TV ruidosa e a transferabilidade, além de explorar fluxos relacionados à segurança e eficiência na exploração. Um marco importante foi o treinamento de um agente com o algoritmo PPO,

utilizando o ambiente ML-Agents (Unity), e o registro do agente treinado no Hugging Face. Já no **Apêndice 5**, o resumo inicial foi incrementado com conteúdos adicionais, incluindo documentações de benchmarks e conceitos extraídos do artigo *Exploration in Deep Reinforcement Learning: A Survey*. Também foi criada uma tabela relacionando ferramentas com conceitos de *Safe Exploration* e catalogados os principais ambientes e frameworks utilizados no campo, com fontes retiradas de repositórios curados no GitHub e artigos especializados. O **Apêndice 6** documenta o progresso das experimentações e testes práticos realizados durante a semana, que englobaram o uso de ambientes como Stable-Retro, Open-Spiel, VizDoom, TextWorld, Gym-Trading, ML-Agents e Atari Learning Environments. Essas iniciativas consolidaram a base prática necessária para a fase de validação e aplicação.

Nas **Semanas 7, 8 e 9**, o foco esteve na experimentação, aplicação prática e documentação, explorando conceitos teóricos em contextos simulados. Na Semana 7, foram realizados experimentos detalhados com o algoritmo Deep Q-learning aplicado ao problema do *Cartpole*, bem como com estratégias de exploração e exploração no contexto de Multi-Armed Bandit. Esses experimentos, devidamente documentados, estão descritos no **Apêndice 7**, juntamente com os testes realizados nos ambientes ML-Agents e TextWorld, com foco no impacto do tradeoff entre exploração e exploração. O ambiente TextWorld foi explorado em cenários como A Simple Game, The Cooking Game e Treasure Hunter, permitindo uma análise rica de como os agentes adaptam estratégias em narrativas dinâmicas. Na Semana 8, os esforços concentraram-se em aprofundar o uso do TextWorld, com a criação de missões complexas e a adaptação de agentes especializados em jogos baseados em texto, como o *LeDeepChef*, um agente de aprendizado profundo para famílias de jogos baseados em texto. Paralelamente, o ambiente de desenvolvimento Unity ML-Agents foi configurado, e materiais como tutoriais, vídeos explicativos e exemplos práticos básicos foram levantados para guiar os experimentos, conforme documentado no **Apêndice 8**. Já durante a Semana 9, houve um avanço significativo com a criação de um repositório para consolidar os testes realizados nos ambientes ML-Agents e TextWorld, disponível em <https://github.com/Fabrycio-Nakano/Studies-in-Deep-Reinforcement-Learning-Exploration-and-Exploitation>. Testes avançados em ambientes do ML-Agents, como Pushblock, Pyramids,

WallJump e SoccerTwos, permitiram desenvolver habilidades motoras, estratégias competitivas e de cooperação, além de interação com objetos no ambiente. Adicionalmente, foi realizada uma tentativa inicial de aplicar aprendizado por reforço no robô hexápode RealAnt, conforme descrito no **Apêndice 9**. Essas etapas foram fundamentais para consolidar o aprendizado prático e documentar insights valiosos para futuras implementações.

Por fim, na **Semana 10**, concentrei-me na preparação do ambiente necessário para a etapa de Sim to Real Transfer, um marco crítico para validar os conceitos aprendidos em um contexto real. O trabalho incluiu a leitura e análise do artigo *RealAnt: An Open-Source Low-Cost Quadruped for Education and Research in Real-World Reinforcement Learning*, que forneceu insights fundamentais sobre a aplicação de aprendizado por reforço em ambientes físicos. Com base nisso, documentei e descrevi o processo de reprodução, treinamento e análise do repositório `realant-rl`, que foi explorado em detalhes no notebook `realant-rl(walk).ipynb`, reproduzido no Google Colab. Além disso, revisei o repositório *A curated list of awesome exploration RL resources* para integrar os recursos mais atualizados ao meu estudo, consolidando referências essenciais para a etapa de transferência. Também revisei e atualizei a taxonomia criada no *Stage-Gate 5*, relacionada a *Exploration in Reinforcement Learning*, incorporando novas perspectivas e ferramentas exploradas ao longo do processo. Essa etapa, documentada no **Apêndice 10**, consolidou as bases técnicas e conceituais para garantir a replicabilidade e eficiência dos testes no ambiente real, marcando o encerramento de uma etapa desafiadora, porém enriquecedora, desta jornada.

Em função de tudo que vivi nesta jornada, gostaria de deixar registrado que o aprendizado foi muito além do domínio técnico. Esta experiência desafiadora me ensinou a importância de persistir diante da incerteza, buscar continuamente o aperfeiçoamento e valorizar tanto o processo quanto os resultados alcançados. Partindo de um conhecimento inicial limitado, aprofundei-me em conceitos teóricos, explorei aplicações práticas e enfrentei desafios reais, culminando na integração entre ambientes simulados e o mundo físico. Cada etapa foi marcada por momentos de reflexão, descoberta e superação, que não apenas expandiram minha compreensão sobre Aprendizado por Reforço, mas também

consolidaram habilidades que levarei para toda a minha trajetória acadêmica e profissional. Essa jornada, registrada em cada apêndice e experimento documentado, representa não apenas o progresso técnico, mas também o amadurecimento intelectual e pessoal que experimentei ao longo dessas Semanas.

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 set. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Fabrycio Leite Nakano Almada

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

Nesta primeira entrega, houve da minha parte um esforço em resgatar o desenvolvimento histórico da área de Aprendizado por reforço e isso se desenvolveu a partir das seguintes etapas:

1. Estudo do artigo Deep Reinforcement Learning: A brief survey (Arulkumaran et Al., 2017) (<https://discovery.ucl.ac.uk/id/eprint/10083557/1/1708.05866v2.pdf>)
2. Estudo do vídeo:  Markov Decision Processes - Computerphile
3. Leitura dos seguintes artigos:
 - a. "[Learning from Delayed Rewards](#) " (Watkins, 1989)
 - b. "[Policy Gradient Methods for Reinforcement Learning with Function Approximation](#)" (Sutton et Al. 1999)
 - c. [Playing atari with deep reinforcement learning](#) (Mnih et Al., 2013)
 - d. [Trust Region Policy Optimization](#) (Schulman et. Al, 2015)
 - e. [Asynchronous Methods for Deep Reinforcement Learning](#) (Mnih et Al., 2016)
 - f. [Proximal Policy Optimization Algorithms](#) (Schulman et Al., 2017)
 - g. [Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm](#) (Silver et Al., 2017)
 - h. [Curiosity-driven Exploration by Self-supervised Prediction](#) (Pathak et Al., 2017)

Resumo dos artigos:  Reinforcement Learning: roots

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

- Realizar a leitura e análise de artigos de estado da arte, com especial atenção para os disponíveis em <https://rl-conference.cc>.
- Identificar tópicos emergentes e relevantes dentro do Aprendizado por Reforço e outras áreas correlatas, sobretudo com aplicações além da robótica.

- Aprofundar a pesquisa nesses tópicos com base na percepção obtida, buscando artigos complementares que explorem os desafios e soluções mais recentes.

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

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go! ▾

[Arquivo Reinforcement Learning: roots citado no Termo de Aceite de Entrega de 19 de setembro]

- **Deep Reinforcement Learning: A Brief Survey**

O artigo "Deep Reinforcement Learning: A Brief Survey" oferece uma análise detalhada do estado da arte do aprendizado por reforço profundo (DRL), abordando como essa combinação de aprendizado profundo e aprendizado por reforço melhora a capacidade de sistemas autônomos para realizar tarefas complexas. Ele revisa algoritmos como DQN (Deep Q-Networks), A3C (Asynchronous Advantage Actor-Critic) e PPO (Proximal Policy Optimization), discutindo seus benefícios, limitações e desafios, como a exploração eficiente, a estabilidade do treinamento e a capacidade de generalização.

O DRL tem sido aplicado em diversas áreas, como robótica, jogos (com destaque para a performance em jogos como Atari e Go), sistemas autônomos de navegação, e até mesmo no tratamento de problemas em visão computacional. No entanto, desafios persistem, como o custo computacional elevado, a dificuldade em generalizar para novos ambientes e a sensibilidade a mudanças nos hiperparâmetros de treino.

Além disso, o artigo também destaca como a combinação dessas duas áreas é promissora para lidar com ambientes dinâmicos, onde a tomada de decisão precisa

ser continuamente ajustada com base em interações sucessivas e aprendizado de recompensas. No entanto, o DRL ainda enfrenta questões como a instabilidade do treinamento, a necessidade de redes robustas e a escassez de soluções generalizáveis para problemas mais amplos.

- **Learning from Delayed Rewards (1989)**

No artigo, Watkins propõe uma metodologia para que agentes autônomos aprendam a tomar decisões sequenciais em ambientes onde as recompensas podem ser atrasadas. A principal contribuição é o desenvolvimento do algoritmo Q-Learning, que permite que um agente aprenda uma política ótima sem a necessidade de um modelo prévio do ambiente.

Algoritmo Q-Learning

O Q-Learning é um algoritmo de aprendizado por reforço *off-policy*, o que significa que o agente pode aprender a política ótima independentemente da política seguida durante a exploração. O algoritmo funciona atualizando uma função de valor de ação, conhecida como **função Q**, que estima a qualidade de uma ação em um determinado estado.

Funcionamento do Q-Learning:

1. **Inicialização:** Inicializa a tabela $Q(s, a)$ arbitrariamente para todos os pares estado-ação.
2. **Observação do Estado Atual:** O agente observa o estado atual s .
3. **Seleção de Ação:** Utiliza uma política (por exemplo, ϵ -greedy) para selecionar uma ação a .
4. **Execução da Ação:** O agente executa a ação a , observa a recompensa r e o próximo estado s' .
5. **Atualização da Função Q:** Atualiza o valor $Q(s, a)$ usando a equação de atualização:

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

Onde:

- **α (alpha):** Taxa de aprendizado.
- **γ (gamma):** Fator de desconto que prioriza recompensas imediatas sobre futuras.

6. **Transição de Estado:** O agente transita para o estado s' e repete o processo.

Propriedades

- **Convergência Garantida:** Sob certas condições (como decaimento apropriado da taxa de aprendizado e exploração suficiente), o Q-Learning converge para a função Q ótima, permitindo ao agente seguir a política ótima.
- **Aprendizado Fora da Política:** O agente pode explorar o ambiente usando uma política diferente da política ótima que está sendo aprendida, o que facilita a exploração sem comprometer a aprendizagem.

Limitações e Avanços Posteriores

Embora o Q-Learning seja poderoso, ele possui algumas limitações, como a escalabilidade para ambientes com grandes espaços de estado e ação. Para mitigar essas limitações, foram desenvolvidas extensões e variações, como:

- **Deep Q-Learning (DQN):** Integra redes neurais profundas para aproximar a função Q, permitindo o tratamento de espaços de estado contínuos e de alta dimensionalidade.
- **Q-Learning com Funções de Valor Aproximadas:** Utiliza técnicas de aproximação para generalizar o aprendizado em espaços de estado extensos.
- Policy Gradient Methods for Reinforcement Learning with Function Approximation (2000)

O artigo introduz métodos de gradiente de política no contexto de aprendizado por reforço (Reinforcement Learning, RL) com aproximação de função. Este trabalho representa uma inovação significativa no campo do RL, fornecendo um

framework matemático rigoroso para otimizar políticas de decisão em ambientes contínuos e complexos.

As principais inovações do artigo podem ser destacadas nos seguintes pontos:

1. **Métodos de Gradiente de Política:** O trabalho introduz os métodos de gradiente de política como uma alternativa ao aprendizado baseado em valores, como Q-learning. O método explora diretamente o espaço de políticas ao invés de buscar uma estimativa precisa do valor de uma função de valor, que pode ser difícil em domínios com espaços de estado e ação grandes ou contínuos.
2. **Aproximação de Função:** Um dos principais desafios em RL é a escalabilidade dos métodos em grandes espaços de estados e ações. O artigo lida com isso utilizando aproximação de função para representar as políticas, como funções paramétricas (ex.: redes neurais). Isso permite que o método funcione em problemas de grande escala ou contínuos, onde métodos baseados em tabelas (como Q-learning) não são viáveis.
3. **Política Estocástica:** O artigo se baseia no uso de políticas estocásticas (em vez de determinísticas), que facilitam a exploração e evitam alguns dos problemas relacionados à convergência em políticas determinísticas. Eles propõem uma formulação clara de gradiente de política estocástica, que se tornou um fundamento central em muitas abordagens modernas de RL.
4. **Derivação do Gradiente de Política:** O trabalho apresenta uma derivação do gradiente de política para RL com aproximação de função, que é livre de viés. Isso foi crucial, já que métodos anteriores de gradiente de política enfrentavam desafios com viés ao aplicar aproximação de função. Eles mostraram que a aproximação de valor pode ser usada junto com o gradiente de política de forma eficiente e rigorosa.

- **Playing Atari with Deep Reinforcement Learning (2013)**

O artigo introduz o algoritmo DQN (*Deep Q-Network*), combinando aprendizado por reforço (RL) com redes neurais profundas para jogar jogos do Atari a partir de entrada visual, sem conhecimento prévio das dinâmicas dos jogos.

Inovações e Contribuições

1. **Combinação de RL e Redes Neurais Profundas:**

- A principal inovação foi a utilização de uma rede neural profunda para aproximar a função de valor Q, algo que até então era desafiador devido à instabilidade e dificuldade de convergência de métodos anteriores. O artigo introduziu uma nova arquitetura que conseguiu superar esses problemas e aprender diretamente a partir de representações de alto nível de estados do ambiente, como imagens de pixels.

2. **Replay Experience:**

- Um dos desafios do uso de redes neurais com RL era a correlação entre os dados sequenciais, que gerava instabilidades no treinamento. O *Replay Experience* foi uma das inovações centrais: o agente armazena transições (estado, ação, recompensa, próximo estado) em uma memória e, ao treinar a rede, amostra aleatoriamente essas transições. Isso reduz a correlação nos dados e melhora a estabilidade e eficiência no aprendizado.

3. **Separação de Redes de Alvo e Treinamento:**

- Outra inovação fundamental foi o uso de uma rede alvo (*target network*), separada da rede principal de treinamento. Essa rede alvo é uma cópia da rede principal que é atualizada a intervalos regulares. Ao manter a rede alvo fixa durante pequenos períodos de treinamento, eles conseguiram reduzir as oscilações e melhorar a convergência do agente.

4. **Aprendizado Direto de Pixels:**

- O trabalho demonstrou a capacidade de o agente aprender diretamente a partir de dados brutos de pixels. O agente é treinado a partir de entradas visuais 2D dos jogos do Atari, sem qualquer modelagem explícita das dinâmicas do jogo ou do espaço de estado,

confiando completamente no processo de aprendizado para extrair essas representações.

Estado da Arte em Reinforcement Learning na Época

Na época da publicação, o campo de RL estava em grande parte limitado a problemas relativamente simples ou com espaço de estado pequeno. Métodos como Q-Learning e SARSA eram populares, mas tinham limitações ao serem aplicados em ambientes com grandes espaços de estados e ações, ou quando exigiam o uso de representações de alto nível, como imagens de entrada. Redes neurais profundas já vinham ganhando destaque em tarefas de classificação e reconhecimento de imagem, mas sua aplicação ao RL ainda era limitada devido à dificuldade de estabilizar o treinamento.

O artigo foi revolucionário por mostrar que um agente de aprendizado por reforço pode superar desempenhos humanos em uma ampla gama de jogos do Atari utilizando apenas informações visuais brutas. A arquitetura do DQN se tornou um ponto de partida para uma série de avanços subsequentes, como algoritmos A3C, DDPG, PPO e outros métodos que combinam aprendizado profundo com aprendizado por reforço.

- **Asynchronous Methods for Deep Reinforcement Learning (2016)**

O artigo introduziu uma nova classe de algoritmos chamada **Asynchronous Advantage Actor-Critic (A3C)**, que representa um avanço significativo no campo de aprendizado por reforço profundo (Deep Reinforcement Learning). O artigo propõe o uso de métodos assíncronos para resolver os problemas de eficiência e estabilidade enfrentados pelos algoritmos anteriores, como o DQN, ao treinar redes neurais profundas.

Inovações:

1. Métodos Assíncronos:

- A principal inovação do artigo é o uso de múltiplos agentes que interagem com o ambiente de forma assíncrona, executando diferentes instâncias do ambiente em paralelo. Isso permite um uso mais eficiente dos recursos computacionais, eliminando a necessidade de memória

de replay (usada no DQN), e garante que os agentes explorem diferentes partes do espaço de estado simultaneamente.

2. A3C - Asynchronous Advantage Actor-Critic:

- O A3C combina aprendizado baseado em políticas (Actor) e aprendizado baseado em valores (Critic) em uma arquitetura eficiente. O método utiliza um único agente que otimiza tanto a política (a ação a ser tomada) quanto a função de valor (avaliar a vantagem dessa ação em relação ao estado atual), de maneira que ambos se reforçam. Este método se destaca por ser altamente eficiente e fácil de paralelizar, o que melhora significativamente a convergência em relação a métodos anteriores.

3. Sem Memória de Replay:

- Uma das limitações do DQN era o uso de *Replay Experience*, que exigia uma grande quantidade de memória e processamento adicional para armazenar e amostrar as transições passadas. O A3C elimina a necessidade de replay ao utilizar o aprendizado on-policy, ou seja, os agentes aprendem diretamente a partir de suas interações com o ambiente, em tempo real.

4. Uso de Vários Ambientes:

- O artigo demonstra que a execução de vários threads que interagem com diferentes cópias do ambiente de forma assíncrona ajuda a estabilizar o treinamento, pois evita que os parâmetros da rede neural oscilem de maneira instável. Isso acelera a convergência e garante uma exploração mais abrangente do espaço de estados.

5. Redução da Correlação nas Atualizações:

- Ao contrário de métodos anteriores que utilizavam uma única cópia do ambiente para gerar dados de treinamento, a natureza assíncrona dos métodos apresentados aqui reduz significativamente a correlação entre atualizações consecutivas, o que melhora a robustez e estabilidade do algoritmo.

- **"Trust Region Policy Optimization (TRPO)" (2015)**

O artigo introduziu um método de otimização de políticas robusto e eficiente para aprendizado por reforço (Reinforcement Learning, RL). TRPO se destacou por fornecer uma abordagem de otimização que equilibra a exploração e a estabilidade,

superando alguns problemas comuns relacionados à divergência de políticas que ocorriam em algoritmos baseados em gradiente de política.

Inovações e Contribuições

1. Região de Confiança para Otimização de Políticas:

- O método TRPO propõe um algoritmo que limita a atualização da política dentro de uma região de confiança (*trust region*), o que significa que as mudanças na política a cada iteração são controladas para garantir que a nova política não se desvie drasticamente da anterior. Isso ajuda a evitar atualizações muito agressivas que poderiam levar a políticas subótimas ou divergências no treinamento.

2. Divergência de Kullback-Leibler (KL) como Métrica de Restrição:

- Para implementar a noção de região de confiança, o algoritmo impõe uma restrição sobre a divergência de Kullback-Leibler (KL) entre a política atual e a nova política. O TRPO otimiza a função objetivo de aprendizado por reforço, garantindo que a mudança na política seja limitada por uma cota no KL, garantindo estabilidade e melhor desempenho em domínios complexos.

3. Política Proximal:

- TRPO é considerado uma melhoria dos métodos de gradiente de política, pois oferece uma abordagem mais direta para garantir que as atualizações não se afastem demais da política atual. Em vez de otimizar diretamente o gradiente de política, o algoritmo resolve um problema de otimização quadrático-convexo aproximado com uma restrição de KL, o que leva a atualizações mais seguras e estáveis.

4. Eficácia em Ambientes de Alto Desempenho:

- O artigo demonstrou que o TRPO supera muitos algoritmos contemporâneos de aprendizado por reforço, especialmente em ambientes complexos de controle contínuo, como simulações robóticas. Ele provou ser mais estável e eficiente, sendo amplamente utilizado em problemas que requerem políticas sensíveis e ajustáveis, como controle de robôs e jogos.

- "Proximal Policy Optimization Algorithms" (2017)

O artigo "*Proximal Policy Optimization Algorithms*" (PPO), publicado por John Schulman e colaboradores em 2017, introduziu um novo método de otimização de políticas que combina a simplicidade e eficiência computacional com a robustez necessária para ambientes complexos de aprendizado por reforço. PPO tornou-se uma alternativa direta aos métodos mais complexos, como o TRPO, oferecendo uma maneira mais fácil de implementar e ajustar algoritmos de aprendizado por reforço profundo.

Inovações e Contribuições

1. Combinação de Simplicidade e Estabilidade:
 - O principal objetivo do PPO era fornecer uma versão simplificada do Trust Region Policy Optimization (TRPO), mantendo a estabilidade das atualizações de política. PPO utiliza uma técnica de otimização baseada em gradiente que ajusta a política dentro de limites controlados, evitando as oscilações bruscas e instabilidade observadas em outros métodos de gradiente de política.
2. Clipping da Função Objetivo:
 - A principal inovação do PPO é a introdução de uma função de objetivo com um "clipping", que restringe a magnitude das mudanças de política. Em vez de garantir explicitamente que a divergência de Kullback-Leibler (KL) entre a nova política e a anterior seja pequena, como em TRPO, PPO limita as mudanças por meio de uma penalidade em torno da razão de probabilidade (probability ratio). Esse clipping garante que as atualizações de política não sejam muito grandes, prevenindo a degradação do desempenho.
3. Duas Versões de PPO:
 - O artigo apresenta duas variantes principais do PPO: uma versão com penalização da divergência KL e uma com o clipping na função de objetivo. A versão com clipping provou ser mais simples e, na maioria dos casos, mais eficiente, tornando-se a variante mais amplamente adotada.
4. Eficiência Computacional:

- Diferente do TRPO, que requer a solução de um problema de otimização complexo a cada atualização, o PPO é muito mais simples de implementar e eficiente computacionalmente. Ele pode ser executado com técnicas padrão de otimização por gradiente, como Adam, e não necessita de uma grande quantidade de ajustes e cálculos computacionalmente intensivos, tornando-o uma escolha popular para aplicações em larga escala.
5. Desempenho Superior em Tarefas de Controle Contínuo:
- O PPO se destacou por seu desempenho em tarefas de controle contínuo, como as simulações robóticas, e em ambientes complexos de aprendizado por reforço, como jogos e ambientes de simulação em larga escala, mostrando um equilíbrio robusto entre simplicidade e eficácia.

Estado da Arte em Reinforcement Learning na Época

Antes do PPO, o TRPO era amplamente utilizado devido à sua estabilidade, especialmente em tarefas complexas e de controle contínuo. No entanto, o TRPO era computacionalmente caro e de difícil implementação. O DDPG (Deep Deterministic Policy Gradient) também era popular, mas apresentava dificuldades com estabilidade em ambientes de alto ruído ou incerteza. O PPO foi projetado para preencher a lacuna entre a simplicidade de métodos como DDPG e a robustez oferecida pelo TRPO.

O artigo de PPO teve um impacto significativo na comunidade de aprendizado por reforço profundo, ao introduzir um método que balanceava a simplicidade com desempenho de ponta. Sua capacidade de combinar a robustez de métodos complexos, como o TRPO, com a eficiência e facilidade de implementação o tornaram o algoritmo de escolha para muitas tarefas, desde controle robótico até sistemas de jogos.

- **"Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm" (2017)**

O artigo apresenta um marco histórico no campo do aprendizado por reforço (Reinforcement Learning, RL) ao introduzir o **AlphaZero**, um algoritmo geral que

alcança desempenho de nível super-humano em jogos complexos como xadrez e shogi por meio de aprendizado por autajogo (self-play). O trabalho baseia-se em avanços anteriores feitos com o AlphaGo, o primeiro sistema a derrotar campeões mundiais no jogo Go, e estende esses princípios para jogos de tabuleiro estratégicos complexos.

Inovações e Contribuições

1. Algoritmo AlphaZero:

- O AlphaZero é um algoritmo geral de aprendizado por reforço que combina Deep Neural Networks e Monte Carlo Tree Search (MCTS) para jogar jogos de tabuleiro como xadrez, shogi e Go sem o uso de conhecimento humano específico do jogo. A rede neural é treinada para prever tanto as melhores ações (próximos movimentos) quanto a função de valor (probabilidade de vitória) com base na posição do tabuleiro.

2. Aprendizado por Autojogo (Self-Play):

- Uma das principais inovações do AlphaZero é o aprendizado por autajogo. O algoritmo não é alimentado com estratégias humanas ou base de dados de partidas anteriores, mas aprende jogando contra si mesmo. A cada iteração, ele ajusta a política e a função de valor com base nos resultados dos jogos simulados, melhorando progressivamente até atingir um nível super-humano de desempenho.

3. Integração com MCTS:

- O algoritmo utiliza Monte Carlo Tree Search (MCTS) como parte de sua estratégia de decisão durante o jogo. O MCTS explora várias sequências de movimentos simulados para identificar a ação mais promissora. A rede neural é utilizada para melhorar a eficiência do MCTS, fornecendo estimativas precisas de valor e políticas, o que reduz significativamente o número de simulações necessárias em comparação com versões anteriores do AlphaGo.

4. Generalidade do Algoritmo:

- O AlphaZero se destaca por ser um algoritmo geral de aprendizado por reforço, capaz de jogar e dominar diferentes jogos (xadrez, shogi e Go) com as mesmas regras de aprendizado. Isso demonstra a aplicabilidade ampla do método, sem a necessidade de ajustes específicos para cada jogo, o que foi um grande avanço na área de RL.

5. Desempenho Super-humano:

- O AlphaZero rapidamente superou os programas de computador mais fortes em xadrez (Stockfish) e shogi (Elmo), que até então eram considerados invencíveis. O algoritmo demonstrou que, a partir de regras simples de autajogo e sem conhecimento prévio, é possível superar programas que foram ajustados com base em décadas de pesquisa humana.

Antes do AlphaZero, o AlphaGo já havia revolucionado o campo ao derrotar campeões mundiais no Go, usando uma combinação de aprendizado supervisionado a partir de jogos humanos e autajogo. Porém, o AlphaGo ainda dependia de grandes conjuntos de dados de partidas humanas. Além disso, programas de xadrez e shogi eram baseados em abordagens altamente especializadas e otimizadas ao longo de muitos anos, utilizando regras codificadas manualmente e conhecimento heurístico.

- **"Curiosity-driven Exploration by Self-supervised Prediction" (2017)**

O artigo *"Curiosity-driven Exploration by Self-supervised Prediction"*, publicado por Deepak Pathak e colaboradores em 2017, apresenta um avanço importante no campo de aprendizado por reforço (Reinforcement Learning, RL) ao introduzir o conceito de exploração motivada pela curiosidade em ambientes desconhecidos. A ideia central do trabalho é usar uma forma de aprendizado auto-supervisionado para incentivar um agente a explorar novos estados no ambiente, mesmo na ausência de recompensas extrínsecas claras, o que é particularmente útil em cenários onde as recompensas são escassas ou inexistentes.

Inovações:

1. Exploração Guiada pela Curiosidade:

- O principal conceito introduzido é o de um mecanismo de exploração guiado pela curiosidade intrínseca, onde o agente é incentivado a explorar com base na novidade de um estado. Isso é implementado

através da previsão de mudanças de estado, onde o agente tenta prever a consequência de suas ações no ambiente. Quando há um erro na previsão, o agente recebe uma recompensa intrínseca, o que o incentiva a explorar mais.

2. Modelo de Predição Auto-supervisionada:

- O artigo propõe um modelo de aprendizado auto-supervisionado, no qual o agente utiliza uma rede neural para prever como suas ações alteram o estado do ambiente. O erro de predição (a diferença entre o estado previsto e o estado real observado) é utilizado como uma métrica de curiosidade. Quanto maior o erro, maior a novidade percebida pelo agente e, portanto, maior a recompensa intrínseca gerada.

3. Formulação com Redes Dinâmicas:

- A abordagem introduz uma rede neural dinâmica que aprende a prever as transições de estados ao longo do tempo. Esse modelo é treinado em tempo real, permitindo que o agente aprenda a navegar de forma autônoma em ambientes onde as recompensas externas são limitadas ou inexistentes. Isso é especialmente útil em ambientes exploratórios e sem mapeamento prévio, como jogos ou simulações de robótica.

4. Aplicação em Ambientes Complexos:

- O artigo demonstra a eficácia do mecanismo de curiosidade em uma série de experimentos, incluindo jogos complexos e ambientes de simulação 3D, onde as recompensas extrínsecas são raras. O mecanismo de curiosidade permite que o agente aprenda a explorar ambientes de forma eficiente e eficaz, mesmo sem receber recompensas externas diretas.

5. Curiosidade Baseada em Estado Próprio:

- Uma das inovações específicas do artigo é que a curiosidade é calculada com base em um modelo de predição do próprio agente, em vez de depender de modelos externos ou conhecimento prévio do ambiente. Essa abordagem de curiosidade intrínseca é o que permite que o agente aprenda de forma completamente auto-supervisionada.

Na época da publicação, grande parte do trabalho em RL focava-se em ambientes com recompensas explícitas e densas, como em jogos de tabuleiro ou ambientes simulados com objetivos bem definidos. No entanto, muitos ambientes do mundo real, especialmente aqueles envolvendo robótica ou exploração autônoma, oferecem

poucas ou nenhuma recompensa externa imediata. Isso gerava desafios para os agentes de RL, que tinham dificuldades em aprender comportamentos úteis nesses cenários.

O artigo é inovador porque propõe uma solução eficaz para o problema da escassez de recompensas, usando a curiosidade intrínseca como um método de incentivo. Essa abordagem abriu caminho para o desenvolvimento de agentes que podem explorar ambientes complexos e aprender comportamentos sem a necessidade de recompensas externas frequentes, tornando-o aplicável a uma gama mais ampla de cenários, incluindo robótica e simulações em grande escala.

Impacto a Longo Prazo:

- **Exploração em Ambientes Reais:** O mecanismo de curiosidade intrínseca tem aplicações diretas em robótica e veículos autônomos, onde o aprendizado baseado em recompensas explícitas pode ser difícil de implementar devido à natureza do ambiente.
- **Aprendizado Auto-supervisionado:** O uso de aprendizado auto-supervisionado para explorar e descobrir novos estados tornou-se um componente essencial em muitos algoritmos modernos de RL, especialmente aqueles que lidam com ambientes desconhecidos ou com tarefas de exploração.
- **Base para Exploração Contínua:** Esta abordagem de curiosidade também influenciou outros trabalhos no campo, especialmente aqueles focados em aprendizado contínuo, onde o agente deve continuar explorando e descobrindo novas áreas de conhecimento sem depender de recompensas externas.

Bibliografia:

ARULKUMARAN, Kai; DEISENROTH, Marc Peter; BRUNDAGE, Miles; BHARATH, Anil Anthony. A brief survey of deep reinforcement learning. arXiv preprint arXiv:1708.05866, 2017. Disponível em: <https://arxiv.org/abs/1708.05866>. Acesso em: 18 set. 2024.

MNIH, Volodymyr; BADIA, Adrià Puigdomènech; MIRZA, Mehdi; GRAVES, Alex; HARLEY, Tim; SILVER, David; KAVUKCUOGLU, Koray. **Asynchronous Methods**

for **Deep Reinforcement Learning**. *Proceedings of the 33rd International Conference on Machine Learning*, v. 48, p. 1928-1937, 2016.

MNIH, Volodymyr; KAVUKCUOGLU, Koray; SILVER, David; GRAVES, Alex; ANTONOGLOU, Ioannis; WIERSTRA, Daan; RIEDMILLER, Martin. **Playing Atari with Deep Reinforcement Learning**. arXiv preprint arXiv:1312.5602, 2013.

PATHAK, Deepak; AGARWAL, Pulkit; DOLLÁR, Piotr; EFROS, Alexei A.; DARRELL, Trevor. **Curiosity-driven Exploration by Self-supervised Prediction**. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, v. 2017, p. 16-17, 2017.

Richard S. et al. **Policy Gradient Methods for Reinforcement Learning with Function Approximation**. *Advances in Neural Information Processing Systems*, v. 12, p. 1057–1063, 2000.

SCHULMAN, John; LEVINE, Sergey; MORITZ, Philipp; JORDAN, Michael I.; ABBEEL, Pieter. **Trust Region Policy Optimization**. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, v. 37, p. 1889-1897, 2015.

SCHULMAN, John; WOLSKI, Filip; DHARIWAL, Prafulla; RADFORD, Alec; KLIMOV, Oleg. **Proximal Policy Optimization Algorithms**. arXiv preprint arXiv:1707.06347, 2017.

SILVER, David; HUBERT, Thomas; SCHRITTWIESER, Julian; ANTONOGLOU, Ioannis; LAI, Matthew; GUEZ, Arthur; LANCTOT, Marc; SIFRE, Laurent; KUFOURIS, George; GRAEPEL, Thore; LILLICRAP, Timothy; SIMONYAN, Karen; HASSABIS, Demis. **Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm**. arXiv preprint arXiv:1712.01815, 2017.

SUTTON, Richard S.; McALLESTER, David; SINGH, Satinder; MANSOUR, Yishay. **Policy Gradient Methods for Reinforcement Learning with Function Approximation**. *Advances in Neural Information Processing Systems*, v. 12, p. 1057-1063, 2000.

WATKINS, C. J. C. H. **Learning from Delayed Rewards**. 1989. Thesis (Ph.D.)—University of Cambridge, 1989.

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: 25 de set. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Fabrycio Leite Nakano Almada

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

Reinforcement Learning: 25/09

- Leitura e análise de artigos de estado da arte, com especial atenção para os disponíveis em <https://rl-conference.cc>.
- Identificar tópicos emergentes e relevantes dentro do Aprendizado por Reforço e outras áreas correlatas, sobretudo com aplicações além da robótica.
- Aprofundar a pesquisa nesses tópicos com base na percepção obtida, buscando artigos complementares que explorem os desafios e soluções mais recentes.
- Estruturação dos Artigos da Semana anterior em tabelas, dividida em Nome, Descrição e Proposta
- Estruturação dos Artigos da Semana atual em tabelas, dividida em Nome, Descrição e Proposta
 - Building Machines That Learn and Think Like People (2016) - Lake et Al.
 - Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning (2021) - Fuchs et al.
 - Benchmarking Safe Deep Reinforcement Learning in Aquatic Navigation (2021) - Marchesini et Al.
 - A Super-human Vision-based Reinforcement Learning Agent for Autonomous Racing in Gran Turismo (2024) - Vasco et Al.
 - Aquatic Navigation: A Challenging Benchmark for Deep Reinforcement Learning (2024) - Corsi et Al.
 - Diversity-Enhanced Reinforcement Learning in Complex Multi-Agent Systems (2024) - Kumar et Al.
- Divisão em Áreas de estudo
- Tabela de abordagens

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

- Buscar artigos relacionados a DRL aplicado a NLP
- Buscar artigos relacionados a Adversarial DRL
- Estudo e primeiros experimentos de ferramentas específicas de DRL

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

ACEITE DA ENTREGA:

LEONARDO ALVES: [Go! ▾](#)

Artigos Clássicos (GATE 1)

Nome do Artigo	Descrição	Inovação
Learning from Delayed Rewards (1971) - Christopher Watkins	Introduziu o algoritmo Q-Learning, que permite a agentes aprenderem com base na maximização de recompensas futuras.	Q-Learning, base para muitos algoritmos modernos de RL.
Policy Gradient Methods for Reinforcement Learning with Function Approximation (2000) - Richard S. Sutton et al.	Introduziu métodos de gradiente de política, permitindo que agentes aprendam políticas diretamente, ao invés de aprenderem o valor de ações ou estados.	Primeira introdução de métodos de gradiente de política, base para políticas determinísticas em RL.

Playing Atari with Deep Reinforcement Learning (2013) - Volodymyr Mnih et al.	Marcou um avanço significativo na aplicação do RL com redes neurais profundas, introduzindo o DQN (Deep Q-Network) para jogar Atari.	Primeiro uso bem-sucedido de deep learning com RL para controle de agentes complexos.
Asynchronous Methods for Deep Reinforcement Learning (2016) - Volodymyr Mnih et al.	Introduziu o A3C (Asynchronous Advantage Actor-Critic), acelerando o aprendizado por RL ao executar múltiplas instâncias de RL simultaneamente.	Introduziu técnicas assíncronas para melhorar a eficiência do aprendizado profundo.
Trust Region Policy Optimization (2015) - John Schulman et al.	Apresentou o TRPO (Trust Region Policy Optimization), uma abordagem robusta para otimização de políticas, trazendo mais estabilidade ao RL.	O TRPO trouxe mais estabilidade e segurança ao otimizar políticas em RL.
Proximal Policy Optimization Algorithms (2017) - John Schulman et al.	Uma extensão dos métodos de gradiente de política, o PPO equilibra exploração e exploração com simplicidade e eficácia.	PPO tornou-se amplamente popular por sua simplicidade e balanceamento eficaz entre exploração e exploração.

Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm (2017) - David Silver et al.	Descreveu o AlphaZero, que superou os melhores programas de xadrez e shogi usando autojogo e aprendizado por reforço.	AlphaZero demonstrou a capacidade de RL superar os melhores algoritmos por meio de autojogo.
Curiosity-driven Exploration by Self-supervised Prediction (2017) - Deepak Pathak et al.	Explorou aprendizado por curiosidade, onde agentes são motivados a explorar ambientes complexos pela imprevisibilidade do ambiente.	Introduziu aprendizado impulsionado pela curiosidade, uma abordagem inovadora para exploração em RL.

Artigos do GATE 2 (25/09/2024)

Áreas de estudo

Áreas	Descrição	Como funciona	Exemplo
Associative Reinforcement Learning	Combina aprendizado supervisionado e automação estocástica para associar padrões de entrada com ações.	Interage com o ambiente em um ciclo fechado, associando entradas a ações adequadas com base em recompensas e punições.	Associar padrões de entrada com ações adequadas em sistemas de controle automatizado.

Deep Reinforcement Learning (DRL)	Utiliza redes neurais profundas para aprendizado em ambientes complexos sem necessidade de definir estados explicitamente.	Usa uma rede neural profunda para extrair características dos dados brutos, como imagens, e aprender diretamente a partir delas.	Jogos de ATARI pela Google DeepMind, onde o agente aprende a jogar diretamente dos pixels da tela.
Adversarial Deep Reinforcement Learning	Foca em vulnerabilidades em políticas de RL profundo, revelando fragilidades por meio de manipulações adversárias.	Adiciona pequenas perturbações às entradas ou manipula o ambiente para explorar vulnerabilidades em políticas aprendidas.	Exploração de fraquezas em RL profundo em veículos autônomos ou sistemas de segurança.
Fuzzy Reinforcement Learning (FRL)	Introduz lógica fuzzy em RL para aproximar funções de valor contínuas e gerar regras mais flexíveis e interpretáveis.	Usa regras fuzzy no formato IF-THEN para criar aproximações contínuas da função de valor estado-ação.	Controle robótico que necessita de tomadas de decisão em ambientes incertos e contínuos.
Inverse Reinforcement Learning (IRL)	Inicia sem uma função de recompensa e a infere observando o comportamento de especialistas para imitar decisões.	Observa e imita o comportamento de um especialista, inferindo a função de recompensa otimizada pelo especialista.	Robótica e veículos autônomos onde o comportamento de um humano especialista é observado para imitar decisões.
Safe Reinforcement Learning (SRL)	Aprende políticas que maximizam recompensas enquanto respeitam restrições de segurança e mitigam riscos.	Maximiza a expectativa de retorno, levando em conta restrições de segurança e otimização de risco, como o CVaR.	Otimização de controle em sistemas robóticos onde a segurança é essencial, evitando ações perigosas.

Abordagens de aprendizado por reforço:

Abordagem	Descrição	Exemplo	Principais Algoritmos
Aprendizado por Reforço Baseado em Valor (Value-Based RL)	O agente aprende uma função de valor que estima o quão bom é estar em um estado ou realizar uma ação.	Q-Learning para aprendizado de uma função de valor associada a um jogo de tabuleiro.	Q-Learning, SARSA
Aprendizado por Reforço Baseado em Política (Policy-Based RL)	O agente aprende diretamente uma política que mapeia estados para ações.	REINFORCE para ajustar uma política diretamente em tarefas de controle contínuo.	REINFORCE, Proximal Policy Optimization (PPO)
Métodos Baseados em Ator-Crítico (Actor-Critic Methods)	Combina Value-Based e Policy-Based RL, com um ator que escolhe ações e um crítico que avalia as ações.	A2C para controlar robôs com uma política de ator-crítico.	A2C (Advantage Actor-Critic), DDPG (Deep Deterministic Policy Gradient)
Aprendizado por Reforço Multi-Agente (Multi-Agent RL)	Vários agentes interagem em um ambiente compartilhado, podendo cooperar, competir ou coexistir.	Simulações de tráfego onde múltiplos agentes aprendem a gerenciar o fluxo de veículos.	MARL (Multi-Agent Reinforcement Learning)
Deep Reinforcement Learning (DRL)	Utiliza redes neurais profundas para aprender em ambientes de alta dimensionalidade.	Deep Q-Network (DQN) para jogar jogos ATARI utilizando redes neurais profundas.	Deep Q-Networks (DQN)
Aprendizado por Reforço Hierárquico (Hierarchical RL)	Divide o problema de RL em subproblemas menores com políticas de alto e baixo nível.	Hierarchical-DQN para dividir a tarefa de um robô em múltiplos níveis de abstração.	Hierarchical-DQN
Model-Based RL (Aprendizado por Reforço Baseado em Modelos)	O agente constrói um modelo do ambiente e usa-o para prever e planejar ações futuras.	Model Predictive Control (MPC) para controle de sistemas dinâmicos.	Model Predictive Control (MPC)
Model-Free RL (Aprendizado por Reforço Livre de Modelos)	O agente aprende diretamente com interações, sem tentar modelar o ambiente.	Q-Learning e SARSA em ambientes de pequena escala sem modelagem do ambiente.	Q-Learning, SARSA

Artigos	Descrição	Proposta
Building Machines That Learn and Think Like People (2016) - Lake et Al.	Explora como construir sistemas de IA que aprendam e pensem de maneira mais semelhante aos humanos, analisando os avanços da IA e comparando com o aprendizado humano.	O artigo propõe que para as máquinas atingirem uma inteligência verdadeiramente semelhante à humana, elas precisam aprender modelos causais do mundo, ter uma base de teorias intuitivas de física e psicologia, e utilizar a composicionalidade para adquirir e generalizar rapidamente o conhecimento. Ele sugere a combinação de redes neurais com modelos cognitivos mais estruturados.
Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning (2021) - Fuchs et al.	O artigo apresenta um sistema de Aprendizado por Reforço Profundo (Deep Reinforcement Learning - DRL) para condução autônoma de veículos no simulador de corridas Gran Turismo Sport. O sistema supera tanto a IA padrão do jogo quanto os melhores tempos registrados por pilotos humanos.	Combinação de uma rede neural com uma função de recompensa baseada no progresso do curso para controlar veículos no limite de sua aderência e dinâmica física. O sistema atinge um desempenho super-humano sem a necessidade de planejamento explícito de trajetórias, o que sugere que o Aprendizado por Reforço pode ser eficaz no controle de veículos em condições extremas.

Benchmarking Safe Deep Reinforcement Learning in Aquatic Navigation (2021) - Marchesini et Al.	Propõe um ambiente de benchmark para o Aprendizado por Reforço Profundo Seguro (Safe DRL), focado na navegação aquática. O trabalho analisa o comportamento de redes treinadas para evitar situações perigosas, como colisões, utilizando drones aquáticos.	- Combinação de métodos baseados em gradiente e livres de gradiente com um crossover para melhorar a eficiência de amostragem, além de estratégia de verificação baseada em análise intervalar para quantificar e verificar as configurações de comportamento seguras, representando um avanço na avaliação de segurança em tarefas de navegação aquática
A Super-human Vision-based Reinforcement Learning Agent for Autonomous Racing in Gran Turismo (2024) - Vasco et Al.	Apresenta um agente de Aprendizado por Reforço baseado em visão capaz de superar pilotos humanos no jogo de simulação de corrida Gran Turismo 7. O agente utiliza uma câmera egocêntrica e dados internos do carro para executar a corrida, superando pilotos humanos em várias pistas.	Propõe um agente que utiliza exclusivamente entradas locais (visão e dados de sensores embarcados) durante a execução, sem informações globais. O treinamento assimétrico do ator-crítico, que usa recursos globais apenas durante o treinamento, é um avanço importante, permitindo que o agente atue apenas com entradas locais.
Aquatic Navigation: A Challenging Benchmark for Deep Reinforcement Learning (2024) - Corsi et Al.	Este artigo apresenta um benchmark desafiador para o Aprendizado por Reforço Profundo (Deep Reinforcement Learning - DRL) em ambientes de navegação aquática, simulando condições difíceis de ambientes subaquáticos e de superfície.	Desenvolvimento de um ambiente de simulação que integra realismo e flexibilidade para a navegação autônoma, incluindo forças de correntes marítimas. O benchmark também aborda aspectos de segurança, explorando funções de recompensa customizadas para evitar colisões e melhorar a navegação em condições imprevisíveis.

Diversity-Enhanced Reinforcement Learning in Complex Multi-Agent Systems (2024) - Kumar et Al.	O artigo propõe uma abordagem de aprendizado por reforço aprimorado pela diversidade em sistemas multiagente complexos, explorando a utilização de políticas diversificadas para otimizar a interação e coordenação entre agentes em ambientes dinâmicos.	Introdução de um mecanismo que aumenta a diversidade estratégica entre agentes, melhorando a capacidade do sistema de explorar e encontrar soluções eficientes em cenários complexos. A abordagem resulta em uma maior robustez e flexibilidade nos algoritmos de aprendizado por reforço aplicados a contextos de multiagentes.
--	---	--

Referências:

CORSI, Davide; CAMPONOGARA, Davide; FARINELLI, Alessandro. *Aquatic Navigation: A Challenging Benchmark for Deep Reinforcement Learning*. RLJ | RLC 2024, 2024. Disponível em: https://github.com/dadecampo/aquatic_navigation_envs. Acesso em: 18 set 2024.

FUCHS, Florian; SONG, Yunlong; KAUFMANN, Elia; SCARAMUZZA, Davide; DÜRR, Peter. Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning. *IEEE Robotics and Automation Letters*, v. 6, n. 2, p. 1-8, fev. 2021. Disponível em: <https://arxiv.org/abs/2112.10593>. Acesso em: 21 set. 2024

KUMAR, Rajesh; SINGH, Shweta; TAYLOR, Martin G. Diversity-Enhanced Reinforcement Learning in Complex Multi-Agent Systems. RLJ | RLC 2024, 2024.

LAKE, B. M.; ULLMAN, T. D.; TENENBAUM, J. B.; GERSHMAN, S. J. Building Machines That Learn and Think Like People. *Behavioral and Brain Sciences*, 2016.

MARCHESINI, Enrico; CORSI, Davide; FARINELLI, Alessandro. Benchmarking Safe Deep Reinforcement Learning in Aquatic Navigation. 2021. Disponível em: <https://arxiv.org/abs/2112.10593v1>. Acesso em: 20 set. 2024.

VASCO, Miguel; SENO, Takuma; KAWAMOTO, Kenta; SUBRAMANIAN, Kaushik; WURMAN, Peter R.; STONE, Peter. A Super-human Vision-based Reinforcement

Learning Agent for Autonomous Racing in Gran Turismo. RLC 2024, 2024.
Disponível em: <https://arxiv.org/abs/2406.12563v1>. Acesso em: 20 set. 2024.

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: 3 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Fabrycio Leite Nakano Almada

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

☰ Reinforcement Learning: até 03/10

- Resumo de artigos nos campos: Adversariais e NLP
 - [Adversarial Reinforcement Learning under Partial Observability in Autonomous Computer Network Defence](#)
 - Double Deep Q-Networks (DDQN)
 - Neural Episodic Control to Deep Q-Network (NEC2DQN ou N2D)
 - [Adversarial Deep Reinforcement Learning for Cyber Security in Software Defined Networks](#)
 - Double Deep Q-Networks (DDQN)
 - Asynchronous Advantage Actor-Critic (A3C)
 - [Deep Reinforcement Learning from Human Preferences](#)
 - Advantage Actor-Critic (A2C) - Jogos
 - Trust Region Policy Optimization - Robótica
 - [WebGPT: Browser-assisted question-answering with human feedback](#)
 - Behavior Cloning (BC)
 - Reward Modeling (RM)
 - Rejection Sampling
 - Proximal Policy Optimization (PPO)
- Estudo da Unidade 1 de: [Introduction to Deep Reinforcement Learning - Hugging Face Deep RL Course](#)
 - Hands on: [Deep Reinforcement Learning - Introduction.ipynb](#)
 - Upload do resultado no <https://huggingface.co/fabrycionakano/ppo-LunarLander-v2>

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

- Leitura do artigo: Exploration in Deep Reinforcement Learning: A survey
- Unidade 2 de [Introduction to Deep Reinforcement Learning - Hugging Face Deep RL Course](#)

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

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go! ▾

- Tabela dos Artigos do GATE 3 (03/10/2024)

Artigos	Descrição	Proposta
Adversarial Reinforcement Learning under Partial Observability in Autonomous Computer Network Defence	O estudo explora o impacto de ataques causativos em algoritmos de aprendizado por reforço (RL) para a defesa autônoma de redes sob a limitação de observabilidade parcial.	Os autores propõem um método de defesa de inversão que aplica uma perturbação oposta à introduzida por um atacante, buscando reverter os efeitos de ataques adversários em RL. O método de defesa proposto demonstrou reduzir com sucesso o impacto dos ataques adversários, sem prejudicar significativamente o treinamento do agente em cenários sem ataques.

Adversarial Deep Reinforcement Learning for Cyber Security in Software Defined Networks	O artigo investiga o uso de <i>Deep Reinforcement Learning</i> (DRL) adversarial para segurança autônoma em redes definidas por software (SDN), analisando ataques cibernéticos.	Compara dois algoritmos, Double Deep Q-Network (DDQN) e Neural Episodic Control to Deep Q-Network (N2D), para avaliar o impacto de ataques causativos em ambientes SDN. A pesquisa mostrou que, com pequenas mudanças nos parâmetros, o N2D teve melhor desempenho como atacante, e os modelos são robustos o suficiente para defender redes SDN.
Deep Reinforcement Learning from Human Preferences	Explora como comunicar objetivos complexos a sistemas de RL por meio de preferências humanas.	Propõe usar comparações de trajetórias humanas para gerar uma função de recompensa aprendida. A abordagem reduz significativamente o feedback humano necessário para RL e supera RL clássico em tarefas complexas.
WebGPT: Browser-assisted question-answering with human feedback	Introduz o WebGPT , um modelo de IA que responde perguntas usando uma interface de navegação na web, sendo treinado com feedback humano para melhorar a qualidade das respostas. Ele utiliza o GPT-3 como base e aprimora as respostas de longo formato (long-form question answering) utilizando aprendizado por imitação e aprendizado por reforço.	O WebGPT permite que o modelo navegue na web em uma interface de texto para buscar informações e coletar referências em tempo real. As respostas são geradas com base no conteúdo encontrado durante a navegação, e os rótulos humanos ajudam a otimizar a precisão e a utilidade das respostas por meio de aprendizado por reforço e amostragem por rejeição. O WebGPT demonstrou ser mais eficaz que humanos na tarefa de responder perguntas de longo formato do conjunto ELI5, com respostas preferidas em 56% das vezes em comparação com as respostas humanas. Ele também superou os modelos anteriores de GPT-3 na geração de respostas verdadeiras e informativas, destacando o potencial do uso de navegação web assistida.

Análise dos Artigos

- **Adversarial Deep Reinforcement Learning for Cyber Security in Software Defined Networks**

O artigo "*Adversarial Deep Reinforcement Learning for Cyber Security in Software Defined Networks*" (Aprendizado Profundo por Reforço Adversarial para Cibersegurança em Redes Definidas por Software) explora como o aprendizado por reforço profundo (DRL) pode ser utilizado para melhorar a cibersegurança em Redes Definidas por Software (SDN). A pesquisa foca na aplicação de técnicas de aprendizado adversarial para fortalecer os mecanismos de defesa em SDN, simulando um ambiente semelhante a um jogo onde atacantes e defensores interagem. Dois algoritmos principais são usados no estudo: Double Deep Q-Networks (DDQN) e Neural Episodic Control to Deep Q-Network (NEC2DQN ou N2D).

Conceitos-chave:

1. Redes Definidas por Software (SDN): SDN é uma arquitetura de rede de três camadas que permite a separação do plano de controle do plano de dados. Isso permite uma gestão de rede eficiente e programável, o que é particularmente útil para adaptar-se às ameaças cibernéticas modernas.
2. Aprendizado por Reforço Profundo (DRL): DRL é aplicado para criar agentes autônomos que aprendem interagindo com o ambiente, com o objetivo de maximizar recompensas acumuladas. Neste contexto, o objetivo é treinar agentes para defenderem a rede contra ciberataques.
3. Aprendizado Adversarial: Esta técnica envolve o uso de agentes adversários (atacantes) para enganar ou corromper o processo de aprendizado de um modelo. No artigo, o aprendizado adversarial é implementado por meio de ataques de envenenamento de dados, que modificam as entradas dos agentes de aprendizado, criando um ambiente mais desafiador para os defensores.

Metodologia:

O estudo compara o desempenho dos algoritmos DDQN e NEC2DQN em um ambiente que simula um jogo de captura da bandeira. O atacante e o defensor se revezam usando esses

algoritmos, e seu desempenho é avaliado em condições normais e com um ataque adversarial ativo (causativo). O atacante, neste caso, tem acesso total ao modelo do defensor (cenário de caixa branca) e manipula o processo de aprendizado injetando falsos positivos e falsos negativos.

Resultados:

O estudo descobriu que:

- A introdução do aprendizado adversarial tornou os atacantes mais eficazes em alguns cenários. Com pequenos ajustes nos parâmetros, os atacantes mostraram melhorias em seu desempenho e na capacidade de vencer jogos.
- Defensores que usavam o algoritmo NEC2DQN tiveram um desempenho melhor do que os que usavam DDQN, mostrando maior resistência a ataques adversariais.
- Em cenários com ataques causativos ativos, embora o desempenho do defensor tenha sido prejudicado, houve aumentos notáveis na quantidade de dados de treinamento disponíveis para os defensores, o que pode melhorar a robustez dos modelos no futuro.

Conclusão:

O artigo destaca que, embora SDN forneça uma plataforma flexível e programável para a gestão de redes, também cria novas vulnerabilidades que podem ser exploradas por agentes adversários. A pesquisa demonstra o potencial de usar aprendizado adversarial para treinar sistemas de cibersegurança mais resilientes em SDN. Também sugere trabalhos futuros focando em topologias de rede mais complexas e na inclusão de observabilidade parcial para os atacantes, com o objetivo de melhorar ainda mais a robustez dos mecanismos de defesa.

Implicações:

Este estudo mostra que o aprendizado adversarial pode ser tanto um desafio quanto uma oportunidade na cibersegurança, oferecendo insights valiosos sobre como os sistemas de defesa podem ser reforçados contra atacantes inteligentes e adaptativos.

- **Adversarial Reinforcement Learning under Partial Observability in Autonomous Computer Network Defence**

Conceitos principais:

1. **Aprendizado por Reforço (RL):** O RL envolve agentes que tomam decisões com base em interações com o ambiente, buscando maximizar recompensas acumuladas. O estudo utiliza dois algoritmos populares de RL: Double Deep Q-Networks (DDQN) e Asynchronous Advantage Actor-Critic (A3C).
2. **Ataques Causativos:** Este tipo de ataque visa manipular o processo de treinamento dos agentes de RL. O objetivo do atacante é distorcer o estado percebido pelo agente, levando-o a tomar decisões subótimas, o que compromete a rede.
3. **Observabilidade Parcial:** No estudo, os atacantes têm acesso limitado ao estado da rede. Eles só podem visualizar parte da topologia, enquanto os defensores possuem uma visão completa. Essa configuração reflete cenários reais, onde os atacantes muitas vezes não têm uma visão completa da infraestrutura alvo.

Metodologia:

O estudo simula um ambiente de defesa de rede, onde os defensores usam RL para proteger servidores críticos de serem comprometidos por atacantes. São explorados dois cenários de ataque:

- No primeiro, o atacante pode observar cerca de um terço da rede.
- No segundo, o atacante tem acesso a metade da rede.

Em ambos os cenários, os atacantes usam técnicas de envenenamento de dados para enganar os agentes defensores, alterando a percepção do estado da rede.

Resultados:

1. **Eficácia dos Ataques:** Os ataques adversariais foram eficazes em comprometer os servidores críticos ou reduzir o número de nós preservados na rede. Mesmo com observabilidade parcial, os atacantes conseguiram manipular o processo de aprendizado do RL.
2. **Defesa por Inversão:** Para mitigar esses ataques, os autores propõem um método de defesa por inversão. Essa técnica tenta reverter as manipulações introduzidas pelo atacante nos estados observados. Os experimentos mostraram que essa abordagem

pode reduzir significativamente o impacto dos ataques, sem comprometer o desempenho do agente de RL em cenários normais (sem ataque).

Conclusão:

O artigo conclui que agentes de RL podem ser vulneráveis a ataques adversariais mesmo quando o atacante tem acesso limitado à rede. No entanto, a defesa por inversão proposta foi capaz de mitigar esses ataques. O estudo sugere que futuros trabalhos explorem cenários com observabilidade parcial também para o defensor, além de ataques mais sofisticados.

Implicações:

O trabalho contribui para o campo da cibersegurança autônoma, mostrando como ataques adversariais podem comprometer a segurança de redes e como contramedidas podem ser aplicadas para proteger sistemas em cenários de defesa autônoma.

- Deep Reinforcement Learning from Human Preferences

1. Problema

O artigo aborda o problema de como comunicar metas complexas a sistemas de aprendizado por reforço (RL) que interagem com ambientes do mundo real. Em muitas tarefas, o objetivo desejado é difícil de formalizar como uma função de recompensa simples. A solução proposta busca reduzir a necessidade de um feedback humano direto e extenso, que, para sistemas de RL tradicionais, pode se tornar extremamente caro em termos de tempo e recursos.

2. Abordagem

Treinar agentes de RL com base em preferências humanas em relação a segmentos de trajetórias, em vez de utilizar funções de recompensa explícitas. Isso é feito ao solicitar que humanos comparem pares de trajetórias de agentes (segmentos de ações e observações) e escolham qual preferem. Com esse feedback, o sistema aprende uma função de recompensa que é, então, otimizada usando RL tradicional.

Etapas Principais:

1. **Interação com o ambiente:** O agente coleta trajetórias (sequências de observações e ações) enquanto interage com o ambiente.
2. **Coleta de feedback humano:** Pares de segmentos de trajetórias são selecionados e apresentados a humanos, que expressam preferência sobre qual segmento é melhor.
3. **Ajuste da função de recompensa:** O feedback humano é usado para treinar uma função de recompensa que aproxima as preferências humanas.
4. **Otimização da política:** O agente usa RL para maximizar a soma dos valores preditos pela função de recompensa aprendida.

3. Domínios de Aplicação

Os experimentos são realizados em dois domínios principais:

- **Jogos de Atari** no Arcade Learning Environment: usados para testar a eficácia do método em tarefas complexas e baseadas em simulações.
- **Robótica simulada** no MuJoCo: usado para avaliar o método em tarefas de controle contínuo e com alta dimensionalidade, como movimentos de robôs.

4. Resultados

Os resultados demonstram que é possível aprender comportamentos complexos a partir de um feedback humano mínimo, atingindo resultados quase equivalentes aos obtidos por algoritmos de RL que têm acesso à função de recompensa real. Em alguns casos, como em tarefas com robôs simulados, o método baseado em preferências humanas supera o RL tradicional.

Exemplos Notáveis:

- **Robótica:** O agente foi capaz de realizar comportamentos complexos, como fazer backflips ou andar em uma perna, com menos de uma hora de feedback humano.
- **Atari:** Em jogos como Pong e BeamRider, o feedback humano gerou desempenhos comparáveis ao RL tradicional, mesmo sem a função de recompensa original.

5. Comparação com Trabalhos Relacionados

O artigo menciona diversos trabalhos anteriores que lidam com aprendizado por reforço a partir de preferências ou feedback humano. A inovação principal do trabalho é escalar essas abordagens para sistemas de RL profundos (deep RL), em domínios mais complexos e com comportamentos mais sofisticados. O artigo também compara com trabalhos anteriores em

RL inverso e aprendizado por imitação, que dependem de demonstrações humanas, algo que não é sempre viável.

6. Contribuições Principais

- **Redução do feedback humano necessário:** O método reduz a quantidade de feedback humano em ordens de grandeza, tornando a interação com sistemas de RL mais prática para tarefas complexas.
- **Aprendizado de comportamentos complexos a partir de preferências:** O artigo mostra que é possível aprender comportamentos não triviais, como dirigir no fluxo de tráfego, a partir de menos de 1% das interações do agente com o ambiente recebendo feedback humano.
- **Flexibilidade:** O método é flexível o suficiente para ser aplicado em várias tarefas, desde jogos de Atari até controle de robôs simulados.

7. Limitações e Discussões

- **Dependência de preferências humanas:** A qualidade do comportamento aprendido depende da consistência do feedback humano. O artigo sugere que, em alguns casos, como em Qbert, essa abordagem pode falhar quando os cliques mostrados aos humanos são difíceis de avaliar.
- **Ajustes manuais em alguns parâmetros:** Algumas configurações de hiperparâmetros, como a regularização na função de recompensa, exigem ajustes cuidadosos para cada tarefa.

8. Conclusão

O trabalho representa um avanço significativo no uso de feedback humano para guiar agentes de RL. Ele demonstra que, ao invés de treinar com funções de recompensa explícitas, os sistemas podem aprender de preferências humanas, expandindo o leque de tarefas que podem ser resolvidas por RL. A abordagem apresentada é promissora para reduzir o custo de supervisão humana e tornar o aprendizado por reforço aplicável a problemas reais mais complexos.

- **CodeRL: Mastering Code Generation through Pretrained Models and Deep Reinforcement Learning**

Descrição do Problema: A síntese de código, que consiste em gerar programas a partir de descrições de problemas, é uma tarefa desafiadora devido à necessidade de correção funcional dos códigos gerados. Modelos pré-treinados de linguagem, como GPT-3, têm sido

usados para gerar código, mas enfrentam limitações significativas ao depender apenas da predição do próximo token. Essas limitações levam a erros acumulados que prejudicam a correção funcional dos programas.

Proposta do CodeRL: O **CodeRL** combina modelos de linguagem pré-treinados (como os usados para completar código) com aprendizado por reforço profundo (RL) para melhorar a geração de código. O modelo utiliza uma arquitetura de ator-crítico, onde:

- **Ator (Actor Network):** Um modelo pré-treinado gera código com base em uma descrição textual.
- **Crítico (Critic Network):** Avalia a correção funcional dos códigos gerados, fornecendo feedback com base nos resultados de testes unitários. O feedback obtido desses testes unitários é usado para otimizar o modelo de ator, regenerando programas com base nas correções apontadas.

Métodos Utilizados:

1. **Testes Unitários como Feedback:** Testes unitários são usados para verificar a correção funcional dos programas gerados. Em vez de depender apenas de tokens preditivos, o CodeRL utiliza o sucesso ou falha desses testes como sinal de recompensa para melhorar o modelo.
2. **Integração com Aprendizado por Reforço (RL):** A função de recompensa no aprendizado por reforço é projetada para incentivar o ator a gerar códigos funcionais com base no feedback dos testes.

Resultados Experimentais: O CodeRL foi testado em benchmarks como o **APPS** (um conjunto de dados para avaliação de geração de código) e o **MBPP** (um benchmark para pequenos problemas de programação). O modelo alcançou novos resultados de estado da arte, superando abordagens anteriores, como o Codex da OpenAI e o AlphaCode do DeepMind. O desempenho foi particularmente notável em tarefas que exigiam correção funcional de código.

Conclusão: O CodeRL demonstra que a combinação de aprendizado por reforço com modelos de linguagem pré-treinados melhora significativamente a correção funcional na geração de código. Além disso, o uso de testes unitários como parte do processo de feedback fornece uma maneira prática e eficiente de regenerar e corrigir o código. O framework é promissor para aplicações futuras em síntese de código e outras tarefas de programação automatizada.

- **WebGPT: Browser-assisted Question-answering with Human Feedback**

Descrição do Problema: Responder perguntas de forma precisa e detalhada, especialmente perguntas de formato longo (long-form question-answering, LFQA), continua a ser um desafio em sistemas de NLP. Modelos como GPT-3 podem gerar respostas, mas frequentemente carecem de precisão factual e de habilidade de citar fontes confiáveis. Além disso, muitos modelos atuais dependem de informações pré-treinadas, o que pode levar à geração de respostas desatualizadas ou incorretas.

Proposta do WebGPT: O **WebGPT** foi projetado para superar essas limitações combinando a capacidade de geração de texto do GPT-3 com a navegação em uma interface web. Esse modelo navega na web, utilizando um ambiente de busca e citação de fontes em tempo real para coletar informações e gerar respostas. A abordagem inovadora inclui:

- **Navegação Web Textual:** O modelo interage com uma interface de navegação em texto, realizando buscas, clicando em links e citando fontes durante o processo de geração de resposta.
- **Feedback Humano:** As respostas geradas pelo modelo são avaliadas por humanos, que fornecem feedback sobre a precisão, utilidade e coerência das respostas. Esse feedback é utilizado para treinar o modelo por meio de aprendizado por imitação (behavior cloning) e aprendizado por reforço (RL).

Métodos Utilizados:

1. **Ambiente de Navegação na Web:** O modelo utiliza a API do Bing para realizar buscas e simula uma experiência de navegação em texto. Ele pode percorrer páginas da web, extrair citações e usá-las para compor respostas completas.
2. **Treinamento com Feedback Humano:** O WebGPT é treinado em um ambiente que permite aos humanos fornecer demonstrações de como responder a perguntas usando navegação web, e também comparar respostas geradas pelo modelo, avaliando a precisão factual e a coerência.
3. **Amostragem por Rejeição (Rejection Sampling):** O WebGPT utiliza amostragem por rejeição para escolher as melhores respostas a partir de várias tentativas, otimizando as respostas com base em um modelo de recompensa treinado para prever preferências humanas.

Resultados Experimentais:

1. **Desempenho em ELI5:** O WebGPT foi testado no conjunto de dados ELI5 (Explain Like I'm Five), que contém perguntas abertas e respostas detalhadas. As respostas geradas pelo WebGPT foram preferidas pelos avaliadores humanos em 56% dos

casos em comparação com respostas humanas, e em 69% das vezes em comparação com as respostas mais bem votadas do Reddit.

2. **TruthfulQA:** O WebGPT também foi testado em um conjunto de perguntas adversariais, onde obteve resultados superiores aos modelos GPT-3, sendo capaz de fornecer respostas verdadeiras e informativas com mais frequência.

Conclusão: O WebGPT mostrou que a combinação de navegação web assistida com feedback humano e aprendizado por reforço pode superar as limitações dos modelos de geração de texto atuais, oferecendo respostas mais precisas e confiáveis. Ele se destaca particularmente em cenários onde é necessária a citação de fontes ou quando as respostas precisam ser baseadas em informações atualizadas. No entanto, desafios permanecem, como a possibilidade de o modelo exagerar sua confiança em certas fontes ou reforçar vieses.

- **Adversarial Reinforcement Learning under Partial Observability in Autonomous Computer Network Defence**

Descrição do Problema: O artigo explora a vulnerabilidade de agentes de aprendizado por reforço (RL) a ataques adversariais, com foco em ambientes de defesa cibernética autônoma em redes de computadores sob observabilidade parcial. Em sistemas autônomos de defesa de rede, agentes RL são usados para proteger servidores críticos de ataques. No entanto, esses agentes podem ser manipulados durante o treinamento por adversários que perturbam os estados observados.

Contribuições Principais:

1. **Ataques Causativos em RL:** O artigo mostra que agentes de RL podem ser manipulados por ataques causativos (durante o treinamento) mesmo em cenários de observação parcial, onde o atacante tem acesso limitado ao ambiente.
2. **Método de Defesa Inverso:** A solução proposta é um mecanismo de defesa que tenta reverter as perturbações adversariais aplicadas ao estado do ambiente. Esse método é eficaz em mitigar ataques sem afetar significativamente o desempenho do agente quando não há ataque.
3. **Cenários de Rede:** O artigo utiliza redes de computadores simuladas com 100 nós e 172 links, onde o atacante tenta comprometer servidores críticos e o defensor, com observação total, treina agentes RL para proteger o sistema.

Resultados Experimentais:

1. **Ataques sob Observação Parcial:** Demonstrou-se que, em cenários onde o atacante só tem visibilidade parcial da rede, ele ainda pode realizar ataques eficazes que desviam os agentes de RL de suas políticas ótimas.

2. **Eficácia da Defesa:** O mecanismo de defesa inverso proposto demonstrou reduzir substancialmente os impactos dos ataques causativos, garantindo que os agentes de RL ainda possam aprender e tomar decisões adequadas, mesmo sob ataque.
3. **Comparação entre Algoritmos RL:** Foram testados dois algoritmos, Double Deep Q-Network (DDQN) e Asynchronous Advantage Actor-Critic (A3C), e ambos se mostraram suscetíveis a ataques. A defesa inversa se mostrou eficaz contra ambos os métodos.

Conclusão: O artigo conclui que, mesmo com observação parcial, ataques adversariais podem comprometer o treinamento de agentes RL em defesa de rede, mas o método de defesa inversa proposto é uma solução viável. O estudo sugere direções futuras para investigar mais cenários de observabilidade parcial e ataques mais sofisticados.

Extras

[An Overview of Advancements in Deep Reinforcement Learning \(Deep RL\) - MarkTechPost](#)

Descrição Geral: O artigo aborda o progresso do aprendizado por reforço profundo (Deep RL), que combina aprendizado por reforço (RL) com redes neurais profundas, permitindo que máquinas alcancem desempenhos em nível humano ou até superem humanos em várias tarefas complexas, como jogos de múltiplos jogadores.

História e Fundamentos: Deep RL surgiu como uma solução para superar as limitações do RL tradicional, que dependia de estados totalmente observáveis e de recursos com baixa dimensionalidade. Um marco importante foi o programa TD-Gammon (1992), que usava redes neurais para jogar backgammon. Em 2013, a DeepMind aplicou Deep RL a jogos de Atari, resultando no algoritmo Deep Q-Network (DQN), que superou jogadores humanos em alguns jogos.

Arquitetura de Deep RL: Deep RL envolve a interação entre um agente e um ambiente, onde o agente aprende por tentativa e erro, recebendo recompensas. As redes neurais profundas são utilizadas para lidar com espaços de observação de alta dimensão e tomar decisões complexas com base em entradas sensoriais brutas.

Algoritmos Importantes:

1. **Deep Q-Network (DQN):** Um dos primeiros algoritmos de sucesso que aplicou aprendizado profundo ao RL, permitindo que agentes aprendessem políticas diretamente de entradas sensoriais brutas.

2. **Deep Deterministic Policy Gradient (DDPG):** Para espaços de ação contínuos, combinando gradientes de política determinísticos e aprendizado profundo.
3. **Proximal Policy Optimization (PPO):** Um método de gradiente de política introduzido para melhorar a estabilidade e eficiência do treinamento.
4. **Trust Region Policy Optimization (TRPO):** Enfoca a atualização de políticas com passos pequenos para evitar grandes mudanças que possam prejudicar o desempenho.
5. **Soft Actor-Critic (SAC):** Um algoritmo de ator-crítico que otimiza uma política estocástica, maximizando o retorno esperado e a entropia.

Conclusão: Deep RL tem mostrado sucesso em tarefas complexas, como jogos, graças a algoritmos como DQN, DDPG, PPO, TRPO e SAC. Esses avanços resolvem desafios relacionados a entradas de alta dimensão, controle contínuo e estabilidade, prometendo ainda mais avanços em várias áreas da IA.

[LLM reinforcement learning: What is Essential in 2024 ? | by Wiem Souai | Medium](#)

O artigo explora os conceitos essenciais e os avanços mais recentes no campo de aprendizado por reforço (RL) e sua aplicação em Grandes Modelos de Linguagem (LLMs). Com a crescente aplicação de RL, especialmente no contexto de LLMs, compreender seus princípios fundamentais tornou-se crucial.

Fundamentos do Aprendizado por Reforço:

- **Agente:** O agente é o responsável por interagir com o ambiente, tomando decisões com base nas observações.
- **Ambiente:** O sistema externo no qual o agente opera, fornecendo o contexto para as interações.
- **Ações:** As escolhas possíveis que o agente pode realizar dentro do ambiente.
- **Recompensas:** Feedback que o agente recebe ao tomar uma ação, moldando seu aprendizado.
- **Políticas:** Estratégias que guiam o agente nas tomadas de decisão.

Algoritmos de RL:

1. **Q-Learning:** Focado em estimar o valor das ações em estados específicos, convergindo gradualmente para uma política ótima.
2. **Deep Q-Network (DQN):** Integra redes neurais profundas ao RL para aproximar valores-Q, tornando viável lidar com espaços de estado de alta dimensão.

3. **Policy Gradient Methods:** Em vez de estimar valores-Q, esses métodos otimizam diretamente a política do agente.
4. **Processos de Decisão de Markov (MDPs):** Base matemática para modelar problemas de RL, capturando estados, ações, transições e recompensas.

Avanços Recentes em RL:

1. **Aprendizado por Reforço Profundo (DRL):** Integra redes neurais profundas, permitindo que RL lide com ambientes complexos de alta dimensão.
2. **RL Baseado em Modelos:** Melhora a eficiência de amostras ao utilizar modelos aprendidos ou simulados do ambiente para planejar e otimizar ações.
3. **Meta-RL:** Ensina agentes a aprender em ambientes dinâmicos, promovendo adaptação rápida a novas tarefas.
4. **RL Multi-Agente (MARL):** Expande o RL para cenários com múltiplos agentes interagindo de forma cooperativa ou competitiva.
5. **Transferência e Aprendizado Contínuo (TLRL):** Facilita a transferência de conhecimento entre tarefas, permitindo que agentes acumulem expertise ao longo do tempo.

RL para Grandes Modelos de Linguagem (LLMs) com Feedback Humano (RLHF):

A combinação de RL com feedback humano torna os LLMs mais alinhados com valores e preferências humanas. Aplicações incluem geração de texto, sistemas de diálogo, tradução, sumarização, resposta a perguntas e programação de computadores.

Exemplos de Produtos que Utilizam RL:

- **Scale AI:** Integra RL em aplicações de linguagem com feedback humano.
- **OpenAI ChatGPT:** Usa RL para melhorar as respostas textuais geradas.
- **Labelbox:** Software de rotulagem que utiliza RL para refinar modelos de LLM.
- **Hugging Face:** Oferece ferramentas como RL4LLMs, permitindo a modificação de LLMs com diferentes algoritmos de RL.
- **DeepMind AlphaStar:** Utiliza RL para dominar jogos de estratégia em tempo real.

Desafios no RL:

1. Eficiência de amostra.
2. Generalização e transferência de aprendizado.
3. Equilíbrio entre exploração e exploração.
4. Segurança e robustez dos agentes.

5. Escalabilidade e implementação no mundo real.

Conclusão:

O RL continua a ser um pilar fundamental no campo da IA, proporcionando capacidades de tomada de decisão autônoma em aplicações variadas. Apesar dos desafios, a pesquisa contínua promete desbloquear novos horizontes na IA, permitindo que sistemas inteligentes prosperem em ambientes dinâmicos.

Estudo da Unidade 1 de: [Introduction to Deep Reinforcement Learning - Hugging Face Deep RL Course](#)

🔗 Deep Reinforcement Learning - Introduction (Unit 1) - Fabrycio L N Almada.ipynb

Upload do resultado:

<https://huggingface.co/fabrycionakano/ppo-LunarLander-v2/tree/main>

APÊNDICE 4

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 out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Fabrycio Leite Nakano Almada

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

- Resumo inicial criado por mim  Reinforcement Learning 09/10
 - Benchmarks relevantes citados no paper:
 - Atari Games
 - VizDoom
 - Malmö
 - Mujoco
 - Compreensão dos fluxos principais da Exploração (Exploration):
 - Segurança
 - Eficiência
 - Desafios futuros:
 - Exploration-exploitation dilemma,
 - Noisy TV problems,
 - Transferability.
- Treino de um agente utilizando o Algoritmo PPO com o auxílio de um ambiente criado no ML-Agents (Unity)
 - Agente treinado no Huggingface: [Fabrycionakano/ppo-Huggy](https://huggingface.co/Fabrycionakano/ppo-Huggy)
 -  Unit 1 (bonus_train with huggy) - Fabrycio Leite Nakano Almada.ipynb

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

Reestruturar o resumo, incrementar com mais artigos, estudar e me familiarizar com os frameworks

(TensorFlow Agents, PyTorch RLlib, Stable Baselines3...) , além de estudar os ambientes de simulação

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

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: **Go!**

- Minha Adaptação da taxidermia apresentada no artigo: [Taxidermia - FLNA](#)

Resumo do Artigo: [Exploration in Deep Reinforcement Learning: A Survey](#)

1 - Introduction

Sparse reward problems: Reward is rare and without a clear link to previous actions, thousands of decisions might need to be made before the outcomes are visible.

- SRP are common in real world, ex:
 - During search and rescue missions, the reward is only given when an object is found
 - During delivery, the reward is only given when an object is delivered.

Techniques for exploration in reinforcement learning

- Random exploration
 - The agent decides what to do randomly regardless of its progress
 - most commonly-used technique of this type: e-greedy
 - e-greedy: uses the time decaying parameter to reduce exploration over time
 - often impractical due to large learning times

- has shown some impressive performance in Atari games , Mujoco simulator, controllers tuning, autonomous landing, self-driving cars and healthcare.
- Reward shaping
 - The designer 'artificially' imposes a reward more often.
 - ex: for search and rescue missions, agents can be given a negative reward every time they do not find the victim.
 - Is a challenging problem that is heavily dependent on the experience of the designer
 - Punishing the agent too much could lead to the agent not moving at all, while rewarding it too much may cause the agent to repeat certain actions infinitely.
- Exploration in RL still under development, but exploration algorithms has shown a significant gain in performance compared to non-exploratory algorithms
 - Diversity is all you need (DIYAN) improved on MuJoCo benchmarks
 - Random network distillation (RND) and pseudocounts were the first to score on difficult Montezuma's Revenge problem
 - Agent57 is the first to beat humans in all 57 Atari games
- The review focuses on exploratory approaches which fulfill at least one of the following criteria
 1. determines the exploration degree based on the agent's learning,
 2. actively decides to take certain actions in hopes of finding new outcomes, and
 3. motivates itself to continue exploring despite a lack of environmental rewards.

2 - Preliminaries

2.1. Introduction to Reinforcement Learning

2.1.1. Markov Decision Process

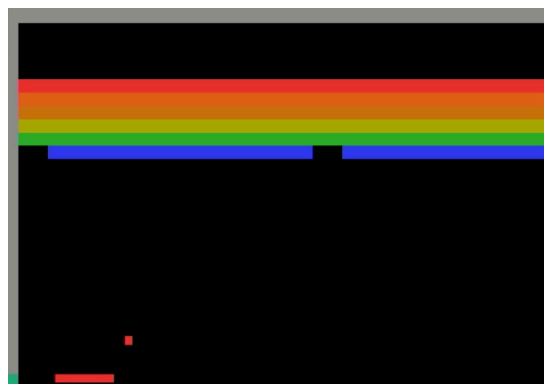
2.1.2. Value-Based Methods

2.1.3. Policy-Based Methods

2.2 Exploration

- Activity of searching and finding out about something
- In RL, "something" is a reward function and the "searching and finding" is an agent's attempt to try to maximise the reward function.
- Exploration is the most useful in problems in which a route to the actual solution (i.e. reward) is obstructed by the local minima (maxima) or areas of flat rewards.
- **discovering the true nature of rewards is challenging.**

- 2.3. Challenging Problems
 - Noisy-TV
 - the agent is stuck in exploring an infinite number of states which lead to no reward
 - Sparse Reward Problems
 - there is a long gap between an action and a reward (the reward is relatively rare).
- 2.4 Benchmarks
 - 2.4.1. Atari Games
 - A set of 57 Atari games combined under the Atari Learning Environment (ALE)
 - The state space is normally either images or random-access memory (RAM) snapshots.
 - The action space consists of five joystick actions (up, down, left, right, and action button).
 - split into two groups: easy (54 games) and difficult exploration (3 games)
 - Easy exploration: reward is relatively easy to find



Breakout for Atari 2600. Fonte: [Breakout - Atari 2600 Gameplay](#)

ALE: [Atari - Gymnasium Documentation](#)

- Hard exploration: reward is not often given, and the association between states and rewards is complex.



Pitfall for Atari 2600. Fonte: [AtariAge - Atari 2600 Screenshots - Pitfall! \(Activision\)](#)

- 2.4.2. VizDoom

- Benchmark based on the Doom game.
- The game has a first-person perspective and the image seen by the character is normally used as a state space.
- The action space is normally eight directional control and two action buttons (picking up key cards and opening doors).
- More actions can be added, if needed.
- Availability of easy-to-use tools for editing scenarios and low computational burden.



Doom. Fonte: [VDAIC 2017 @ CIG](#)

Vizdoom: [ViZDoom](#)

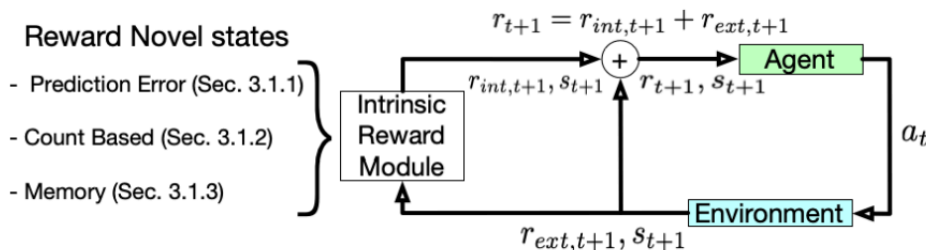
- 2.4.3 Malmo

- Benchmark based on the game Minecraft.
- It is also from the first-person perspective, and the image is the state space.

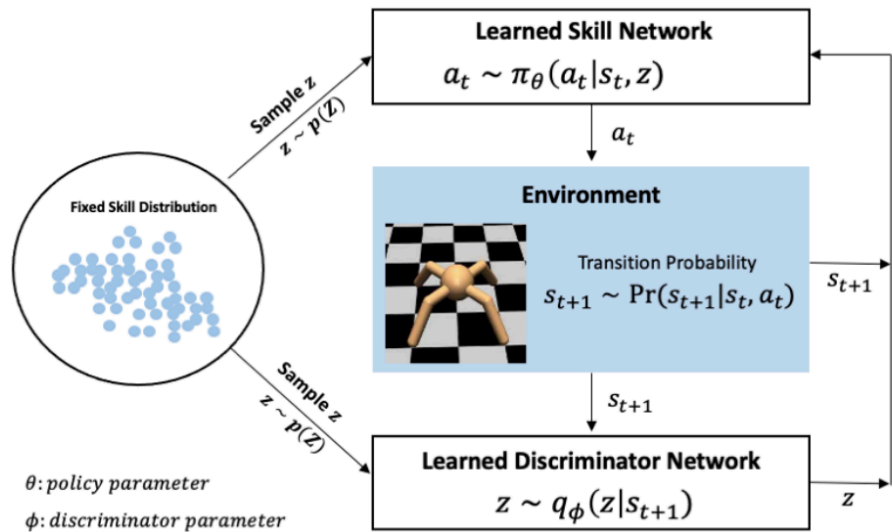
- The key advantage of Malmo is its flexibility in terms of the environment structure, domain size, custom scripts, and reward functions.
- Malmo: [GitHub - tsmatz/minecraft-rl-example: Applying Reinforcement Learning in Minecraft \(Project Malmo Tutorial\)](#)
- 2.4.4. Mujoco
 - Represents multi-joint dynamics with contact
 - Popular benchmark used for physics-based simulations.
 - Typically used to simulate walking robots.
 - Mujoco: [MuJoCo Documentation](#)

3 - Exploration in Reinforcement Learning:

- Can be split into two main streams:
 - Efficiency
 - Make exploration more sample efficient so that the agent can explore in as few steps as possible
 - Safe Exploration
 - Ensuring safety during exploration
- 3.1. Reward Novel States
 - Reward novel state approaches give agents a reward for discovering new states.
 - This reward is called an intrinsic reward

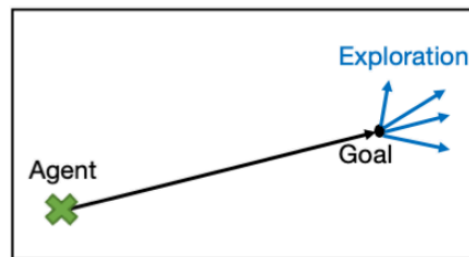


- 3.2. Reward Diverse Behaviours
 - The agent collects as many different experiences as possible
 - Exploration as an objective rather than a reward finding
 - Diversity is all you need (DIAYN)

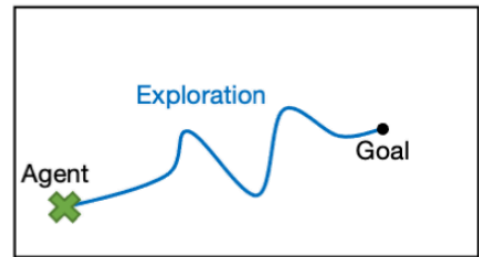


- 3.3. Goal-Based Methods

- The states of interest for exploration are used to guide the agent's exploration



Goals to Explore from (Sec. 3.3.1)



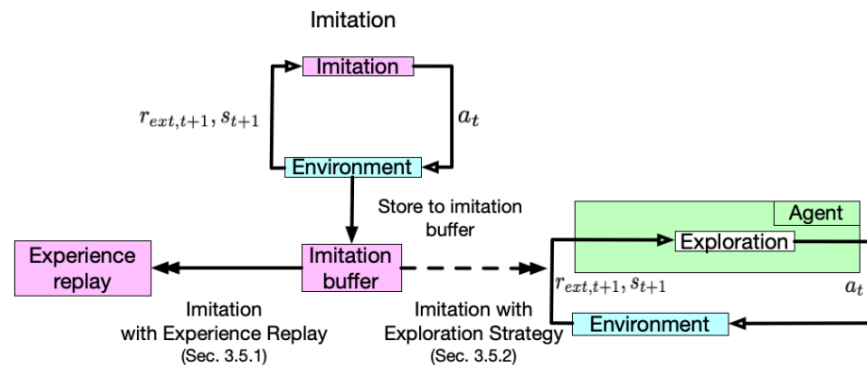
Exploratory Goals (Sec. 3.3.2)

- In goal-based methods, the agent's task is to reach a specific goal (or state). Then, this goal is explored using another exploration method or to generate an exploratory target goal
- The key concept is to guide agents directly to unexplored areas.

- 3.4. Probabilistic Methods

- The agent holds a probability over states, actions, values, rewards or their combination and chooses the next action based on that probability.
- Can be split into optimistic and uncertain methods
 - In optimistic methods, the estimation needs to depend on a reward, either implicitly or explicitly.
 - Then, the upper bound of the estimate is used to make the action.

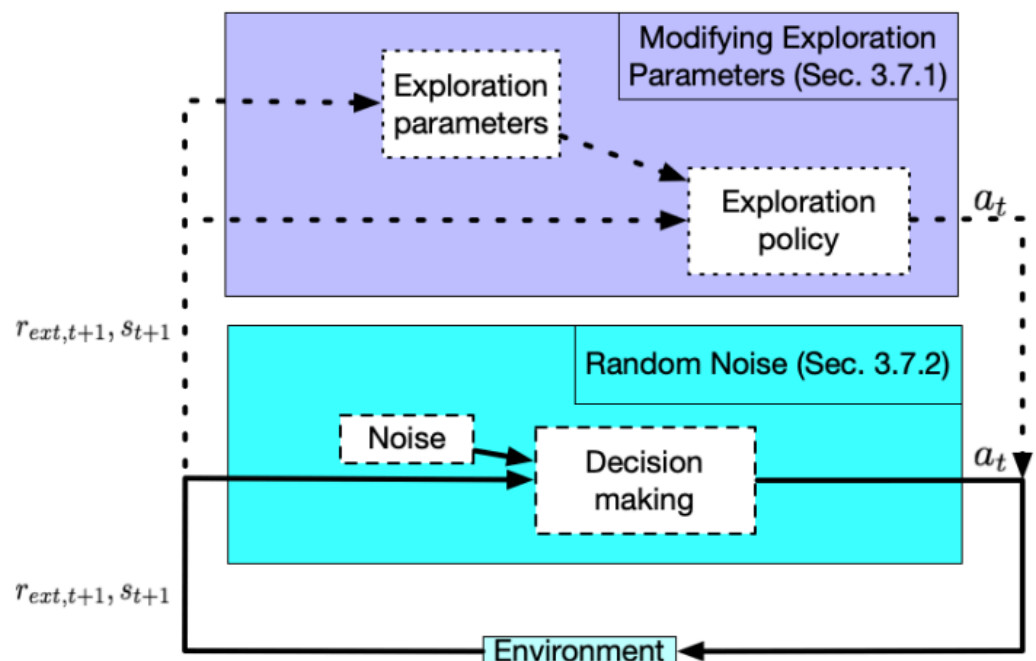
- In uncertainty-based methods, the estimate is the uncertainty about the environment, such as the value function and state prediction.
 - In the uncertainty based method, the agent takes actions that minimise environmental uncertainty.
 - Uncertainty methods can use estimations from optimistic methods but they utilize them differently.
- 3.5. Imitation-Based Methods
- Exploration is 'kick-started' with demonstrations from different sources (usually humans).
 - Can be categorized to imitation in experience replay and imitation with exploration strategy



- 3.6. Safe Exploration
 - The problem of preventing agents from unsafe behaviours is considered. This is an important aspect of exploration research, as the agent's safety needs to be ensured.
 - Can be split into three categories
 - Human designer knowledge,
 - Prediction model, and
 - Auxiliary reward



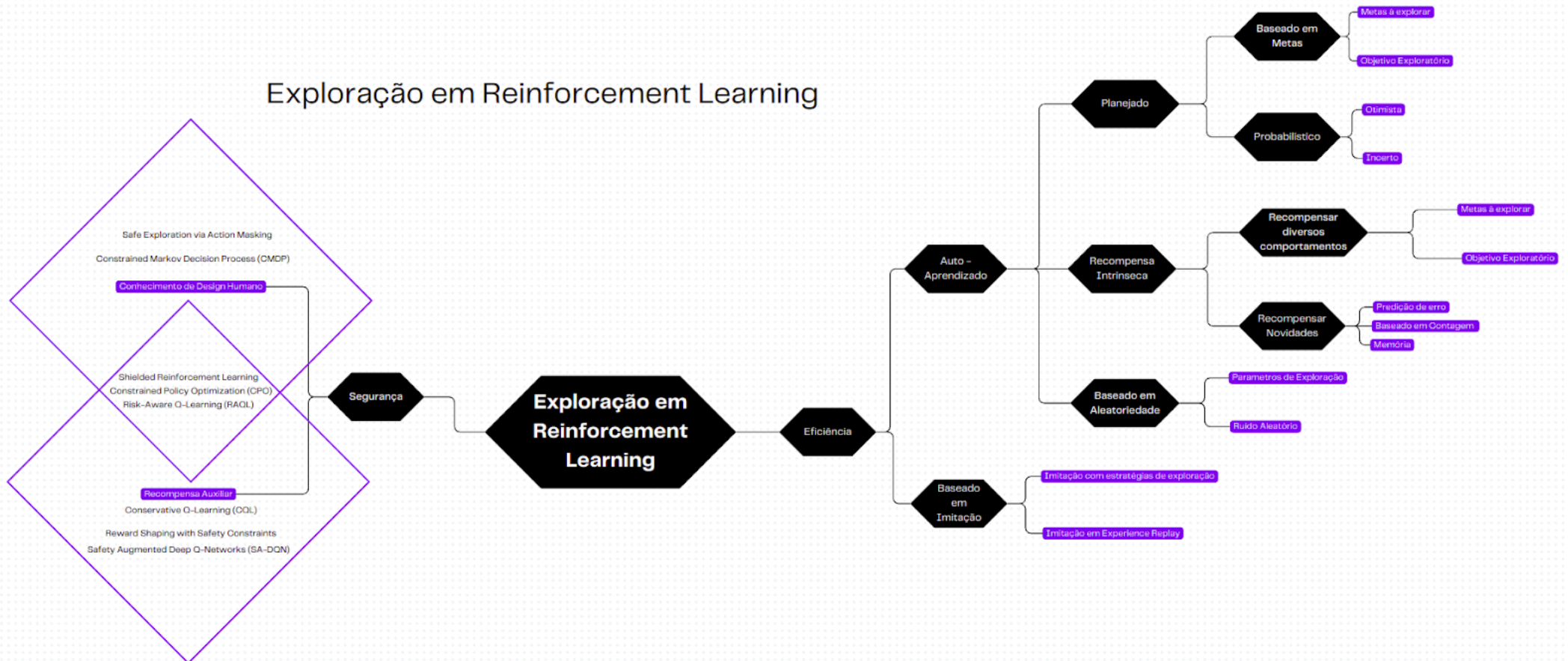
- 3.7. Random-Based Methods
 - Improvements to simple random exploration are discussed. Random exploration tends to be inefficient as it often revisits the same states.
 - To solve this problem, the following approaches are considered:
 - reduced states/actions for exploration methods,
 - exploration parameters methods and,
 - network parameter noise methods



- 4. Future Challenges
 - evaluation,

- scalability,
- exploration-exploitation dilemma,
- intrinsic reward,
- noisy TV problems,
- safety, and
- transferability.

Exploração em Reinforcement Learning



APÊNDICE 5

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: 17 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Fabrycio Leite Nakano Almada

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

Entregas da semana:

A tabela que apresentava os conceitos de Exploration em RL foi atualizada, agora relacionando a novas ferramentas relacionadas a Safe Exploration

- [Exploração em Reinforcement Learning](#)

O Resumo inicial da semana anterior, que foi construído a partir do artigo: [Exploration in Deep Reinforcement Learning: A Survey](#) foi atualizado e incrementado com conteúdos extra-artigo, como documentações do benchmark e conceitos encontrados em alguns dos papers referenciados no mesmo:

- [Exploration in Deep Reinforcement Learning: A Survey](#)

Criação de uma tabela visando catalogar alguns dos principais environments utilizados em RL e cujos conteúdos foram retirados das seguintes fontes:

- [GitHub - clvr/awesome-rl-envs](#)
- [List of Reinforcement Learning Environments | by Mauricio Fadel Argerich | Medium](#)
- [GitHub - tshirjn/env-zoo: A curated list of reinforcement learning environments and frameworks.](#)
- [Principais Frameworks: Reinforcement Learning](#)

Criação de uma tabela visando catalogar alguns dos principais frameworks utilizados em RL e cujos conteúdos foram retirados das seguintes fontes:

- [GitHub - kengz/awesome-deep-rl: A curated list of awesome Deep Reinforcement Learning resources.](#)

Tabela catálogo de Ambientes [RL Enviroments](#)

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

- Continuar a atualizar a tabela

- Continuar estudos com frameworks, buscando definir qual(is) usar
- Continuar estudos com ambientes, buscando definir qual(is) usar

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

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go! ▾

Minha Adaptação da taxidermia apresentada no artigo: [Taxidermia - FLNA](#)

Resumo completo do Artigo: [Exploration in Deep Reinforcement Learning: A Survey](#)

1 - Introduction

Problemas de Recompensa Escassa (Sparse reward problems): *Reward is rare and without a clear link to previous actions, thousands of decisions might need to be made before the outcomes are visible.*

Os problemas de recompensa escassa são comuns no mundo real, por exemplo:

- Durante missões de busca e resgate, a recompensa é dada apenas quando um objeto é encontrado.
- Durante uma entrega, a recompensa é dada apenas quando um objeto é entregue.

Técnicas para Exploração em Aprendizado por Reforço

- Exploração Aleatória

- O agente decide o que fazer aleatoriamente, independentemente de seu progresso.
 - A técnica mais comumente usada desse tipo é o ϵ -greedy:
 - ϵ -greedy: usa o parâmetro de decaimento temporal para reduzir a exploração ao longo do tempo.
 - Muitas vezes impraticável devido aos longos tempos de aprendizado.
 - Mostrou um desempenho impressionante em jogos de Atari, simulador Mujoco, ajustes de controladores, pouso autônomo, carros autônomos e cuidados com a saúde.
- Moldagem de Recompensa
 - O projetista impõe 'artificialmente' uma recompensa mais frequente.
 - Ex: Para missões de busca e resgate, os agentes podem receber uma recompensa negativa toda vez que não encontram a vítima.
 - É um problema desafiador que depende fortemente da experiência do projetista.
 - É um problema desafiador que depende fortemente da experiência do projetista.
- Exploração em Aprendizado por Reforço ainda em desenvolvimento, mas algoritmos de exploração têm mostrado um ganho significativo em desempenho em comparação com algoritmos não-exploratórios.
 - Diversity is All You Need (DIYAN) aprimorou benchmarks no MuJoCo.
 - Random Network Distillation (RND) e pseudocontagens foram os primeiros a pontuar no problema desafiador de Montezuma's Revenge.
 - Agent57 é o primeiro a superar os humanos em todos os 57 jogos do Atari.
- O review foca em abordagens exploratórias que atendem pelo menos um dos seguintes critérios:
 1. Determina o grau de exploração com base no aprendizado do agente,
 2. Decide ativamente realizar certas ações na esperança de encontrar novos resultados, e
 3. Motiva-se a continuar explorando apesar da falta de recompensas ambientais.

2 - Preliminares

2.1. Introdução ao Aprendizado por Reforço

2.1.1. Processo de Decisão de Markov

- Processo de Decisão de Markov (MDP): Ambiente estocástico e totalmente observável onde um agente interage escolhendo ações de forma sequencial, em etapas discretas de tempo, com o objetivo de maximizar recompensas acumuladas.
- Descrito por uma tupla (S, A, P, R, γ) onde:
 - S é o conjunto de estados,
 - A é o conjunto de ações que o agente pode selecionar,
 - P é a probabilidade de transição que satisfaz a propriedade de Markov, representada por:
 - $p(st+1|st, at)$,
 - R é o conjunto de recompensas,
 - $\gamma \in (0, 1]$ é o fator de desconto.
- Em cada instante de tempo t , o agente recebe um estado $st \in St$ do ambiente e seleciona a melhor ação $at \in At$ conforme a política $\pi(at|st)$, que mapeia estados para ações. Após realizar uma ação at , o agente recebe uma recompensa $R_t \in R$ do ambiente. O objetivo do agente é maximizar a recompensa esperada descontada
 $G_t = \sum_{k=0}^{\infty} \gamma^k * r_{k+t}$ a partir de cada estado st .

2.1.2. Métodos Baseados em Valor

- Quando o agente segue a política π , a função de valor de estado é definida como $V_{\pi}(st) = E_{\pi} [G_t | St=s]$
- De forma similar, a função de valor de ação
 $Q(st, at) = Q(st, at) + \alpha [r_t + \gamma \max_{a'} Q(st+1, a') - Q(st, at)]$
-

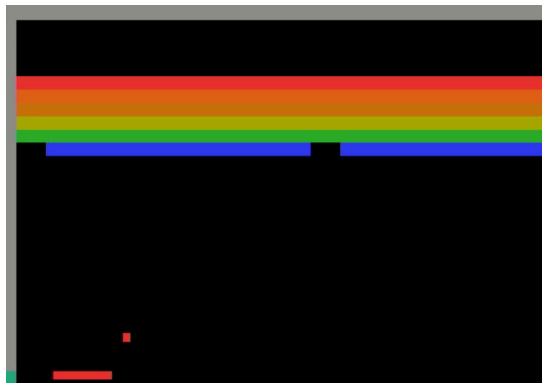
2.1.3. Métodos Baseados em Política

- Ao contrário dos métodos baseados em valor, os métodos baseados em política atualizam diretamente a política parametrizada por θ .

2.2 Exploration

- Atividade de buscar e descobrir algo
- No Aprendizado por Reforço, "algo" é uma função de recompensa, e "buscar e descobrir" é a tentativa do agente de maximizar essa função de recompensa.
- A exploração é mais útil em problemas onde o caminho para a solução real (ou seja, recompensa) é obstruído por mínimos (máximos) locais ou áreas de recompensas planas.
- **Descobrir a verdadeira natureza das recompensas é desafiador.**

- 2.3. Problemas Desafiadores
 - Noisy-TV
 - O agente fica preso explorando um número infinito de estados que não levam a nenhuma recompensa.
 - Problemas de Recompensa Escassa
 - Há um longo intervalo entre uma ação e uma recompensa (a recompensa é relativamente rara).
- 2.4 Benchmarks
 - 2.4.1. Jogos Atari
 - Conjunto de 57 jogos Atari combinados sob o Atari Learning Environment (ALE).
 - O espaço de estados geralmente consiste em imagens ou snapshots da memória RAM.
 - O espaço de ações inclui cinco movimentos do joystick (cima, baixo, esquerda, direita e botão de ação).
 - O espaço de ações inclui cinco movimentos do joystick (cima, baixo, esquerda, direita e botão de ação).
 - Exploração Fácil: A recompensa é relativamente fácil de encontrar.



Breakout for Atari 2600. Fonte: [Breakout - Atari 2600 Gameplay](#)

ALE: [Atari - Gymnasium Documentation](#)

- Exploração Difícil: A recompensa não é dada frequentemente e a associação entre estados e recompensas é complexa.



Pitfall for Atari 2600. Fonte: [AtariAge - Atari 2600 Screenshots - Pitfall! \(Activision\)](#)

- 2.4.2. VizDoom
 - Benchmark baseado no jogo Doom.
 - O jogo tem uma perspectiva em primeira pessoa e a imagem vista pelo personagem é normalmente usada como espaço de estados.
 - O espaço de ações consiste normalmente em oito controles direcionais e dois botões de ação (pegar cartões-chave e abrir portas).
 - Mais ações podem ser adicionadas, se necessário.
 - Ferramentas fáceis de usar para edição de cenários e baixo custo computacional.



Doom. Fonte: [VDAIC 2017 @ CIG](#)
Vizdoom: [ViZDoom](#)

- 2.4.3 Malmo
 - Benchmark baseado no jogo Minecraft.
 - Também com perspectiva em primeira pessoa, e a imagem é o espaço de estados.

- A principal vantagem do Malmo é sua flexibilidade em termos de estrutura do ambiente, tamanho do domínio, scripts personalizados e funções de recompensa.
- Malmo: [GitHub - tsmatz/minecraft-rl-example: Applying Reinforcement Learning in Minecraft \(Project Malmo Tutorial\)](#)
- 2.4.4. Mujoco
 - Representa dinâmicas de múltiplas articulações com contato.
 - Benchmark popular usado para simulações baseadas em física.
 - Normalmente utilizado para simular robôs caminhando.
 - Mujoco: [MuJoCo Documentation](#)

3 - Exploração no Aprendizado por Reforço:

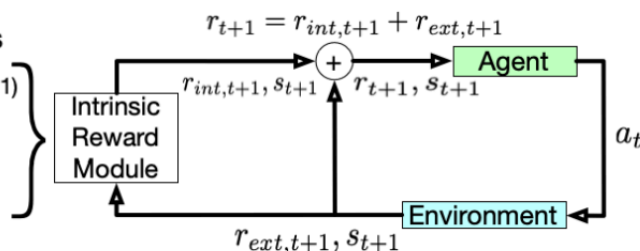
- Pode ser dividida em dois principais enfoques:
 - Eficiência
 - Tornar a exploração mais eficiente em termos de amostras, de modo que o agente possa explorar com o menor número de passos possível.
 - Exploração Segura
 - Garantir a segurança durante o processo de exploração.
- 3.1. Recompensa por Estados Inéditos

As abordagens de recompensa por estados inéditos dão aos agentes uma recompensa por descobrirem novos estados.

 - Essa recompensa é chamada de recompensa intrínseca.

Reward Novel states

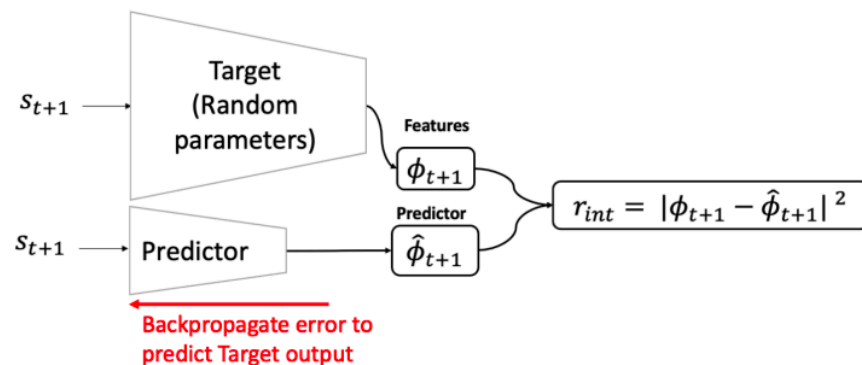
- Prediction Error (Sec. 3.1.1)
- Count Based (Sec. 3.1.2)
- Memory (Sec. 3.1.3)



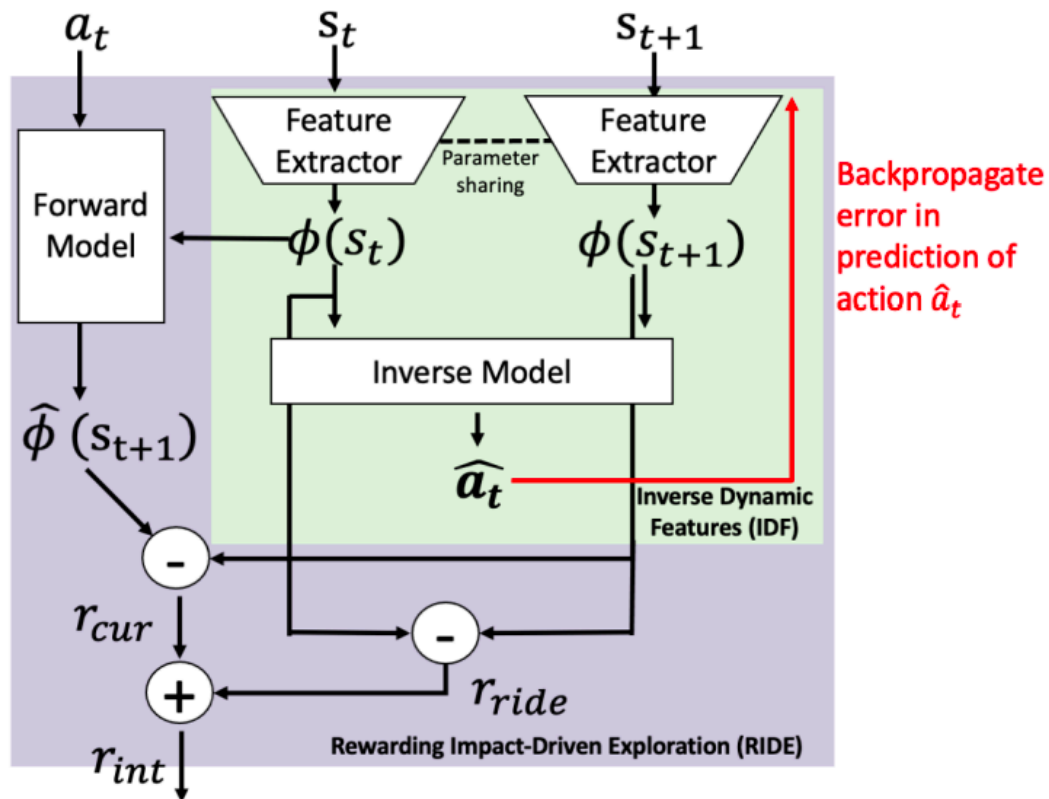
- 3.1.1. Prediction Error Methods
 - O erro de um modelo de previsão ao prever um estado previamente visitado é usado para calcular a recompensa intrínseca.

- Uma das principais questões a ser abordada é como usar o erro do modelo para calcular a recompensa intrínseca.
- Compararam-se duas funções de recompensa intrínseca:
 - (i) o tamanho do erro em um modelo de previsão e
 - (ii) o progresso do aprendizado.
- O primeiro método demonstrou melhor desempenho e, portanto, é recomendado. Isso pode ser formalizado da seguinte maneira:
 - $r_{int} = f(z(s_{t+1}) - M(z(s_t, a)))$
 - onde s representa um estado,
 - M é um modelo ambiental,
 - t e $t+1$ são dois passos de tempo consecutivos,
 - z é um modelo opcional para representação de estados, e
 - f é uma função opcional de escala de recompensa.
- A recompensa intrínseca é medida como a distância Euclidiana entre a previsão de um estado por um modelo e o estado real.
- Desde então, muitas outras abordagens foram desenvolvidas, que podem ser divididas em:
 - (i) previsão de representação de estados,
 - (ii) conhecimento a priori, e
 - (iii) incerteza sobre o ambiente.
- Previsão de representação de estados:
 - O estado é representado em um espaço de dimensão superior.
 - Em seguida, um modelo é encarregado de prever a próxima representação de estado com base na representação de estado anterior.
 - Quanto maior for o erro na previsão, maior será a recompensa intrínseca.
 - Uma maneira de fornecer a representação de estado é usando um autoencoder.
 - Foram considerados tanto autoencoders pré-treinados quanto treinados online, e ambos mostraram desempenho semelhante.
 - Melhorias nas abordagens baseadas em autoencoders foram propostas, nas quais um autoencoder de treinamento lento foi adicionado.
 - Assim, a recompensa intrínseca decai mais lentamente e o agente explora por mais tempo, aumentando a chance de encontrar a recompensa ótima.

- Outro método de fornecer a representação de estado envolve o uso de redes fixas com pesos aleatórios.
 - Em seguida, uma outra rede é usada para prever as saídas de redes inicializadas aleatoriamente.
- A abordagem mais popular desse tipo é chamada de Random Network Distillation (RND).



- Um método de representação de estado derivado de recursos dinâmicos inversos (Inverse Dynamic Features, IDF) foi utilizado. No IDF, a representação vem de forçar o agente a prever a ação.
- O IDF foi comparado com o método de previsão de estado e a representação aleatória, chegando às seguintes conclusões: o IDF apresentou o melhor desempenho e se adapta melhor a ambientes desconhecidos.
- O IDF foi utilizado onde a distância Euclidiana entre duas representações de estado consecutivas foi usada como recompensa intrínseca.
- Intuitivamente, quanto mais significativa é a transição, maior é a mudança na representação de estado do IDF.
- Em outro estudo, o RND e o IDF foram combinados em uma única recompensa intrínseca.
-



- Vantagens dos Métodos de Erro de Previsão
 - Esses métodos dependem apenas de um modelo do ambiente, sem necessidade de buffers ou métodos complexos de aproximação.
- Categorias de Métodos e Seus Desafios
 - Previsão Direta de Estado: Requer pouco ou nenhum conhecimento prévio, mas o modelo precisa aprender a reconhecer diferentes estados e pode ter dificuldades com muitos estados no ambiente.
 - Métodos de Representação de Estado: Lida com grandes espaços de estados, mas com custo de maior esforço do projetista e menor precisão, pois o agente não pode influenciar a representação de estado diretamente.
 - Uso de Conhecimento Prévio: Define elementos específicos do espaço de estado como fonte de erro para calcular a recompensa intrínseca, dependendo da experiência do projetista.
- Incerteza Sobre o Ambiente: A incerteza do agente é usada para gerar a recompensa intrínseca, com alta escalabilidade e transição automática entre exploração e exploração.

- Capacidades dos Métodos de Representação de Estado
 - Embora eficazes em grandes espaços de estado, esses métodos podem representar estados diferentes de maneira semelhante, complicando a precisão.
- Benefícios dos Métodos de Conhecimento Prévio e Incerteza
 - Métodos de conhecimento prévio não têm problemas com velocidade de previsão e reconhecimento de estado, enquanto abordagens baseadas na incerteza são altamente escaláveis.
- Solução para o Problema do Couch-Potato (Noisy-TV)
 - Métodos de erro de previsão resolveram esse problema armazenando observações em um buffer de memória e recompensando o agente apenas quando ele observa estados suficientemente distantes dos já armazenados, evitando a recompensa para estados repetidos e próximos.
- 3.1.2. Métodos Baseados em Contagem
 - Cada estado é associado a um número de contagem de visitas $N(s)$
 - Estados com baixa contagem recebem alta recompensa intrínseca para incentivar revisitas.
 - Fórmulas para Recompensa
 - Calculada como $1/N(s)$, o que proporciona uma taxa de convergência mais rápida do que a fórmula comum $1/N(s)$.
 - Desafios em Ambientes com Muitos Estados
 - Contar visitas em problemas com muitos estados é difícil e demanda muita memória. Para resolver isso, a contagem é normalmente feita em uma representação de estado reduzida.
 - Representação de Estado Reduzida
 - Permitem armazenar a contagem e o estado com menos memória, facilitando a contagem em espaços de estado grandes.
 - Exemplos de Métodos Populares
 - Técnicas como SimHash são usadas para representar imagens como hashes compactos. Métodos como ADM representam o estado pela posição aprendida do agente. SSR combina contagem e ordem dos estados como um substituto da contagem.
 - Aproximações de Contagem

- É possível aproximar a contagem usando modelos de densidade, como CTS, misturas Gaussianas e PixelCNN, para reduzir a necessidade de contagem direta.
- Discussão sobre Métodos Baseados em Contagem
 - Métodos baseados em contagem aproximam a recompensa intrínseca pelo número de vezes que um estado é visitado. Contudo, essas representações de estado podem exigir muita memória e precisam ser cuidadosamente projetadas para serem eficientes.
- 3.1.3. Memory Methods
 - Atribuem uma recompensa intrínseca com base na facilidade de distinguir um estado de todos os outros. Quanto mais fácil distinguir, mais inédito é o estado.
 - Desafios de Comparação Direta de Estados
 - Comparar estados diretamente é caro em termos computacionais, então várias aproximações foram desenvolvidas, como modelos de comparação e experience replay.
 - Modelos de Comparação
 - Modelos podem ser treinados para comparar estados diretamente, reduzindo o custo computacional. Um exemplo é o *exemplar model*, que usa múltiplos classificadores treinados para distinguir estados específicos.
 - Estratégias de Redução de Custos nos Exemplars
 - Para reduzir o custo de treinamento dos classificadores, atualiza-se um exemplar com cada novo ponto de dados e utiliza-se amostragem de exemplares de um buffer limitado.
 - Combinação de Experience Replay e Erro de Previsão
 - Em vez de criar modelos de comparação, foi proposto combinar um experience replay limitado com métodos de erro de previsão para calcular recompensas intrínsecas, incluindo o uso de RND para determinar a novidade a longo prazo.
 - Aprendizado de Múltiplas Políticas
 - Múltiplas políticas são treinadas, cada uma com uma relação diferente entre recompensas extrínsecas e intrínsecas. Um meta-aprendiz propõe escolher automaticamente essa relação em cada etapa.
 - Discussão sobre Métodos Baseados em Memória
 - Métodos baseados em memória permitem ao agente derivar recompensas intrínsecas comparando o estado atual com

estados armazenados. O método de modelo de comparação requer menos memória, mas depende de uma afinação cuidadosa dos parâmetros, enquanto o experience buffer limitado evita problemas de imprecisão de modelo e tem mostrado bom desempenho em jogos desafiadores de Atari.

- 3.1.4. Summary
 - Resumo das Abordagens Baseadas em Estados Inéditos
 - A Tabela 1 resume as abordagens baseadas em recompensas por estados inéditos, utilizando uma legenda para interpretar termos específicos como espaço de ações (A), ação (Ac), referência (R), e outros.
 - Métodos Baseados em Erro de Previsão
 - São os métodos mais usados e apresentam desempenho forte, como o RND, que alcançou uma pontuação de 6.500 em Montezuma's Revenge. Contudo, eles geralmente necessitam de métodos de representação de estado específicos e adaptados para maior eficiência computacional.
 - Limitations of Prediction Error Methods
 - Embora eficazes, esses métodos exigem adaptações específicas ao problema, o que limita sua aplicabilidade em outros contextos.
 - Eficiência dos Métodos Baseados em Contagem
 - Métodos baseados em contagem são computacionalmente eficientes, mas requerem memória para armazenar contagens ou modelos complexos, especialmente em domínios de estado contínuo, onde estados contínuos devem ser agrupados em segmentos discretos.
 - Desafios em Domínios de Estado Contínuo
 - A contagem em estados contínuos é desafiadora e exige uma divisão em partes descontínuas, o que pode reduzir a precisão do método.
 - Desempenho dos Métodos Baseados em Memória
 - Recentemente, esses métodos apresentaram bom desempenho em jogos como Montezuma's Revenge, com pontuações de até 11.000. Eles requerem um equilíbrio cuidadoso da quantidade de dados a serem mantidos em memória.
 - Requisitos e Considerações dos Métodos Baseados em Memória

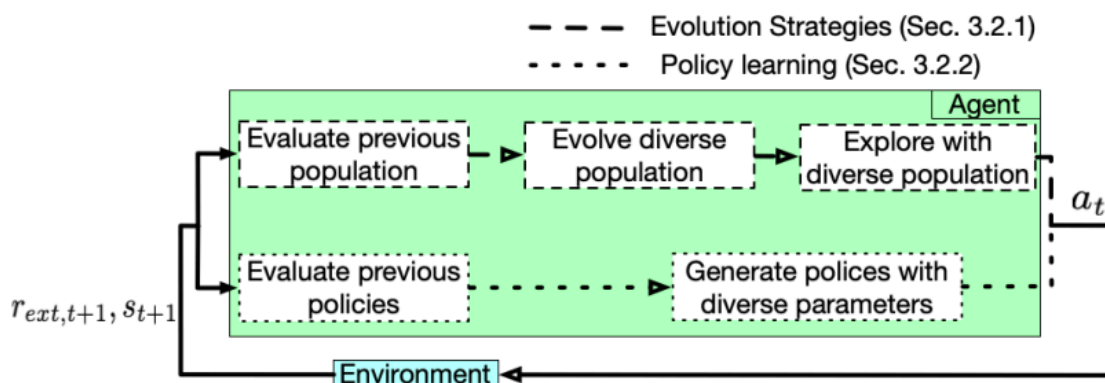
- Os métodos baseados em memória precisam de ajustes na quantidade de dados armazenados para evitar excesso de consumo de recursos enquanto mantém bom desempenho.

R	Prior Knowl- edge	U	Method	Top score on a key benchmark	Input Types	O	MB/ MF	A/ S
Pathak et al. [41]		A3C	Pre- diction error	Vizdoom: very sparse 0.6 (A3C 0)	Viz- doom image	P	MB	D/ D
Stadie et al. [37]	autoen- coder	DQN	Pre- diction error	Atari: Alien 1436 (DQN 300)	Atari images	Q	MB	D/ D
Savinov et al. [60]	pretrained discrim- inator (non- online)	PPO	Pre- diction error	Vizdoom: very sparse 1 (PPO 0); Dmlab: very sparse 30 (PPO+ICM 0)	Viz- doom images/ Mujoco joints angles	Ac	MB	C/ C
Burda et al. [13]		PPO	Pre- diction error	Atari: Mon- tezuma Revenge 7500 (PPO 2500)	Atari images	P	MB	D/ D
Bougie and Ichise [38]		PPO	Pre- diction error	Atari: Mon- tezuma Revenge 20 rooms found (RND 14)	images	Ac	MB	D/ D
Hong et al. [36]		DQN	Pre- diction error	Atari: Mon- tezuma Revenge 200 (DQN 0)	Enu- marated state id/ Atari Image	Q	MB	D/ D
Kim et al. [46]		TRPO	Pre- diction error	Atari: Frost- bite 6000 (ICM 3000)	Atari Images	Ac	MB	D/ D

Stanton and Clune [51]	agent position, reward grid	A2C	Prediction error	Atari: Montezuma Revenge 3200 (A2C 0)	Atari images	Ac	MB	D/D
Achiam and Sasstry [33]		TRPO	Prediction error	Mujoco: halfcheetah 80 (VIME 40); Atari: Venture 400 (VIME 0)	Atari RAM states/ Mujoco joints angles	Ac	MB	C/C
Li et al. [44]		A2C	Prediction error	Atari: Asterix 500000 (RND 10000)	Atari images	Ac	MB	D/D
Kim et al. [47]		PPO	Prediction error	Atari: Montezuma Revenge with distraction 1500 (RND 0)	Atari images	Ac	MB	D/D
Chien and Hsu [59]		DQN	Prediction error	PyDial: 85 (CME 80); OpenAI: Mario 0.8 (CME 0.8)	Images	Q	MB	D/D
Li et al. [34]		DDPG	Prediction error	Robot: Fetch-Push 1 (DDPG 0)	Robot joints angles	Ac	MB	C/C
Raileanu and Rocktäschel [43]		IM-PALA	Prediction error	Vizdoom: 0.95 (ICM 0.95)	Vizdoom Images	Ac	MB	D/D
Mirowski et al. [48]		A3C	Prediction error	DM Lab: Random Goal 96 (LSTM-A3C 65)	DM Lab images	Ac	MB	C/C

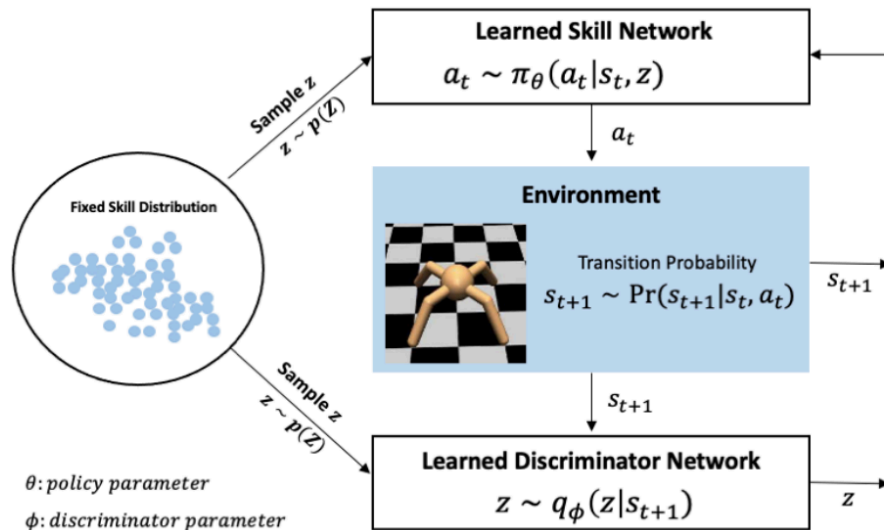
Tang et al. [62]		TRPO	Count-based	Atari: Montezuma Revenge 238 (TRPO 0); Mujoco: swimmergather 0.3 (VIME 0.15)	Atari images/ Mujoco joints angles	P	MF	C/C
Martin et al. [68]	Blob-PROST features	SARSA-e	Count-based	Atari: Montezuma Revenge 2000 (SARSA 200)	Blob-PROST features	Q	MB	D/D
Machado et al. [65]		DQN	Count-based	Atari: Montezuma Revenge 1396 (Psuedo counts 1671)	Atari images	Q	MF	D/D
Ostrovski et al. [67]		DQN and Reactor	Count-based	Atari: Gravitar 1500 (Reactor 1000)	Atari images	Ac	MB	D/D
Badia et al. [71]		R2D2	Memory	Atari: Pitfal 15000 (R2D2 -0.5)	Atari images	P	MB	D/D
Badia et al. [15]		R2D2	Memory	Beat humans in all 57 atari games	Atari images	P	MB	D/D
Fu et al. [70]	state encoder	TRPO	Memory	Mujoco: Sparse-HalfCheetah 173.2 (VIME 98); Atari: Frostbite 4901 (TRPO 2869); Doom: MyWay-Home 0.788 (VIME 0.443)	Atari images/ Mujoco joints angles	Ac	MB	C/C

- 3.2. Recompensar Comportamentos Distintos
 - O agente coleta o máximo de experiências diferentes possível.
 - Exploração como um objetivo ao invés de uma busca por recompensas.



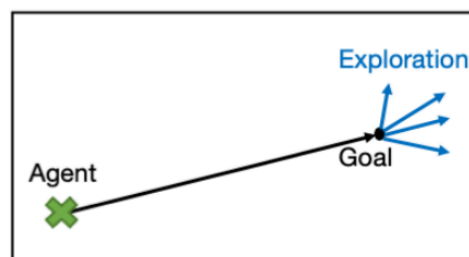
- 3.2.1. Estratégias Evolutivas

- Abordagens baseadas em evolução utilizam uma população de soluções amostrais que evolui com o tempo para se aproximar da solução ideal, embora geralmente não sejam consideradas parte do aprendizado por reforço.
 - Busca por Novidade: Um dos primeiros métodos, chamado novelty search, incentiva o agente a gerar comportamentos diversos utilizando uma métrica de diversidade projetada para cada ambiente. A transferência entre domínios é limitada devido à necessidade de personalização
 - Combinações Recentes com Maximização de Recompensa: Novelty search foi combinado com maximização de recompensa e métodos de estados inéditos.
 - Evolução das Abordagens de Busca por Novidade: Inicialmente usada isoladamente, a busca por novidade tem mostrado mais potencial ao ser combinada com outras técnicas. Essa combinação permite direcionar comportamentos diversos para os que alcançam pontuações mais altas.
- 3.2.2. Aprendizado de Políticas
- Medidas de diversidade foram aplicadas recentemente em abordagens de aprendizado de políticas, medindo a diversidade entre políticas por meio de métricas como divergência de KL ou erro médio quadrado.
 - Função Objetivo para Políticas Diversas
 - A função objetivo inclui
 - (i) maximizar a entropia das habilidades,
 - (ii) inferir comportamentos a partir do estado atual, e
 - (iii) maximizar a aleatoriedade dentro de uma habilidade, como mostrado no método Diversity is All You Need (DIAYN).

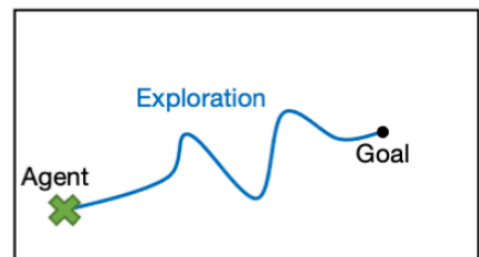


- Abordagem Baseada em Objetivos
 - Uma combinação de diversidade com uma abordagem baseada em objetivos, como no algoritmo skew-fit, permite ao agente aprender objetivos diversos e úteis para recompensas, ajustando a distribuição de objetivos para visitar estados raramente explorados.
- Armazenamento de Políticas de Sucesso
 - O agente armazena políticas bem-sucedidas em um experience replay, minimizando a diferença entre a política atual e as melhores armazenadas, ao mesmo tempo em que maximiza a entropia dos parâmetros entre políticas para permitir a exploração.
- Desempenho e Resultados
 - Esse método apresentou desempenho superior em problemas de recompensa escassa, como no Mujoco, superando estratégias evolutivas e PPO.
- A diversidade em abordagens baseadas em políticas é um conceito novo e promissor, com um critério de diversidade bem projetado apresentando desempenhos significativamente melhores do que os métodos tradicionais de aprendizado por reforço.
- 3.2.3. Summary
 - Estratégias evolutivas utilizam uma população diversa, enquanto o aprendizado de políticas encontra políticas diversas.
 - Essas estratégias têm o potencial de encontrar soluções não previstas pelos projetistas, explorando tanto a estrutura da rede neural quanto a diversidade.

- As estratégias evolutivas sofrem de baixa eficiência de amostragem, tornando o treinamento caro em termos computacionais ou lento.
 - O aprendizado de políticas não ultrapassa as estruturas predefinidas, mas pode apresentar resultados notáveis, especialmente em problemas com recompensas escassas.
 - Uma vantagem adicional do método de aprendizado de políticas é sua adequação para problemas com espaços de estado contínuos e discretos.
- 3.3. Métodos Baseados em Objetivos
- Nos métodos baseados em objetivos, a tarefa do agente é alcançar um objetivo específico (ou estado). A partir daí, o objetivo é explorado usando outro método de exploração ou para gerar uma meta exploratória.
 - O conceito principal é guiar os agentes diretamente para áreas inexploradas, permitindo que a exploração se concentre imediatamente em regiões amplamente desconhecidas.
 - O agente precisa de um gerador de objetivos, uma política para alcançar os objetivos e uma estratégia de exploração. O gerador cria objetivos, a política alcança esses objetivos e a estratégia de exploração atua durante ou após a conquista de um objetivo.
 - Esses métodos são divididos em duas categorias: goals to explore from (objetivos de onde explorar) e exploratory goal methods (métodos de objetivos exploratórios).



Goals to Explore from (Sec. 3.3.1)



Exploratory Goals (Sec. 3.3.2)

- 3.3.1. Goals to Explore from Methods (Métodos para Explorar a Partir de Objetivos)
 - As principais técnicas são:
 - (i) memorizar estados e trajetórias visitadas — armazenando-os em um buffer e escolhendo um objetivo exploratório a partir dele; e
 - (ii) aprender com o objetivo — assumindo que o estado objetivo é conhecido, mas o caminho até ele é desconhecido.
 - Abordagem Go-Explore

- Salva estados e trajetórias em um buffer e os seleciona probabilisticamente. Uma vez escolhido o estado a ser explorado, o agente é "teletransportado" até ele e explora aleatoriamente. Em outras variações, o teletransporte é substituído pelo aprendizado de políticas e foi estendido para domínios contínuos.
- Trajetórias do Passado como Referência
 - Outras abordagens selecionam trajetórias passadas para o agente explorar aleatoriamente, com base na contagem de visitas, ajudando a focar em estados menos explorados.
- Escolha de Estados Menos Visitados
 - Em um estudo, o estado menos visitado em uma grade x-y em jogos de Atari foi escolhido como objetivo, reduzindo o esforço computacional de rastrear onde o agente já esteve.
- Métodos de Aprendizado com o Objetivo
 - Esses métodos assumem que o agente conhece o estado de maior recompensa, mas não como chegar lá. O modelo pode prever passos de volta no aprendizado por reforço, permitindo que o agente explore desde o estado objetivo.
- Exploração Baseada no Estado de Recompensa
 - Permite ao agente "imaginar" estados anteriores ao objetivo e iniciar a exploração a partir dele, mitigando o problema de exploração, uma vez que o agente sabe onde procurar a recompensa.
- Discussão
 - Métodos de memorização de estados e trajetórias visitadas mostraram resultados promissores em benchmarks difíceis, como Montezuma's Revenge. Utilizar o estado de recompensa como objetivo ajuda a orientar a exploração.
- 3.3.2. Exploratory Goal Methods (Métodos de Objetivos Exploratórios)
 - O agente recebe um objetivo exploratório a ser alcançado, promovendo a exploração durante a tentativa de atingir o objetivo. As técnicas incluem:
 - (i) meta-controladores,
 - (ii) objetivos em regiões de alta incerteza, e
 - (iii) métodos de sub-objetivos.
 - Meta-controladores
 - Consistem em um controlador de alto nível e um trabalhador. O controlador define objetivos, e o trabalhador tenta alcançá-los. Abordagens variam de seleção aleatória de objetivos a seleção

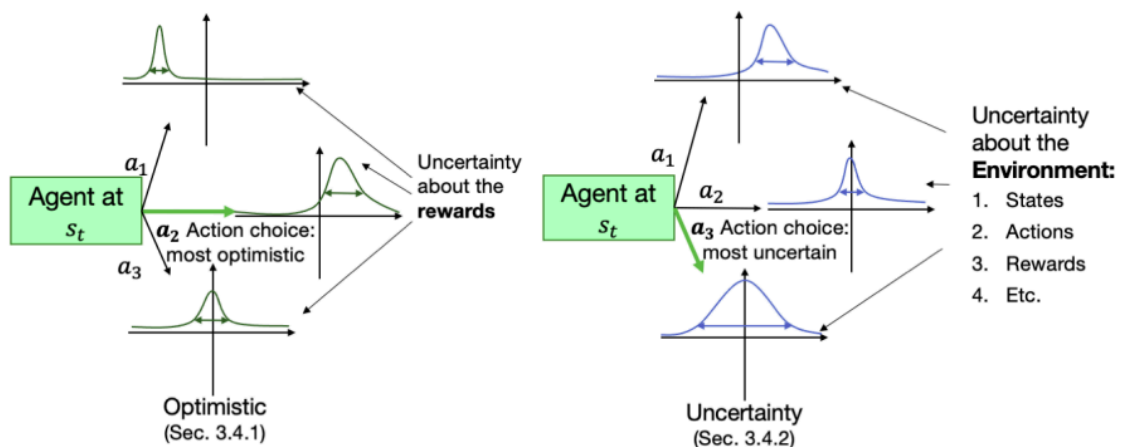
baseada em progresso de aprendizado, e sistemas em que cada subtarefa é atribuída a um aprendiz específico.

- Métodos de Sub-objetivos
 - Esses métodos identificam sub-objetivos para o agente, divididos em: (i) estados de gargalo que levam a outros estados, (ii) progresso em direção ao objetivo principal, e (iii) sub-objetivos baseados em incerteza.
- Estados de Gargalo
 - Descobertos através de técnicas como otimização por colônias de formigas ou funções de valor proto-baseadas em valores próprios. Esses estados servem como pontos-chave frequentemente visitados que conectam várias partes do ambiente.
- Progresso para o Objetivo Principal
 - Projetado para criar sub-tarefas que levam ao objetivo principal, usando componentes como estimadores de progresso de aprendizado e geradores de tarefas, que desenvolvem subtarefas progressivamente mais próximas da meta final.
- Sub-objetivos Baseados em Incerteza
 - Focados em estados de alta incerteza de recompensa, usando técnicas como agrupamento por k-means ou modelos de mistura Gaussiana, o que ajuda a evitar problemas como o comportamento de "couch potato".
- Discussão
 - Os meta-controladores permitem que o agente defina seus próprios objetivos, mas o treinamento do controlador ainda é desafiador. Métodos de sub-objetivos dependem da definição de objetivos pelos projetistas, colocando um grande peso no design de metas adequadas e significativas.
-
- 3.3.3. Summary
 - Desafios dos Métodos Goals to Explore From
 - Esses métodos exigem o armazenamento de estados e trajetórias, além de enfrentar dificuldades para navegar até o objetivo. Isso é parcialmente resolvido utilizando a posição do agente como representação de estado, embora essa abordagem seja altamente específica do problema.
 - Limitações dos Métodos de Objetivos Exploratórios
 - Criar objetivos exploratórios torna-se mais difícil com o aumento da escassez de recompensas, embora abordagens

tenham sido propostas para mitigar isso, como parametrização de tarefas.

- Soluções Parciais
 - Trabalhos recentes abordaram algumas dessas limitações ao parametrizar tarefas e utilizar representações de estado específicas, mas essas soluções ainda dependem da natureza do problema e da capacidade de adaptar o método ao ambiente.
- 3.4. Probabilistic Methods
 - Métodos Probabilísticos
 - Abordagens probabilísticas permitem que o agente mantenha uma probabilidade sobre estados, ações, valores e recompensas, escolhendo a próxima ação com base nessa probabilidade.
 - Divisão dos Métodos Probabilísticos
 - São divididos em métodos otimistas e métodos baseados em incerteza. A principal diferença está em como modelam a probabilidade e como o agente a utiliza.
 - Métodos Otimistas
 - Dependem de uma estimativa de recompensa, seja implícita ou explicitamente, utilizando o limite superior da estimativa para orientar a escolha da ação.
 - Métodos Baseados em Incerteza
 - Estimam a incerteza sobre o ambiente, como a função de valor ou previsão de estado, e o agente toma ações para minimizar essa incerteza ambiental.
 - Interações entre Métodos
 - Métodos de incerteza podem utilizar estimativas de métodos otimistas, mas as aplicam de forma distinta, focando na redução da incerteza ao invés de maximizar recompensas diretamente.
- 3.4.1. Métodos Otimistas
 - Esses métodos seguem o princípio do otimismo sob incerteza, onde o agente usa o limite superior de confiança da estimativa de recompensa.
 - Uso de Processos Gaussianos (GP)
 - Processos Gaussianos são utilizados como modelo de recompensa, oferecendo incerteza para estimativa de recompensa. O algoritmo linear Gaussiano também pode ser usado como modelo.
 - Redes Profundas DQN com Bootstrap e Amostragem de Thompson

- Redes profundas DQN com bootstrap e amostragem de Thompson foram exploradas para fornecer uma distribuição sobre recompensas e valores, permitindo decisões otimistas.
- Conjunto de Funções de Valor
 - Durante a exploração, o agente mantém um conjunto de funções de valor, utilizando a função mais otimista para cada episódio, e atualiza a distribuição ao final.
- Discussão
 - Nos métodos otimistas, o agente aplica otimismo sob incerteza ao modelar recompensas diretamente ou aproximando funções de valor. Essa aproximação é vantajosa em ambientes com recompensas escassas, permitindo ao agente utilizar recompensas parciais das funções de valor para aprendizado.

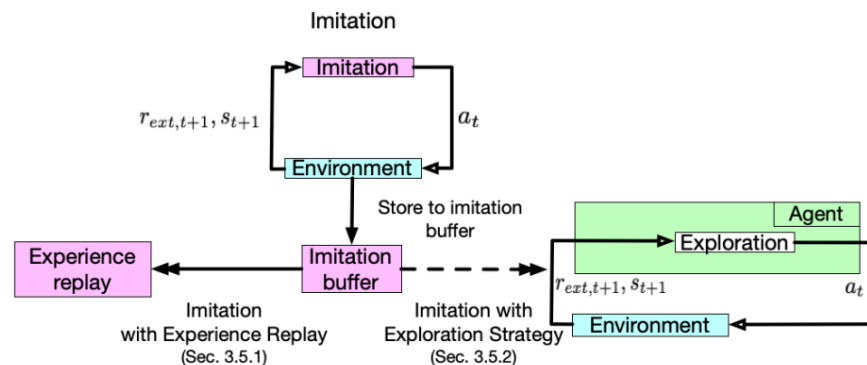


- 3.4.2. Métodos Baseados em Incerteza
 - Esses métodos utilizam uma distribuição de probabilidade sobre ações e/ou estados, representando a incerteza ambiental, e escolhem ações que minimizam essa incerteza.
 - Incerteza de Parâmetros
 - O agente mantém incerteza sobre os parâmetros que definem uma política, amostrando e seguindo essa política por certo tempo antes de atualizar com base no desempenho. Uma abordagem simples é manter uma distribuição sobre os parâmetros da rede.
 - Incerteza de Políticas e Q-Values
 - Nesses métodos, o agente mantém incerteza sobre valores Q/ações e amostra a ação apropriada. Abordagens populares incluem o uso de frameworks Bayesianos e métodos de

otimização como cross-entropy. Métodos baseados em teoria da informação maximizam o ganho de informação, reduzindo a incerteza.

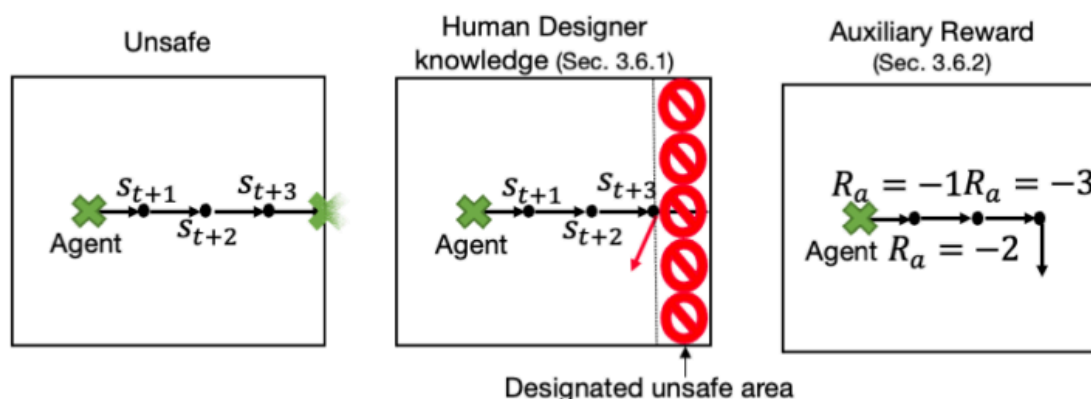
- Ensemble de Redes
 - O agente usa vários modelos (inicializados com parâmetros diferentes) para aproximar a distribuição. Cada modelo no ensemble estimativa valores Q, permitindo que o agente amostre e atualize a política com base na discordância entre os modelos.
- Exemplos de Métodos de Ensemble
 - Técnicas incluem redes DQN com múltiplas "cabeças" que estimam valores Q e distribuições Gaussianas ajustadas. Outro exemplo é maximizar a discordância entre modelos ambientais para decidir a política de exploração.
- Discussão sobre Amostragem de Parâmetros e Políticas/Q-values
 - Amostrar parâmetros permite um ciclo de atualização-exploração contínuo. Em contrapartida, a amostragem de políticas/Q-values permite ajustes mais rápidos da política. No entanto, a estimativa exata das probabilidades pode ser complexa, exigindo simplificações.
- Vantagens e Desafios dos Métodos de Ensemble
 - Embora esses métodos possam ser computacionalmente intensivos, eles permitem atualizações instantâneas e flexíveis, utilizando distribuições aproximadas sobre ações/estados.
- 3.4.3. Summary
 - Conclusões sobre Exploração Otimista
 - O maior problema na exploração otimista é que os conjuntos de confiança são construídos independentemente uns dos outros, o que pode resultar em múltiplos estados com alta confiança.
 - Consequência da Exploração Otimista
 - Isso leva a explorações desnecessárias, pois o agente visita estados que não conduzem à recompensa, e corrigir esse problema seria computacionalmente inviável.
 - Vantagens dos Métodos de Incerteza
 - Nos métodos baseados em incerteza, os limites de confiança são construídos de maneira interdependente, evitando o problema de múltiplos estados com alta confiança que não contribuem para alcançar recompensas.
- 3.5. Aprendizado por Imitação

- A exploração é iniciada com demonstrações de diferentes fontes (geralmente humanas), semelhante ao aprendizado humano
- Imitação como Complemento ao Aprendizado por Reforço
 - O aprendizado por imitação pode complementar o aprendizado por reforço padrão, fornecendo um ponto de partida para o agente.
- Demonstrações como Ponto de Partida
 - As demonstrações não precisam ser perfeitas; basta que sejam um bom ponto de partida para o agente.
- Categorias de Aprendizado por Imitação
 - Pode ser dividido em:
 - (i) imitação em experience replay e
 - (ii) imitação com uma estratégia de exploração
- 3.5.1. Imitations in Experience Replay Methods (Combinação de Amostras de Demonstrações e do Agente)
 - Uma técnica comum é combinar amostras de demonstrações com aquelas coletadas pelo agente em um único experience replay, garantindo que a imitação possa ser usada ao longo do processo de aprendizado, enquanto novas experiências são adquiridas.
 - Replays de Experiência Priorizados
 - Demonstrações são armazenadas em um experience replay priorizado, junto com a experiência do agente, onde as transições de demonstrações têm uma maior probabilidade de serem selecionadas.
 - Deep Q-learning from Demonstration (DQfD)
 - O agente é pré-treinado exclusivamente com demonstrações, e a proporção entre amostras do agente e das demonstrações é controlada por um parâmetro.
 - Armazenamento em Dois Replays
 - Abordagens como R2D2 e outras armazenam estados em dois replays separados, permitindo ao agente amostrar de cada um para o aprendizado.
 - Discussão sobre o Uso de Um ou Dois Replays
 - Usar um ou dois experience replays tem impacto mínimo no desempenho, mas armazenar em um único replay é mais simples conceitualmente e na implementação, permitindo que o agente interrompa o uso de experiências de imitação quando não são mais necessárias.



- 3.5.2. Métodos de Imitação com Estratégia de Exploração
 - Em vez de experience replays, a imitação é combinada diretamente com estratégias de exploração, onde a imitação serve como um “pontapé inicial” para a exploração.
 - Demonstrações como Ponto de Partida
 - Uma única demonstração pode ser usada como ponto inicial para a exploração, com o agente explorando aleatoriamente a partir de um estado em paralelo com a demonstração. Esse método mostrou desempenho elevado em jogos como Montezuma’s Revenge.
 - Recompensa Auxiliar
 - Aytar et al. [26] propuseram um método que combina vídeos do YouTube em um único espaço de embeddings para treinamento. A recompensa auxiliar é adicionada a cada N quadros do vídeo de demonstração.
 - Agentes que Pedem Ajuda
 - Em outra abordagem, o agente pode pedir ajuda ao demonstrador humano se detectar um ambiente desconhecido, permitindo que o humano mostre o caminho.
 - Discussão sobre o Desempenho de Iniciação por Demonstração
 - Usar imitações para iniciar a exploração mostrou desempenho impressionante em jogos difíceis, superando o desafio inicial de exploração com demonstrações. A abordagem de [26] permite que o agente pontue bem com apenas uma demonstração, sendo muito eficiente em termos de amostragem e adequada para problemas com múltiplas fontes de demonstração.
- 3.5.3. Summary
 - Imitação em Experience Replay

- Permite que o agente aprenda de forma contínua e integrada a partir de experiências de demonstração.
- Imitação com Estratégias de Exploração
 - Tem o potencial de encontrar boas estratégias novas ao redor das existentes, mostrando grande capacidade para superar a dificuldade inicial de exploração.
- Desempenho Comparativo
 - Métodos de imitação com estratégias de exploração alcançam melhor desempenho do que a imitação apenas em experience replay, tornando-os mais eficazes em ambientes desafiadores.
- 3.6. Exploração Segura
 - Foca em evitar comportamentos inseguros dos agentes, garantindo sua segurança durante a exploração.
 - Categorias de Exploração Segura
 - Pode ser dividida em três categorias:
 - (i) conhecimento do projetista humano,
 - (ii) modelo de predição, e
 - (iii) recompensa auxiliar.
 - Conhecimento do Projetista Humano
 - Comportamentos do agente são limitados por restrições projetadas por humanos, definindo fronteiras seguras para o agente.
 - Modelos de Predição
 - O agente aprende comportamentos inseguros e como evitá-los, utilizando modelos de predição.
 - Recompensas Auxiliares
 - Os agentes são penalizados ao entrar em estados perigosos, incentivando comportamentos mais seguros.

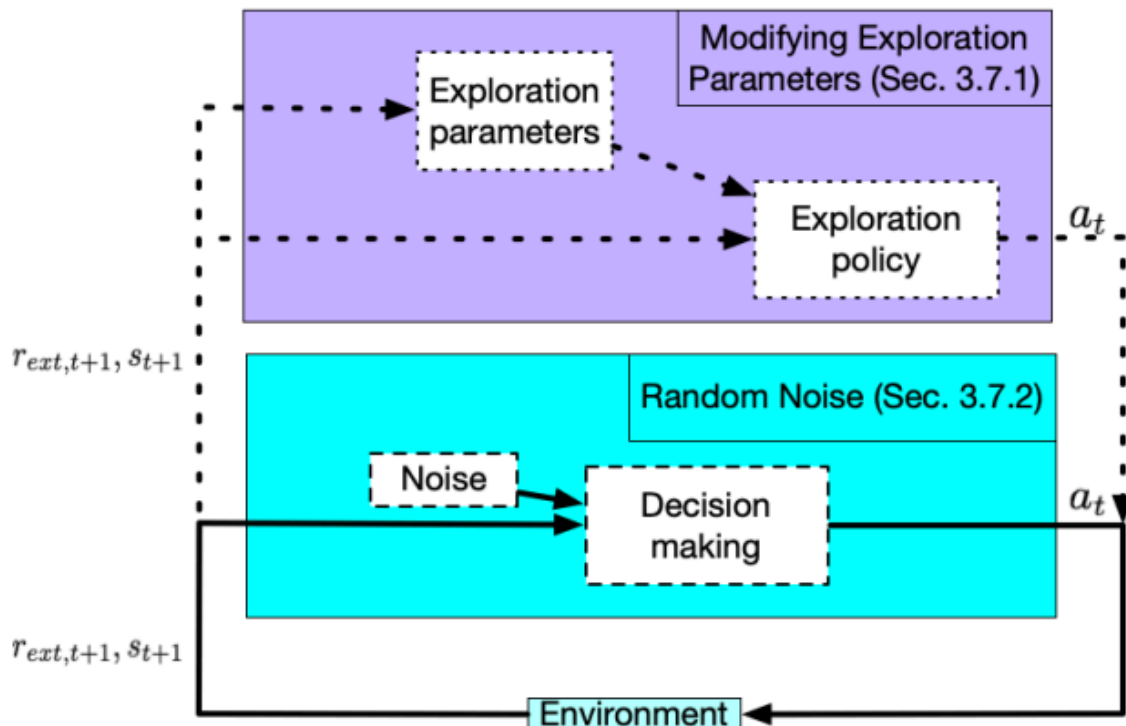


- 3.6.1. Métodos de Conhecimento do Projetista Humano

- Utilizam barreiras de segurança designadas por humanos, dividindo-se em comportamentos de base, intervenção humana direta e modelos de predição.
- Comportamentos de Base
 - Impõem uma linha de segurança intransponível. Exemplo: uma função de risco que identifica estados inseguros e define comportamentos de base para ações nesses estados. Alguns agentes são limitados por módulos pré-treinados para evitar ações perigosas.
- Intervenção Humana
 - Durante as fases iniciais de exploração, humanos intervêm para evitar desastres. Após coleta de dados, uma rede supervisionada treina com base nessas intervenções, substituindo os humanos no processo.
- Modelos de Predição
 - Determinam se a próxima ação do agente leva a um estado inseguro, evitando essas ações. Exemplo: o modelo de processo Gaussiano estima a altura segura de um terreno para navegação de robôs, e um modelo de segurança heurístico propõe o algoritmo de action pruning para evitar ações inseguras.
- Discussão sobre Métodos de Conhecimento Humano
 - Métodos de comportamentos de base e intervenção humana garantem segurança em situações predefinidas. Já os modelos de predição, que podem ser matemáticos ou heurísticos, são mais adaptáveis a ambientes desconhecidos e possuem maior capacidade de generalização.
- 3.6.2. Métodos de Recompensa Auxiliar
 - Nesses métodos, o agente é penalizado ao se colocar em situações perigosas, exigindo pouca intervenção humana, mas gerando comportamentos de segurança mais fracos.
 - Penalidade por Distância até a Morte
 - Em um método, o agente é penalizado ao se aproximar de estados que levam ao término do episódio. A abordagem impõe uma penalidade com base na distância até a morte, desencorajando ações arriscadas.
 - Aprendizado de Estados Indesejáveis
 - O agente aprende autonomamente a evitar estados indesejáveis usando experiências anteriores. Estados com

baixas vantagens são armazenados e recebem recompensas negativas adicionais.

- Discussão sobre Recompensas Auxiliares
 - Recompensas auxiliares são eficazes para desencorajar comportamentos inseguros. Exemplo: em [141], metade das mortes do agente foi evitada. No entanto, pressupõe-se que o agente obtenha uma pontuação baixa ao "perecer", o que pode não ocorrer em todos os casos.
- 3.6.3. Summary
 - As abordagens são divididas em:
 - (i) conhecimento do projetista humano,
 - (ii) modelos de predição e
 - (iii) recompensas auxiliares.
 - Conhecimento do Projetista Humano
 - Garante comportamentos seguros em determinados estados, mas o agente encontra dificuldades para aprender novos comportamentos seguros de forma autônoma.
 - Recompensas Auxiliares
 - Podem se ajustar a novos cenários, mas exigem tempo de treinamento e um design cuidadoso da recompensa negativa para efetividade.
- 3.7. Métodos Baseados em Aleatoriedade
 - Focam em melhorar a exploração aleatória simples, que tende a ser ineficiente ao revisitar os mesmos estados.
 - Três Abordagens de Melhoria
 - (i) redução de estados/ações para métodos de exploração,
 - (ii) métodos de parâmetros de exploração e
 - (iii) métodos de ruído nos parâmetros de rede.
 - Redução de Estados e Ações
 - A exploração é otimizada ao reduzir o número de ações a serem tomadas aleatoriamente, evitando ações repetitivas.
 - Modificação dos Parâmetros de Exploração
 - A exploração é automaticamente ajustada com base em parâmetros que regulam o comportamento exploratório do agente.
 - Ruído nos Parâmetros de Rede
 - Introduce ruído nos parâmetros da política, permitindo uma exploração mais diversificada sem mudar diretamente as ações do agente.



- 3.7.1. Métodos de Parâmetros de Exploração
 - A exploração é parametrizada (como o parâmetro ϵ -greedy), e esses parâmetros são ajustados conforme o progresso de aprendizado do agente.
 - Ajustes Baseados na Recompensa e na Incerteza
 - Uma técnica simples ajusta a taxa de exploração com base na recompensa. Em ambientes de recompensa escassa, ϵ
 - pode depender do erro nas estimativas da função de valor ou da entropia do modelo ambiental.
 - Dependência da Taxa de Aprendizado
 - A taxa de aprendizado também pode depender da exploração. Por exemplo, um parâmetro α reduz o ritmo de aprendizado em caso de alta exploração para lidar com a incerteza.
 - Controle por Recompensas de Curto e Longo Prazo
 - A exploração é aumentada quando a média de recompensas de curto prazo está abaixo da média de longo prazo, incentivando o agente a procurar novas ações.
 - Ajustes por Múltiplos Agentes
 - Vários agentes (como formigas) ajustam os parâmetros de exploração com base em valores de feromônio deixados por outras formigas, influenciando as escolhas de ações.

- Redução de Estados para Exploração
 - O adaptive resonance theorem (ART) divide o conhecimento sobre ações em três categorias (positivas, negativas e não exploradas) e escolhe ações aleatoriamente de estados com recompensas positivas ou ainda não explorados, otimizando a eficiência da amostragem.
- Discussão sobre Ajustes de Parâmetros
 - Diferentes parâmetros podem ser ajustados conforme o aprendizado. Isso evita que o agente aprenda informações incertas, mas pode desacelerar o treinamento. Reduzir estados para exploração aumenta a eficiência, embora possa dificultar a descoberta de novos estados que apareçam posteriormente.
- 3.7.2. Métodos de Ruído Aleatório
 - O ruído aleatório é utilizado para explorar, podendo ser aplicado nos parâmetros da rede ou gerado com base nos estados encontrados durante a exploração.
 - Ruído Fixo
 - O método mais simples é adicionar uma quantidade fixa de ruído aos parâmetros. Pequenas perturbações no espaço de parâmetros podem melhorar a exploração.
 - Redes Caóticas e Ajuste do Ruído
 - Redes caóticas são usadas para induzir ruído, e a força do ruído pode ser ajustada por backpropagation, onde uma fonte de ruído constante é multiplicada por um parâmetro ajustável pelo gradiente.
 - Comparação de Políticas com e sem Ruído
 - Outra abordagem compara decisões da política com e sem ruído. Se as decisões forem suficientemente diferentes, a exploração é aplicada.
 - Geração de Ações Aleatórias Dependentes do Estado
 - Para resolver o problema de atribuição de crédito em estados repetidos, uma função de geração de ações aleatórias dependente do estado foi desenvolvida, garantindo que a mesma ação seja tomada para o mesmo estado.
 - Discussão sobre Ruído nos Parâmetros da Rede
 - Originalmente usado em abordagens evolutivas, o ruído nos parâmetros foi adotado em métodos baseados em políticas. Em [155], o ruído nos parâmetros resultou em uma melhoria de 50% em média em 52 jogos do Atari.
- 3.7.3. Summary
 - Vantagens da Redução de Estados para Exploração

- Métodos que reduzem estados para exploração são eficazes, mas exigem memória para registrar os estados já visitados.
- Métodos de Parâmetros de Exploração
 - Equilibram bem a exploração e a exploração; contudo, o agente ainda pode ficar preso explorando estados desnecessários.
- Abordagens de Ruído Aleatório
 - Simples de implementar e mostram resultados promissores, mas dependem de uma sintonia cuidadosa dos parâmetros por parte dos projetistas.
-
- 4 - Desafios Futuros
 - A exploração no aprendizado por reforço enfrenta desafios de avaliação, escalabilidade, dilema exploração-exploração, recompensa intrínseca, problemas de TV com ruído, exploração segura e transferibilidade.
 - Avaliação
 - Avaliar e comparar algoritmos de exploração é difícil devido à falta de benchmarks e métricas comuns. Embora existam grandes benchmarks, como VizDoom e Atari Games, as diferenças de complexidade entre eles tornam a comparação de algoritmos complexa.
 - Escalabilidade
 - Métodos atuais exigem milhões de amostras e são difíceis de aplicar em problemas do mundo real. Para se adaptar, a eficiência da amostragem deve aumentar, e a representação de estados precisa ser otimizada para grandes domínios.
 - Dilema Exploration-Exploitation
 - Embora muitos métodos lidem com esse dilema, ele ainda é amplamente ajustado manualmente. Soluções automatizadas, como o aprendizado de habilidades que podem ser combinadas, oferecem um caminho promissor.
 - Recompensa Intrínseca
 - Melhorias na recompensa de estados inéditos e comportamentos diversos incluem a liberdade do agente para se recompensar e o equilíbrio entre novidade de longo e curto prazo, que atualmente requer ajuste manual.
 - Problema de TV com Ruído
 - O problema de "noisy-TV" permanece em grande parte sem solução; métodos baseados em memória são limitados e o uso de agrupamento para evitar estados ruidosos tem potencial, mas exige uma correta definição de clusters.

- Exploração Ótima e Exploração Segura
 - A exploração ótima e segura são áreas críticas. A exploração ótima visa evitar revisitas desnecessárias, enquanto a exploração segura exige uma definição mais rigorosa de situações inseguras para melhorar a escalabilidade e a transferibilidade.
- Transferibilidade
 - A maioria das abordagens de exploração é limitada ao domínio treinado. Para enfrentar novos ambientes, é necessário que o agente transfira habilidades de cenários menores para maiores, beneficiando a adaptação em domínios variados.
- 5. Conclusões
 - Este artigo revisa métodos de exploração no aprendizado por reforço, incluindo: recompensa de estados inéditos, recompensa de comportamentos diversos, métodos baseados em objetivos, incerteza, imitação, exploração segura e métodos aleatórios.
 - Recompensa por Estados Inéditos
 - O agente recebe recompensa por descobrir estados novos ou surpreendentes. Métodos de erro de previsão recompensam com base na precisão do modelo ambiental interno do agente; métodos baseados em contagem recompensam pela frequência de visitas a um estado; métodos baseados em memória recompensam pela diferença entre o estado atual e outros em um buffer.
 - Recompensa por Comportamentos Diversos
 - O agente é recompensado por explorar uma variedade de comportamentos. Isso inclui estratégias evolutivas, que incentivam a diversidade entre agentes, e aprendizado de políticas, que incentiva a diversidade nos parâmetros da política.
 - Métodos Baseados em Objetivos
 - O agente explora a partir de um objetivo ou enquanto tenta alcançá-lo. A exploração de um objetivo resulta em visitas a áreas predominantemente desconhecidas, enquanto a exploração orientada por objetivo oferece metas específicas para exploração.
 - Métodos Probabilísticos
 - O agente utiliza um modelo de incerteza para tomar decisões. Métodos otimistas seguem o princípio do otimismo sob incerteza, enquanto métodos de incerteza utilizam amostras para explorar áreas menos conhecidas.
 - Métodos Baseados em Imitação
 - Utilizam demonstrações para facilitar a exploração. Podem combinar amostras de demonstrações em experience replay ou usá-las como

ponto de partida para outras técnicas de exploração, como recompensa por estados inéditos.

- Exploração Segura
 - Foca em garantir comportamentos seguros, com abordagens que utilizam conhecimento de projetistas humanos, modelos de predição para evitar movimentos desastrosos, e penalidades por estados perigosos.
- Métodos de Exploração Aleatória
 - Melhoram a exploração aleatória padrão, modificando estados para exploração, ajustando parâmetros de exploração, e aplicando ruído nos parâmetros da rede.
- Comparação de Desempenho e Implementação
 - Métodos como recompensa por estados inéditos, comportamentos diversos e exploração aleatória são mais fáceis de implementar e exigem menos recursos computacionais. Métodos baseados em objetivos e estados inéditos são os mais eficazes em problemas desafiadores, mas os métodos baseados em objetivos são mais complexos de implementar. Métodos de recompensa por estados inéditos equilibram bem simplicidade e desempenho.

Análise de bibliotecas e *frameworks* de aprendizado por reforço

Nome da ferramenta	Descrição	Aplicação	Vantagens	Desvantagens	Link para a documentação
OpenAI's Gym	Biblioteca de ambientes para desenvolvimento e teste de algoritmos de RL.	Criação de ambientes de RL para simulação e testes.	Simplicidade e ampla adoção na comunidade.	Limitado em alguns tipos de ambientes complexos.	https://www.gymnasium.ml/
TensorFlow's TF-Agents	Biblioteca de RL desenvolvida sobre TensorFlow com implementações de algoritmos comuns.	Treinamento de agentes de RL em TensorFlow.	Integração com TensorFlow e recursos de fácil customização.	Dependente de TensorFlow, o que pode ser menos eficiente para alguns usuários.	https://www.tensorflow.org/agents

PyTorch RLlib	Biblioteca de RL em PyTorch que oferece suporte para treinamento distribuído de agentes.	Implementação de RL distribuído e escalável.	Suporte para várias arquiteturas e ambientes, fácil de integrar com PyTorch.	Complexidade para iniciantes.	https://docs.ray.io/en/latest/rllib/index.html
Stable Baselines 3	Conjunto de algoritmos de RL baseado em PyTorch, fácil de usar e focado na simplicidade e estabilidade.	Treinamento de modelos de RL de forma rápida e fácil.	Fácil de implementar e bom suporte de comunidade.	Suporte limitado para RL distribuído.	https://stable-baselines3.readthedocs.io/
ReAgent by Meta	Plataforma de RL de código aberto desenvolvida pela Meta para soluções de RL personalizadas.	Personalização de algoritmos para aplicações empresariais.	Flexibilidade e foco em aprendizado contextual.	Comunidade menor em comparação a outras ferramentas mais amplamente adotadas.	https://reagent.ai/
DeepMind's OpenSpiel	Biblioteca de jogos e algoritmos de RL para pesquisa em ambientes de jogos.	Pesquisa em RL aplicada a jogos.	Ampla variedade de jogos e algoritmos para pesquisa acadêmica.	Foco em jogos, o que pode ser limitante para outros tipos de aplicações de RL.	https://github.com/deepmind/open_spiel
Amazon SageMaker RL	Plataforma de RL da Amazon integrada ao SageMaker para criação e treinamento de modelos em larga escala.	Implementação de RL em nuvem com recursos de escalabilidade.	Integração com AWS, fácil de escalar e gerenciar.	Custo elevado devido ao uso de recursos de nuvem da AWS.	https://aws.amazon.com/pt/sagemaker/reinforcement-learning/
Dopamine	Biblioteca de RL desenvolvida pelo Google com foco em aprendizado	Desenvolvimento de agentes de RL focados em DQN.	Simplicidade e boa documentação para DQN.	Limitado em diversidade de algoritmos, focado em DQN.	https://github.com/google/dopamine

	por reforço simplificado.				
CleanRL	Coleção de implementações de RL de código aberto focadas em simplicidade e desempenho.	Implementação rápida de algoritmos padrão de RL.	Leve, fácil de usar e bem documentado.	Não foca em escalabilidade ou soluções distribuídas.	https://github.com/vwxyzjn/cleanrl
Mava	Framework de RL multiagente baseado em JAX, criado para pesquisa e desenvolvimento em RL cooperativo.	Implementação de algoritmos para ambientes multiagente.	Suporte a ambientes multiagente e arquitetura escalável.	Requer familiaridade com JAX, o que pode ter uma curva de aprendizado.	https://github.com/instadeepai/Mava
pearl from meta	Implementação de RL meta-aprendizado, desenvolvida para acelerar a adaptação de agentes a novos ambientes.	Meta-aprendizado em RL, foco em adaptação rápida.	Flexibilidade e eficiência para problemas de meta-aprendizado.	Requer conhecimento avançado em meta-aprendizado, comunidade menor.	https://github.com/facebookresearch/pearl

Análise de ambientes de aprendizado por reforço

Nome do ambiente	Descrição	Aplicação	Link para a documentação
OpenAI Gym	Biblioteca de ambientes para desenvolvimento e teste de algoritmos de RL.	Simulações gerais para algoritmos de RL.	https://farama.org/projects
Gym Trading	Ambiente de simulação de mercados financeiros para teste de estratégias de negociação.	Treinamento de agentes para negociação de ativos financeiros.	https://github.com/Kismuz/gym-trading

TensorTrade	Framework de código aberto para criação de ambientes de negociação financeira.	Simulações de mercado para criação de estratégias de trading.	https://tensortrade.org/
VIZDoom	Ambiente de RL para simulações de jogos de tiro em primeira pessoa.	Pesquisa e treinamento de RL em jogos de tiro.	https://github.com/mwydmuch/ViZDoom
OpenSpiel	Ambiente para pesquisa em RL em jogos de tabuleiro e estratégia.	Treinamento de RL em jogos de tabuleiro e estratégia.	https://github.com/deepmind/open_spiel
ns3-gym	Integra ns3 (simulador de redes) com OpenAI Gym para experimentos de RL.	Experimentos de RL em simulações de redes.	https://github.com/tkn-tub/ns3-gym
RecoGym	Ambiente para RL focado em simulação de sistemas de recomendação.	Simulação de sistemas de recomendação para RL.	https://github.com/criteo-research/reco-gym
OpenSim-RL	Plataforma de RL para simulação biomecânica em ambientes 3D.	Pesquisa em biomecânica e robótica com RL.	https://github.com/stanfordnbl/osim-rl
Textworld	Ambiente de RL para aprendizado baseado em textos e interações linguísticas.	Desenvolvimento de RL para interações baseadas em texto.	https://github.com/microsoft/textworld
Atari Learning Environments	Coleção de ambientes de jogos Atari para avaliação de agentes de RL.	Treinamento de agentes de RL em jogos clássicos.	https://github.com/openai/atari-py
Malmö	Plataforma para pesquisa em RL em ambientes de simulação 3D.	Treinamento de agentes de RL em ambientes virtuais.	https://www.microsoft.com/en-us/research/project/project-malmo/
Mujoco	Simulador de física para o desenvolvimento de robótica e biomecânica.	Pesquisa em robótica e biomecânica aplicada.	http://www.mujoco.org/

APÊNDICE 6

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: 31 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Fabrycio Leite Nakano Almada

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

Documentação do progresso da Semana: Reinforcement Learning 31/10

A Semana constitui-se em diversas experimentações e testes dos ambientes e frameworks escolhidos anteriormente.

- Stable-Retro
 - Alternativa mais “moderna” ao ALE
- Open-Spiel
 - Jogos de tabuleiro
- VIZDoom
 - Foco maior em exploração
- TextWorld
 - Jogos baseados em texto
- Gym-Trading
 - Simulação de trading
- ML-Agents
 - Biblioteca de RL da Unity
- Atari Learning Environments
 - Coleção de ambientes de jogos Atari para avaliação de agentes de RL

Artefatos produzidos:

- Gate6

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

- Aprofundamento das ferramentas escolhidas e primeiras experimentações de Exploration/Exploitation

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

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go! ▾

Artefatos produzidos neste Gate:

Atari Learning Environment

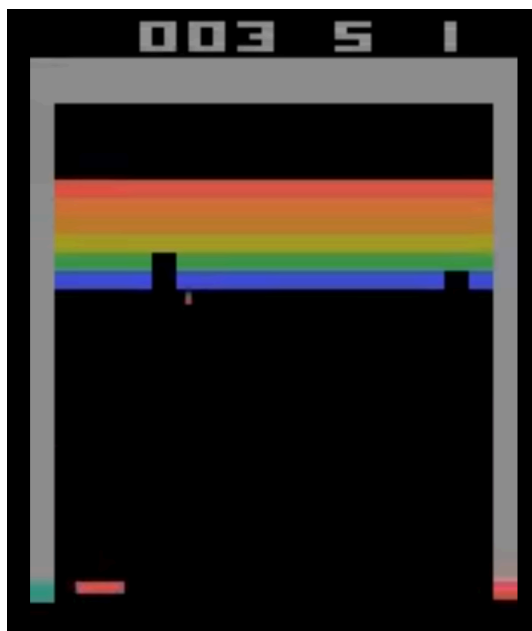
[Atari_learning_enviroment.ipynb](#)

rollout/	
ep_len_mean	1.36e+03
ep_rew_mean	5.82
time/	
fps	315
iterations	10
time_elapsed	259
total_timesteps	81920
train/	
approx_kl	0.06363768
clip_fraction	0.439
clip_range	0.2
entropy_loss	-1.12
explained_variance	0.773
learning_rate	0.0003
loss	-0.127
n_updates	90
policy_gradient_loss	-0.0777
value_loss	0.0227

Exemplo de Log de treino

Vídeo do modelo treinado:

 Breakout-agent.mp4



Frame do agente jogando *Breakout*

Stable-Retro:

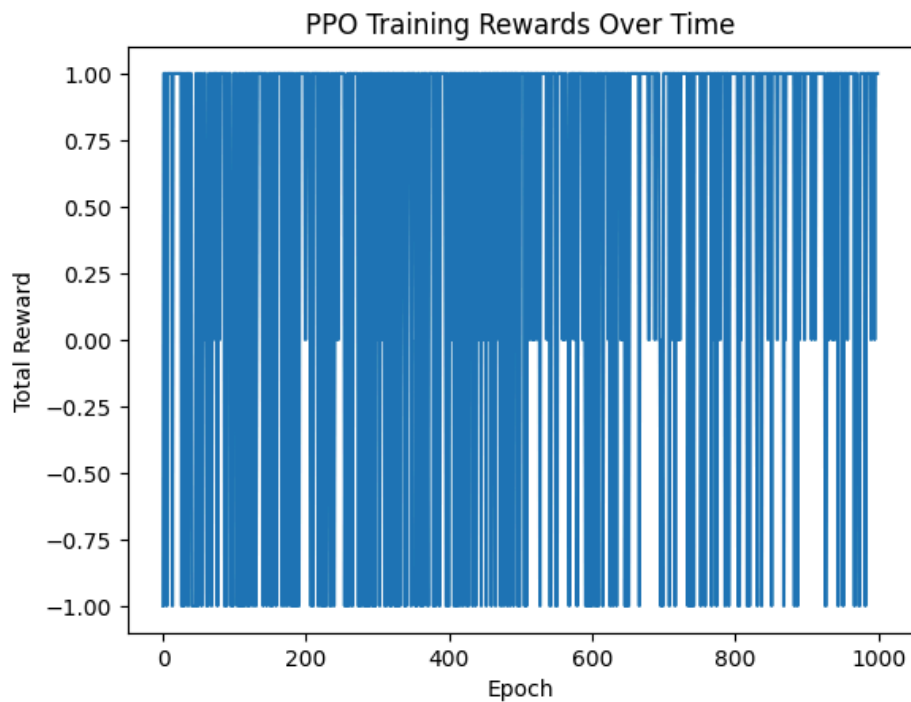
[stable-retro.ipynb](https://stable-retro.github.io)

time/	
fps	11
iterations	8
time_elapsed	1435
total_timesteps	16384
train/	
approx_kl	0.07409881
clip_fraction	0.46
clip_range	0.2
entropy_loss	-8.15
explained_variance	0.6138364
learning_rate	0.0003
loss	0.0569
n_updates	70
policy_gradient_loss	-0.0365
value_loss	2.58

Exemplo de Log de treino

Open-Spiel:

[Tic_tac_Toe_open_spiel.ipynb](#)



Desempenho do agente jogando “jogo da velha” (Tic-Tac-Toe)

Viz_Doom:

[Deep_Q_learning_with_Doom.ipynb](#)

[Hands_on_viz_Doom.ipynb](#)

Text_World:

[Interaction_with_TextWorld.ipynb](#)

[TextWorld_test1.ipynb](#)


```

      |_____| \ |_____| \ |_____| \ |_____| \
      \$$$$$$$| $$$$$$| $$ | $ $ \$$$$$$$
      | $$ | $$ | $$$\ $$$ | $$ | $$
      | $$ | $$ \ >$$ $ $ | $$
      | $$ | $$$$ / $$$$ \ | $$
      | $$ | $$$ | $$ $$$ \ | $$
      | $$ | $$$ \ | $$ | $ $ | $$
      \$$ \$$$$$$$ \$$ \$$ \$$

|_____| \ |_____| \ |_____| \ |_____| \
| $$ / \ | $$ | $$$$$$ \ $$$$$$ \ $ $ | $$$$$$ \
| $$ / $ \ | $$ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $
| $$ $$$ \ $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $
| $$ $$$ \ $ $ | $ $ | $ $ | $$$$$$ \ | $ $ | $ $
| $$$$ \$$$$$ | $ $ / $ $ | $ $ | $ $ | $ $ | $ $ / $ $
| $$$ \$$$ \$$$ \$$$ | $ $ | $ $ | $ $ | $ $ \ | $ $ $ $
\$$ \$$ \$$$ \$$$ \$$$ \$$$ \$$$ \$$$ \$$$ \$$$

```

Hey, thanks for coming over to the TextWorld today, there is something I need you to do for me. First step, retrieve the American limited edition keycard from the type 1 box. And then, unlock the American limited edition gate. After that, open the American limited edition gate. Then, make an effort to move east. Following that, pick-up the shirt from the floor of the attic. Got that? Good!

`-= Scullery =-`

You've stumbled into a normal room. Your mind races to think of what kind of room would be normal. And then it hits you. Of course. You're in the scullery. You begin to take stock of what's in the room.

You can make out a type 1 box. The type 1 box contains an American limited edition keycard.

There is a closed American limited edition gate leading east. There is a closed door leading south.

`>`

`-= Scullery =-0/1`

`> Go east`

You have to open the American limited edition gate first.

`>`

`-= Scullery =-0/2`

`> Open the Gate`

You have to unlock the American limited edition gate with the American limited edition keycard first.

Exemplo de aplicação do *TextWorld*

ML-Agents:

[ML_Agents_with_Colab.ipynb](#)

[\(bonus_train_with_huggy\)_Fabrycio_Leite_Nakano_Almada.ipynb](#)



Ambiente de teste do *Huggy* após agente devidamente treinado

APÊNDICE 7

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 out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Fabrycio Leite Nakano Almada

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

Divisão das tarefas em duas partes principais:

- **Parte 1:** Estudo, Experimentos (e documentação) de Exploration/Exploitation em Reinforcement Learning.
 - Implementação do algoritmo Deep Q-learning para o problema do Cartpole [1]
 -  RL - Cartpole.ipynb
 - Implementação e experimentação com o conceito de Multi-Armed Bandit, com o objetivo de explorar diferentes estratégias de exploração e exploração em problemas simples.[2]
 -  Exploration - e-greedy (multi arms bandit).ipynb
 - Documentação do processo:  Experimentos com Multi Armed Bandit
- **Parte 2:** Testes em ambientes como ML-Agents e TextWorld, com foco no tradeoff exploration/exploitation.
 - Análise do impacto da exploração e exploração nos ambientes de ML-Agents e TextWorld
 - Instalação do ambiente ML-agents no Windows (Descrição do ambiente ML-Agents e seu uso para testar algoritmos de RL)
 -  How to use Machine Learning AI in Unity! (ML-Agents)
 -  ML-Agents: Exploitation/Exploration.ipynb
 - Busca por uma maior compreensão do TextWorld e ao seu uso para experimentos com agentes em ambientes de texto.
 - [TextWorld: A Learning Environment for Text-based Games](#)
 - A Simple Game
 - ocorre em uma casa típica e consiste em encontrar o item de comida certo e cozinhá-lo.
 - Coin Collector

- Envolve um agente que deve atravessar uma cadeia de salas, evitando distrações, para coletar uma moeda no final.
- The Cooking Game
 - Localizar a cozinha, ler o livro de receitas, pegar os ingredientes, processá-los corretamente, preparar a refeição e comê-la.
- Treasure Hunter
 - O agente começa em um labirinto gerado aleatoriamente e deve encontrar um objeto específico mencionado no objetivo exibido quando o jogo começa. (Neural Map)
-  Exploration/Exploitation no Textworld.ipynb

Referências:

[1] (O'Reilly, A.. Mãos à Obra: Aprendizado de máquina com Sci-kit Learn, Keras e TensorFlow. 1. ed. Rio de Janeiro: Novatec, 2018.)

[2] [SUTTON, R. S.; BARTO, A. G. Reinforcement Learning: An Introduction. 2. ed. Cambridge: MIT Press, 2018.](#)

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

- Continuação dos experimentos com diferentes algoritmos de RL.
- Expansão dos testes no ML-Agents e TextWorld.
- Exploração de outras técnicas de RL avançadas para aprimorar a exploração e a exploração.

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

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: 

Análise do Trade-off Exploração-Exploração no Problema Multi-armed Bandit com Estratégias Epsilon-Greedy

Resumo

Este estudo investiga o trade-off entre exploração e exploração no contexto do problema *multi-armed bandit* usando uma estratégia epsilon-greedy. Foram testados diferentes valores de ϵ e número de braços k , observando-se como essas variáveis afetam o desempenho do agente em termos de média de recompensas e taxa de escolhas do braço ótimo. Os resultados mostram que um ϵ moderado é eficaz para alcançar um equilíbrio entre exploração e exploração, enquanto um maior número de braços torna mais desafiador encontrar o braço ótimo.

Introdução

O problema multi-armed bandit é amplamente utilizado para entender o dilema entre exploração e exploração em aprendizado por reforço. Esse dilema é caracterizado pela escolha entre explorar novas ações para ganhar mais conhecimento sobre o ambiente ou explorar ações já conhecidas que maximizem a recompensa. No problema multi-armed bandit, o agente deve escolher entre diferentes "braços" de uma máquina de apostas, cada um oferecendo uma recompensa com média desconhecida.

A estratégia utilizada foi o ϵ -greedy, onde o agente escolhe uma ação aleatória com probabilidade ϵ (exploração) e a ação com a maior recompensa estimada com probabilidade $1-\epsilon$ (exploração). Ademais, investigou-se como diferentes valores de ϵ e diferentes números de braços k impactam o desempenho.

Metodologia

1. Configuração do Problema Multi-Armed Bandit

Cada experimento simula um agente interagindo com um ambiente contendo k braços, cada um com uma recompensa verdadeira média desconhecida

para o agente. Essas recompensas foram geradas aleatoriamente a partir de uma distribuição normal.

2. Estratégia Epsilon-Greedy

A estratégia epsilon-greedy controla o balanceamento entre exploração e exploração. Com probabilidade ϵ , o agente seleciona um braço aleatório para explorar; com probabilidade $1-\epsilon$, escolhe o braço com a maior recompensa estimada.

3. Métricas de Avaliação

Duas métricas foram analisadas:

- **Recompensa Média:** Média cumulativa das recompensas ao longo dos passos.
- **Proporção de Escolha do Melhor Braço:** Percentual de vezes que o agente escolheu o braço com a maior recompensa média verdadeira.

4. Experimentos

Foram realizados dois conjuntos de experimentos:

- **Variação de ϵ :** Testou-se valores de $\epsilon=0.0, 0.01, 0.1, 0.2$ com $k=10$.
- **Variação de k :** Testou-se valores de $k=[5, 10, 15 \text{ e } 20]$ com $\epsilon=0.1$.

Resultados e Discussão

Experimento 1: Variação de ϵ

Os gráficos em anexo mostram como diferentes valores de ϵ afetam a recompensa média e a proporção de escolha do braço ótimo.

Análise dos Resultados para Diferentes Valores de ϵ

- **Recompensa Média:** Observa-se que valores de ϵ mais baixos (como 0.01) resultam em exploração limitada, com o agente estabilizando rapidamente em

um braço, potencialmente ignorando braços melhores. Valores moderados de ϵ (como 0.1) permitem que o agente explore o suficiente para encontrar o braço ótimo, enquanto valores altos (como 0.5) levam a uma queda na média de recompensa devido à exploração excessiva.

- **Proporção de Escolha do Melhor Braço:** Com valores moderados de ϵ (0.1 e 0.2), o agente se aproxima rapidamente de uma escolha ótima. Valores altos, como $\epsilon=0.5$, aumentam a exploração e resultam em uma menor frequência de escolha do braço ótimo.

Experimento 2: Variação de k

Os gráficos em anexo mostram como diferentes valores de k (número de braços) afetam a recompensa média e a proporção de escolha do braço ótimo.

Análise dos Resultados para Diferentes Valores de k:

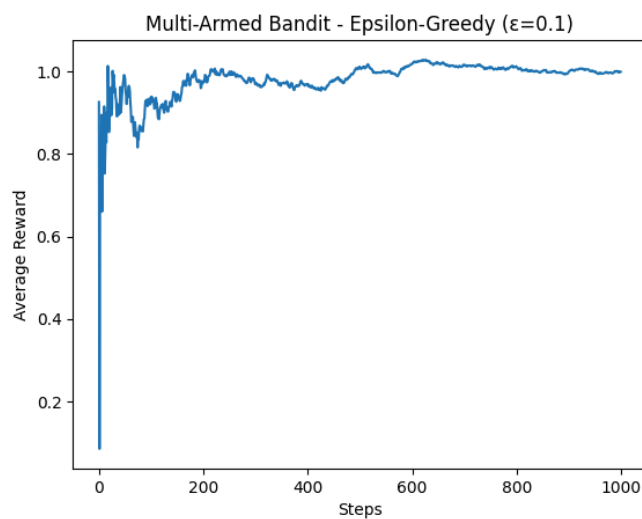
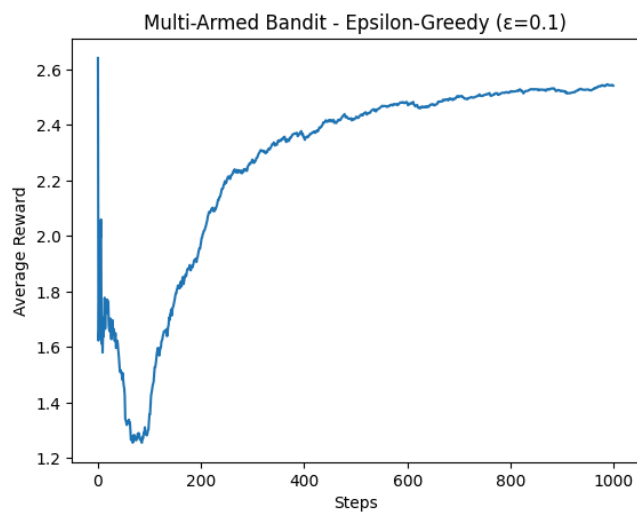
- **Recompensa Média:** Com o aumento do número de braços, o tempo para o agente estabilizar sua recompensa média aumenta. Com mais opções, o agente precisa explorar mais até identificar o melhor braço.
- **Proporção de Escolha do Melhor Braço:** Valores menores de k (como 5 e 10) permitem que o agente identifique rapidamente o braço ótimo. À medida que k aumenta, o processo de exploração se torna mais demorado, dificultando a convergência para a escolha ótima.

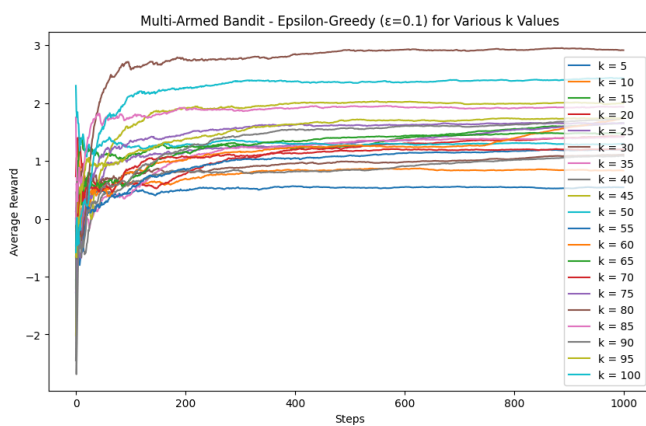
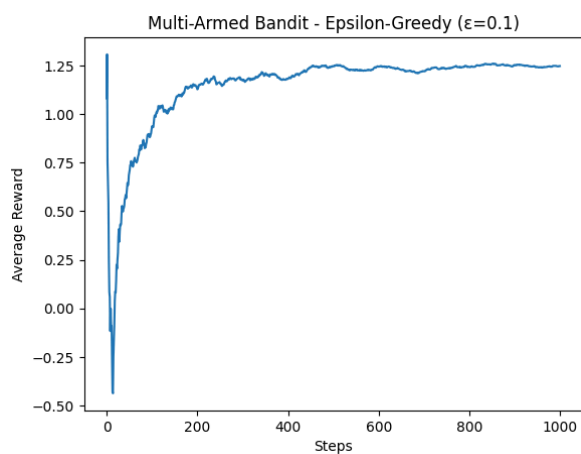
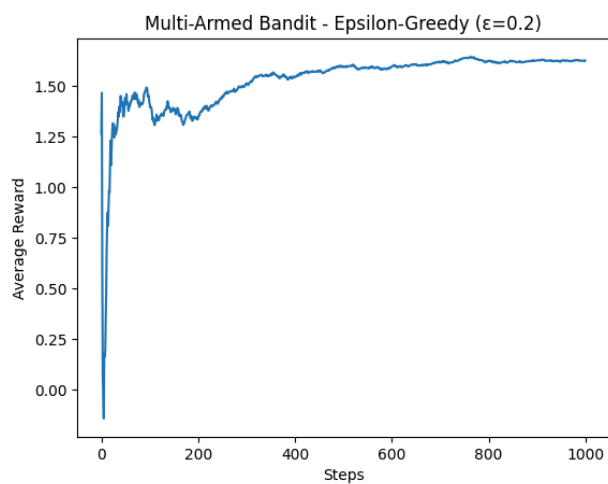
Conclusão

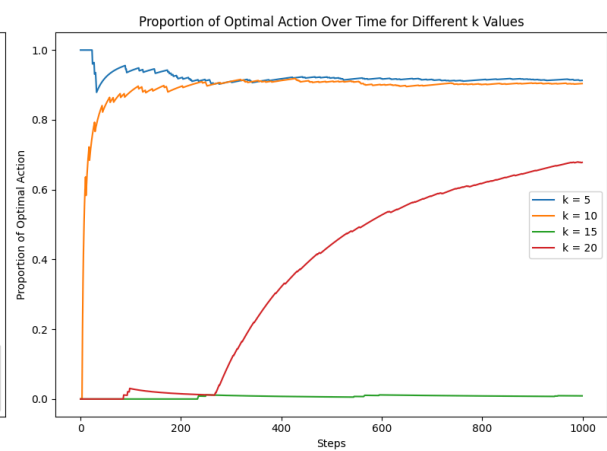
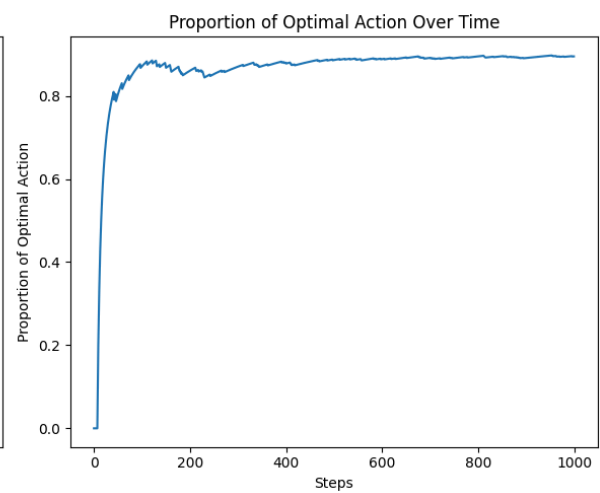
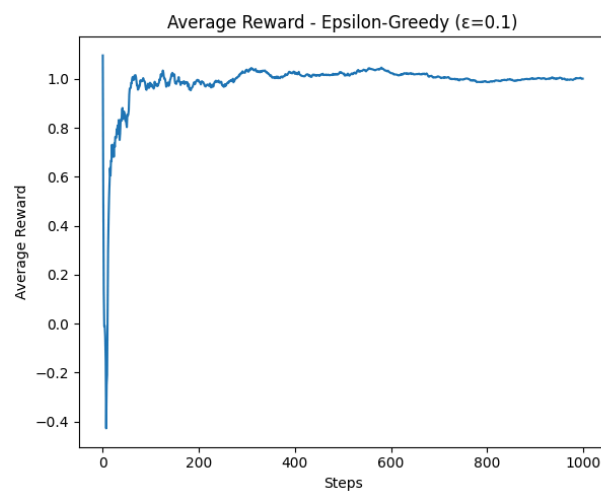
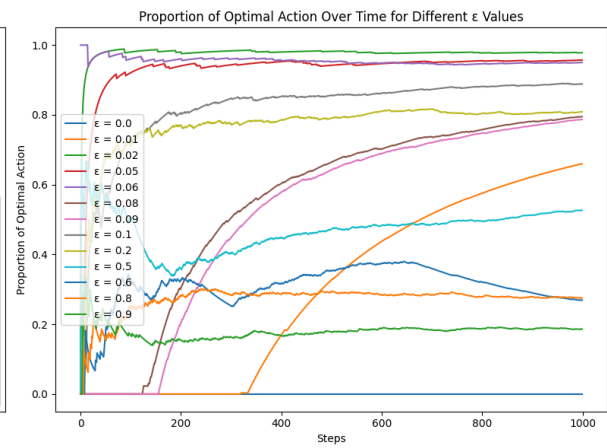
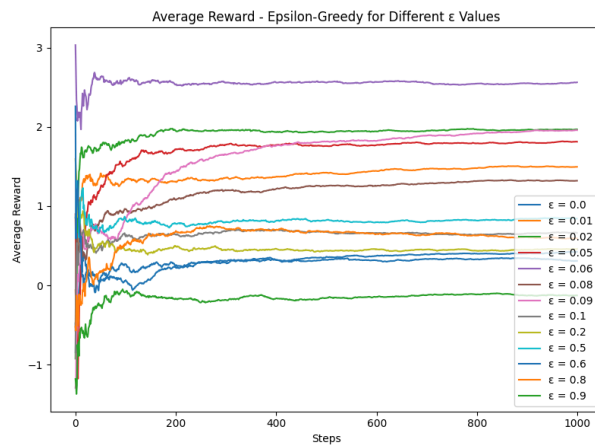
Este estudo demonstrou o impacto do valor de ϵ e do número de braços k no desempenho do agente no problema multi-armed bandit usando a estratégia epsilon-greedy. Observamos que um valor moderado de ϵ , como 0.1, proporciona um bom equilíbrio entre exploração e exploração. Um aumento no número de braços exige mais exploração para encontrar o braço ótimo, o que pode ser um desafio em ambientes com muitas opções.

Essas descobertas oferecem insights valiosos para a escolha de parâmetros em problemas de aprendizado por reforço, onde o equilíbrio entre exploração e exploração é essencial para alcançar o máximo desempenho a longo prazo.

Anexos:







APÊNDICE 8

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 nov. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Fabrycio Leite Nakano Almada

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

Durante a Semana, foram realizadas as seguintes atividades:

Avanços no TextWorld para exploração em ambientes de texto:

- Estudo aprofundado das possibilidades do TextWorld para construção de cenários de aprendizado baseados em texto, visando a criação de missões complexas para os agentes.
- Implementação de modificações para aumentar o desafio dos ambientes, explorando como os agentes se adaptam em tarefas com narrativas e estruturas mais ricas.
- Estudo e implementação de agente especializado em jogos baseados em texto:
 -  [LeDeepChef](#)  Deep Reinforcement Learning Agent for Families of Text-Based Ga...
 - [LeDeepChef Deep Reinforcement Learning Agent for Families of Text-Based Games](#)

Configuração do Ambiente de Desenvolvimento (Unity ML-Agents):

- Levantamento de materiais de estudo:
 - Tutoriais introdutórios
 - Vídeos explicativos
 - Exemplos práticos básicos

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

Documentação e organização dos experimentos:

- Criação de um repositório estruturado contendo os códigos, gráficos de desempenho, e análises dos experimentos realizados para facilitar a revisão e continuidade dos estudos.
- Produção de relatórios preliminares sobre a eficácia das técnicas aplicadas e comparações entre

algoritmos em cenários de exploração/exploração.

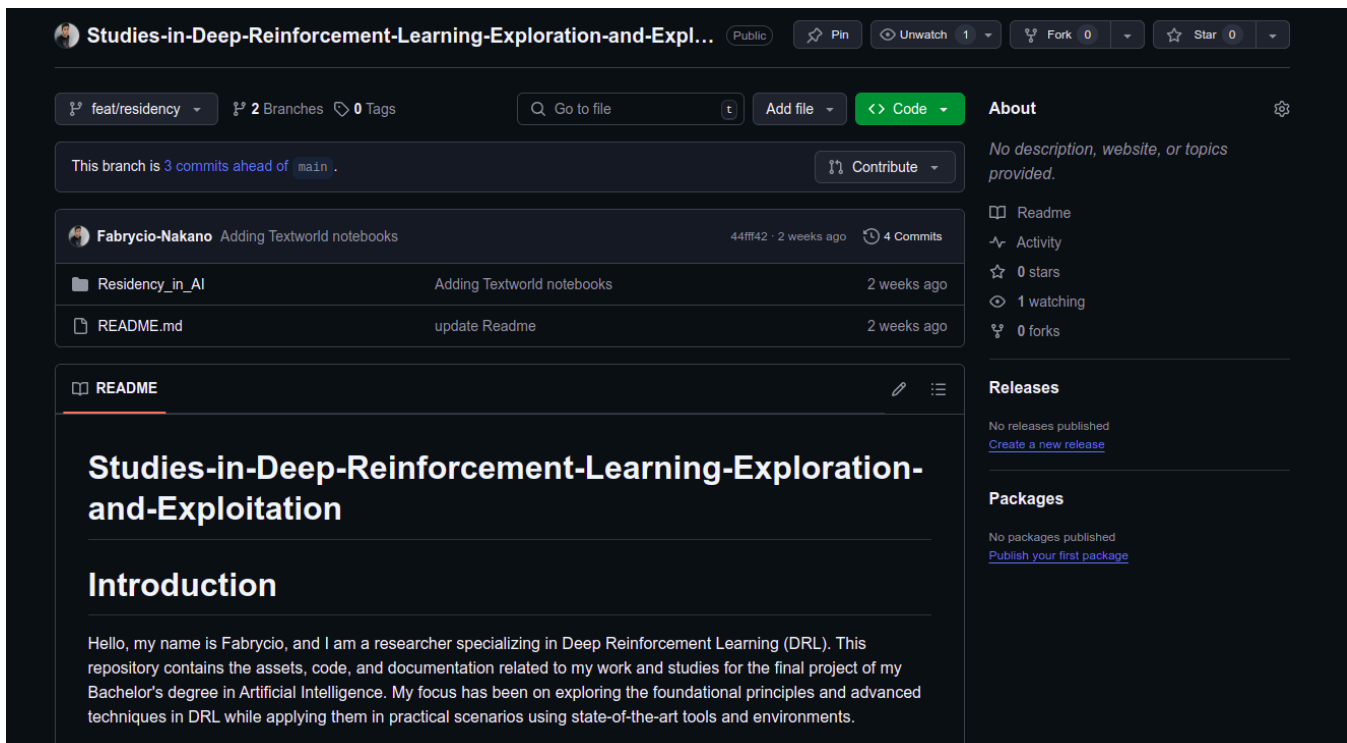
- Criação e adaptação de ambientes personalizados no Unity para simulações complexas, utilizando o ML-Agents como base para avaliar o comportamento dos agentes em situações diversificadas.
- Criação de configurações específicas para medir o impacto de parâmetros como taxa de aprendizado e discount factor no comportamento dos agentes.
- Revisão e implementação de scripts baseados em tutoriais avançados e exemplos práticos de ambientes complexos para melhorar a adaptabilidade dos agentes.

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

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go! ▾

Imagem do repositório criado no *Github*



Repositório Criado

APÊNDICE 9

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: 28 de nov. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Fabrycio Leite Nakano Almada

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

Gate Anterior:

- Avanços no TextWorld para exploração em ambientes de texto:
- Estudo e implementação de agente especializado em jogos baseados em texto:
 - LeDeepChef
- Configuração do Ambiente de Desenvolvimento (Unity ML-Agents):
 - Tutoriais, Vídeos explicativos, exemplos iniciais

Gate Atual:

- Criação do Repositório contendo os testes feitos no ML- Agents e no TextWorld:
 - <https://github.com/Fabrycio-Nakano/Studies-in-Deep-Reinforcement-Learning-Exploration-and-Exploitation/tree/feat/residency>
- Testes no ML-Agents (PPO,SAC,POCA):
 - Pushblock / PushBlock Collab
 - Desenvolver controle fino e interação com objetos no ambiente
 - Pyramids
 - Coleta de objetos ou ativação de mecanismos
 - WallJump
 - Habilidades de movimento e física avançadas, como saltos precisos.
 - SoccerTwos
 - Estratégias competitivas, cooperação dentro do time e habilidades motoras para chutar e defender
 - DungeonEscape
 - Habilidades de exploração e navegação em um espaço complexo.
 - O ambiente pode conter armadilhas ou penalidades.
 - Requer o aprendizado de caminhos ótimos e estratégias de evasão.
- Tentativa de aplicação de RL em Robô Hexápode

-  RealAnt robot takes reinforcement learning to the real world

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

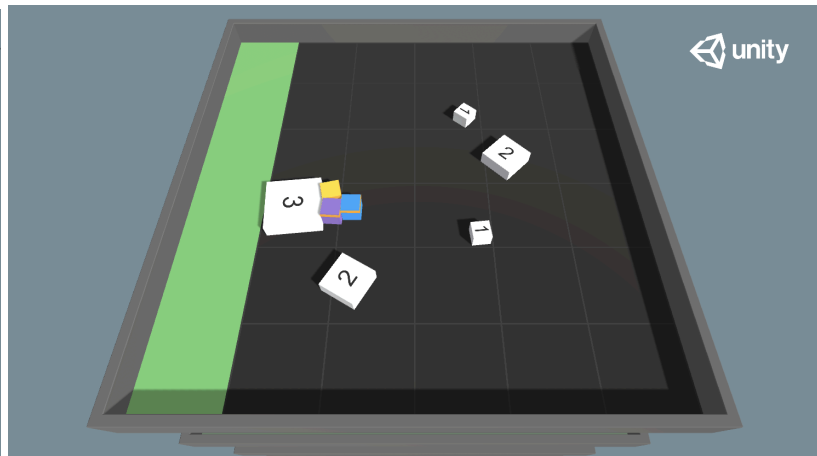
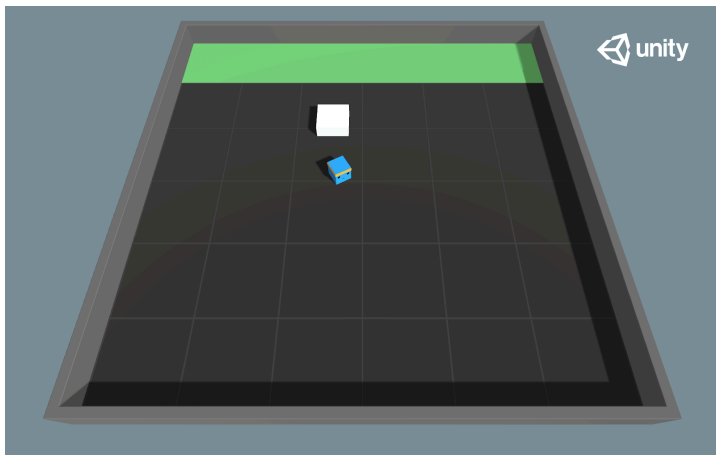
Estudar e replicar o ambiente do repositório: [Reinforcement learning with RealAnt: an open-source low-cost quadruped](#)

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

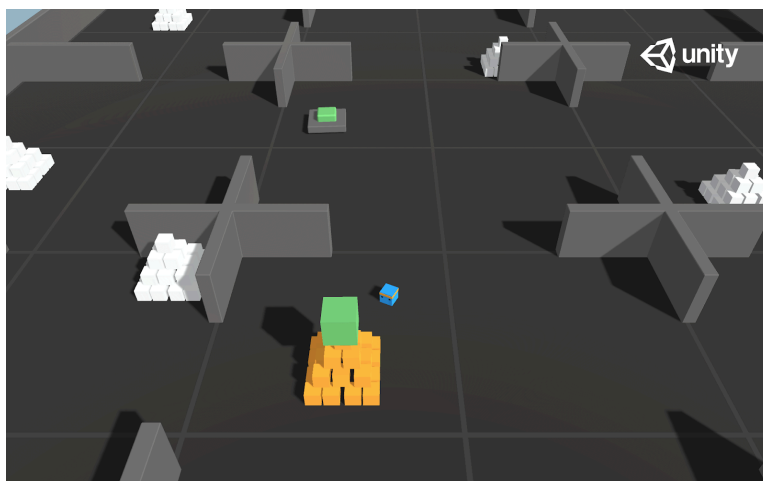
ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: 

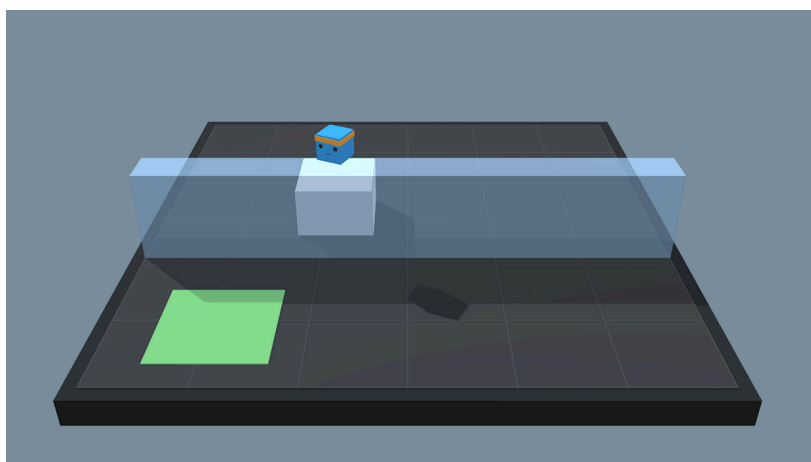
Exemplos os ambientes testados:



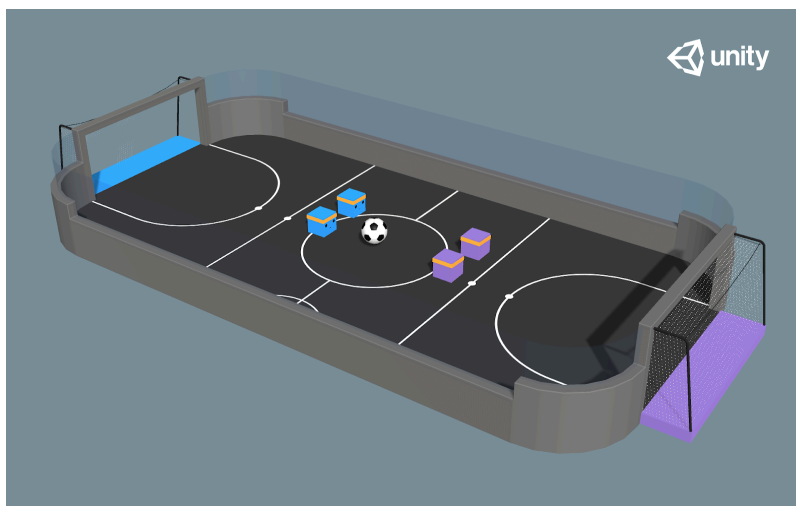
Exemplos dos Ambientes Pushblock e Pushblock Collab



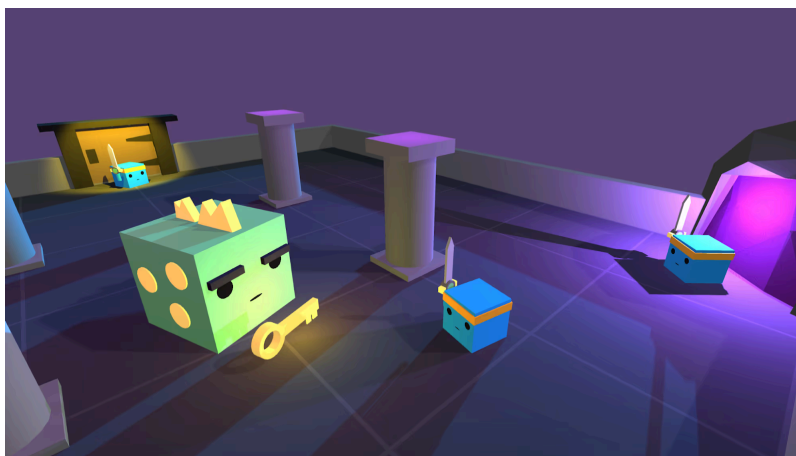
Exemplo do ambiente pyramids



Exemplo do ambiente Walljump



Exemplo do ambiente SoccerTwos



Exemplo do ambiente Dungeon Escape

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: 5 de dez. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

Fabrycio Leite Nakano Almada

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

Entregas do Gate 10 foram as seguintes:

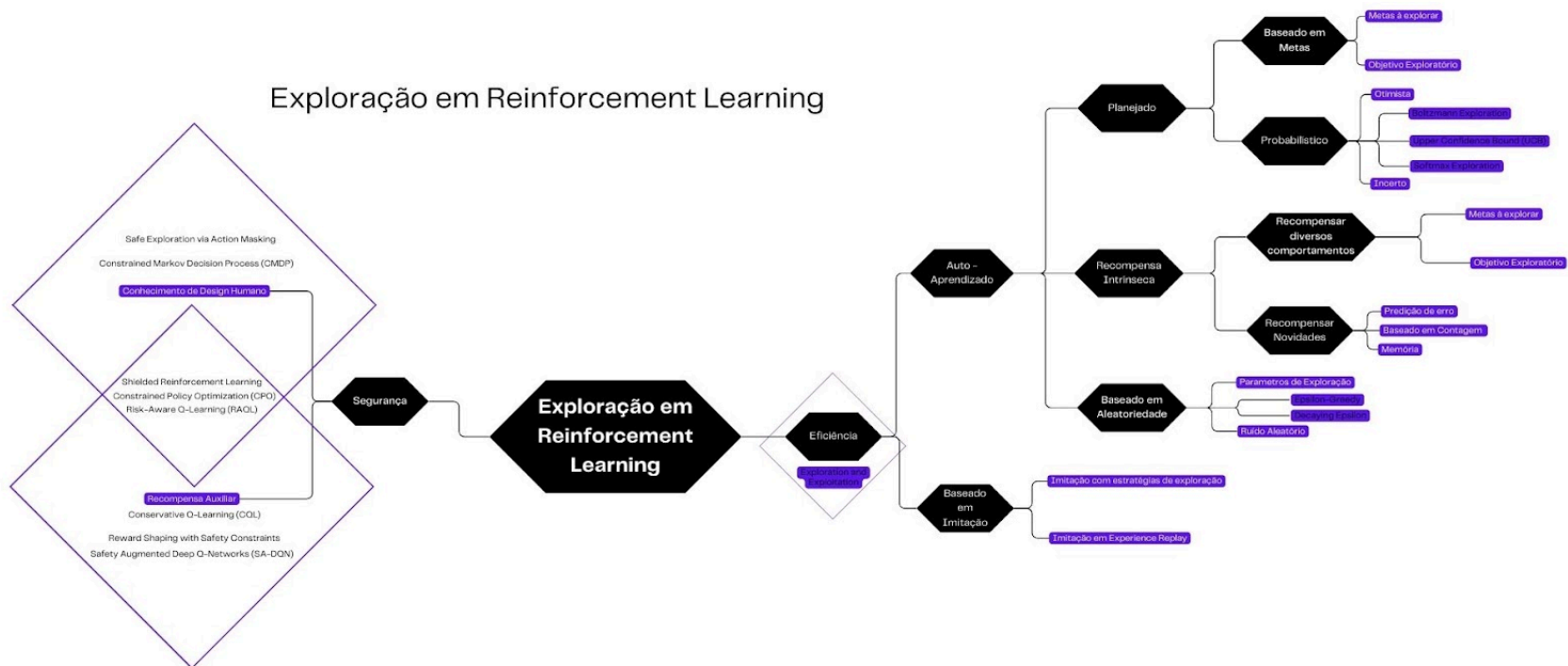
- Leitura do artigo: [RealAnt: An Open-Source Low-Cost Quadruped for Education and Research in Real-World Reinforcement Learning](#)
- Documentação e descrição do processo de reprodução, treinamento e análise do repositório *realant-rl*  *realant-rl*: Considerações
 - Reprodução do notebook no Colab:  *realant-rl(walk).ipynb*
- Revisão do repositório [A curated list of awesome exploration RL resources \(continually updated\)](#)
- Atualização da Taxonomia criada no Gate 5: [Exploration in Reinforcement Learning](#)

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: 



[Versão final: Taxonomia](#)

 `realant-rl(walk).ipynb`

RealAnt: Uma Plataforma de Baixo Custo para Pesquisa em Aprendizado por Reforço no Mundo Real

Resumo

O RealAnt é uma plataforma de pesquisa acessível e de código aberto projetada para explorar o aprendizado por reforço (RL) em robótica, com ênfase em métodos de exploração em ambientes simulados e no mundo real. Este artigo detalha as capacidades do RealAnt, as tecnologias envolvidas e os algoritmos de RL implementados, com foco em estratégias de exploração. A plataforma suporta motores físicos populares, como MuJoCo e PyBullet, e algoritmos modernos como SAC, TD3 e REDQ, oferecendo um ambiente robusto para pesquisas avançadas.

Introdução

A pesquisa em aprendizado por reforço frequentemente utiliza benchmarks simulados, como o ambiente "Ant", para avaliar o desempenho de algoritmos em tarefas de controle. No entanto, a lacuna entre simulação e implementação no mundo real pode dificultar avanços práticos. O RealAnt surge como uma solução inovadora: uma plataforma física de baixo custo (~350 euros) que combina acessibilidade e modularidade, permitindo o estudo de RL em cenários reais.

Este trabalho detalha o código-fonte, as implementações e os resultados obtidos com o RealAnt, com ênfase nos métodos de exploração implementados em algoritmos como Soft Actor-Critic (SAC), Twin Delayed Deep Deterministic Policy Gradient (TD3) e Randomized Ensembled Double Q-Learning (REDQ).

Estrutura e Componentes do RealAnt

Hardware

O RealAnt é um robô quadrúpede físico baseado em componentes eletrônicos acessíveis e um corpo 3D impresso. Seu design modular permite ajustes no hardware para testar diferentes configurações.

Software

O repositório inclui suporte para:

Simulação em MuJoCo e PyBullet.

Implementação de algoritmos de aprendizado por reforço utilizando PyTorch.

Scripts para visualização de episódios e análise de desempenho.

Algoritmos Implementados

O RealAnt suporta os seguintes algoritmos de RL, cada um focando em um aspecto específico do aprendizado eficiente:

Soft Actor-Critic (SAC)

O SAC é um algoritmo baseado em máxima entropia, projetado para melhorar a exploração. Ele incentiva o agente a explorar ações que maximizem tanto o retorno esperado quanto a entropia, promovendo escolhas diversificadas.

Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 introduz ruído na política para melhorar a exploração, usando uma arquitetura dupla para reduzir a superestimação do Q-valor.

Randomized Ensembled Double Q-Learning (REDQ)

O REDQ utiliza um ensemble de redes para melhorar a precisão na estimativa de Q-valores, permitindo exploração mais eficiente em ambientes com alta incerteza.

Métodos de Exploração no RealAnt

1. Exploração baseada em máxima entropia (SAC)

O SAC adota uma abordagem probabilística para explorar o espaço de ações. A política é ajustada para maximizar a entropia, incentivando o agente a explorar ações menos prováveis. Isso reduz o risco de ficar preso em ótimos locais.

2. Uso de ensembles no REDQ

O REDQ emprega múltiplas redes para calcular Q-valores, promovendo uma exploração mais segura ao evitar decisões baseadas em estimativas enviesadas. Isso permite uma busca mais profunda em ambientes desconhecidos.

3. Ruído adicionado no TD3

O TD3 aplica ruído gaussiano às ações durante a exploração, introduzindo variações que ajudam o agente a descobrir estratégias mais robustas.

Ambientes de Simulação

MuJoCo

O MuJoCo é um motor físico de alta precisão amplamente usado em RL. O RealAnt utiliza MuJoCo para simular o comportamento do robô em cenários controlados.

PyBullet

Como uma alternativa gratuita, o PyBullet oferece simulações realistas e suporte para diferentes configurações, permitindo maior acessibilidade ao sistema.

Resultados e Visualizações

O repositório inclui exemplos visuais de episódios e análises de desempenho. Resultados preliminares mostram que o REDQ supera outros algoritmos em ambientes complexos, enquanto o SAC se destaca em cenários de alta entropia.

Análise dos Resultados Obtidos nos Testes

Tendência Geral:

Os valores de retorno por episódio (Return) apresentam grande variabilidade nos episódios iniciais, alternando entre valores positivos e negativos. Isso é esperado no início do treinamento, quando o agente ainda está explorando o ambiente sem uma política otimizada.

À medida que o treinamento avança, os retornos começam a estabilizar em valores positivos, indicando uma melhora na política do agente.

Retornos Negativos nos Primeiros Episódios:

Nos episódios iniciais (ex.: 1, 4), os retornos negativos sugerem que o agente estava explorando ações que não resultaram em recompensas eficazes. Isso é típico em estágios iniciais de aprendizado por reforço, onde a exploração domina.

Estabilização Gradual:

A partir do episódio 15, os retornos mostram uma tendência a flutuar em torno de valores positivos pequenos, refletindo uma política inicial que começa a se formar. Episódios posteriores (ex.: 60–100) mostram retornos consistentemente positivos, sugerindo que o agente aprendeu uma política razoavelmente eficiente.

Oscilações Contínuas:

Mesmo com a estabilização geral, os valores continuam variando. Isso pode ser devido a uma combinação de:

- Ações exploratórias do agente (importante para evitar ótimos locais).

- Mudanças nas estimativas de Q-valores, especialmente em algoritmos como SAC e REDQ, que introduzem ruído para incentivar a exploração.

Recompensa de Avaliação (Eval Reward):

A recompensa de avaliação fornece uma média das performances do agente em episódios específicos. Observa-se uma tendência de crescimento até cerca do episódio 100, após o qual os valores flutuam em torno de um intervalo pequeno. Isso sugere que o agente convergiu para uma política estável, mas ainda está explorando melhorias incrementais.

Aviso de Conversão de Tensor:

O aviso sobre a conversão de listas de `numpy.ndarrays` para tensores indica que pode haver um gargalo de desempenho. Otimizar essa etapa convertendo diretamente para `numpy.array` antes da conversão para tensores pode acelerar o treinamento.

Logs e considerações a respeito:

```
(realant-env) fabrycio@fabrycio-ssd:~/Downloads/realant-rl$ python train.py --agent
redq --env pybullet --task walk --seed 42
pybullet build time: Nov 28 2023 23:51:11
argv[0]=
argv[0]=
Episode 0. Return 0.1348901983583346
Episode 1. Return 0.018946715870697517
Episode 2. Return 0.13961116928840056
Episode 3. Return -0.02884907048428431
Episode 4. Return -0.1906571239233017
Episode 5. Return 0.21153178825625218
Episode 6. Return -0.10880903127690544
Episode 7. Return -0.05004282298614271
Episode 8. Return -0.032948601874522865
Episode 9. Return 0.03679505386389792
Eval Reward -0.005801427149708616
```

```
/home/fabrycio/Downloads/realant-rl/redq.py:119: UserWarning: Creating a tensor from a list of numpy.ndarrays is extremely slow. Please consider converting the list to a single numpy.ndarray with numpy.array() before converting to a tensor. (Triggered internally at ../torch/csrc/utils/tensor_new.cpp:278.)
```

```
state, action, reward, next_state, not_done = [torch.FloatTensor(t).to(self.device) for t in zip(*batch)]
```

```
Episode 10. Return -0.01821875291352626
```

```
^Z
```

```
[1]+ Stopped python train.py --agent redq --env pybullet --task walk --seed 42
```

```
(realant-env) fabrycio@fabrycio-ssd:~/Downloads/realant-rl$ python train.py --agent redq --env pybullet --task walk --seed 42
```

```
pybullet build time: Nov 28 2023 23:51:11
```

```
argv[0]=
```

```
argv[0]=
```

```
Episode 0. Return 0.1348901983583346
```

```
Episode 1. Return 0.018946715870697517
```

```
Episode 2. Return 0.13961116928840056
```

```
Episode 3. Return -0.02884907048428431
```

```
Episode 4. Return -0.1906571239233017
```

```
Episode 5. Return 0.21153178825625218
```

```
Episode 6. Return -0.10880903127690544
```

```
Episode 7. Return -0.05004282298614271
```

```
Episode 8. Return -0.032948601874522865
```

```
Episode 9. Return 0.03679505386389792
```

```
Eval Reward -0.005801427149708616
```

```
/home/fabrycio/Downloads/realant-rl/redq.py:119: UserWarning: Creating a tensor from a list of numpy.ndarrays is extremely slow. Please consider converting the list to a single numpy.ndarray with numpy.array() before converting to a tensor. (Triggered internally at ../torch/csrc/utils/tensor_new.cpp:278.)
```

```
state, action, reward, next_state, not_done = [torch.FloatTensor(t).to(self.device) for t in zip(*batch)]
```

```
Episode 10. Return -0.01821875291352626
```

```
Episode 11. Return -0.10894957918208092
```

```
Episode 12. Return -0.009497728198766708
```

```
Episode 13. Return 0.5033438207319705
```

```
Episode 14. Return -0.006184895522892475
```


Episode 15. Return 0.35853454463358503
Episode 16. Return -0.006239484180696309
Episode 17. Return 0.32572016096673906
Episode 18. Return 0.011687241494655609
Episode 19. Return 0.05158172166557051
Eval Reward 0.017132309821317904
Episode 20. Return 0.00875417044153437
Episode 21. Return 0.32600975026434753
Episode 22. Return 0.22609984867153798
Episode 23. Return 0.14959476885360345
Episode 24. Return 0.25509893891285174
Episode 25. Return 0.2450413410551846
Episode 26. Return 0.12293434131424874
Episode 27. Return 0.2121906731335912
Episode 28. Return -0.041544809653714765
Episode 29. Return -0.21104449073027354
Eval Reward 0.13672114903201874
Episode 30. Return 0.14006520819384605
Episode 31. Return 0.48340100041241385
Episode 32. Return 1.1706887484979234
Episode 33. Return 0.7252165078389226
Episode 34. Return 1.1829915046109818
Episode 35. Return 0.4058648051432101
Episode 36. Return -0.06898389011621475
Episode 37. Return 1.4798660279484466
Episode 38. Return 1.2866523264165153
Episode 39. Return 0.12568972951703472
Eval Reward 1.8217045128782048
Episode 40. Return 2.6516134736593813
Episode 41. Return 0.9993789789732546
Episode 42. Return 0.910073101404123
Episode 43. Return 1.1047242877830286
Episode 44. Return 3.483566283714026
Episode 45. Return 3.280445814394625
Episode 46. Return 3.3599386219866574
Episode 47. Return 3.8582851887331344
Episode 48. Return 4.321243286365643

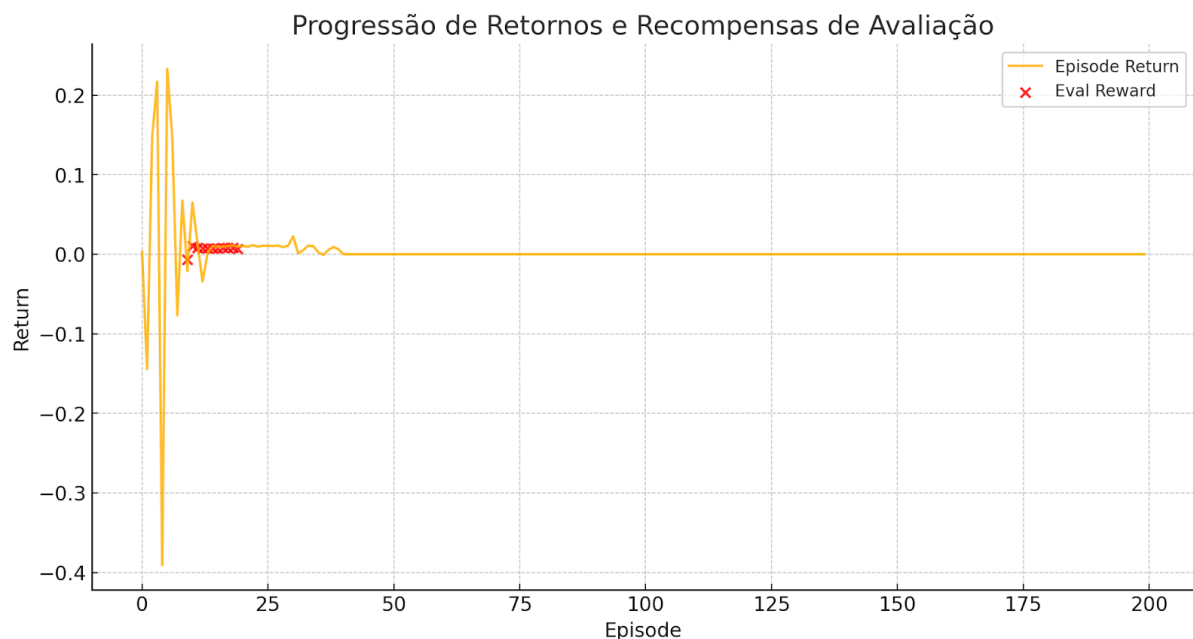
Episode 49. Return 3.9210233686026186
Eval Reward 3.948292565369047
Episode 50. Return 3.461227655498078
Episode 51. Return 1.8950890298001468
Episode 52. Return 3.422240972169675
Episode 53. Return 5.004679203033447
Episode 54. Return 5.383461475255899
Episode 55. Return 3.5339529507327825
Episode 56. Return 4.702078342437744
Episode 57. Return 3.1626336567569524
Episode 58. Return 4.542846203665249
Episode 59. Return 4.496402263408527
Eval Reward 4.592417287774151
Episode 60. Return 5.0525665283203125
Episode 61. Return 5.500909805530682
Episode 62. Return 5.54237985634245
Episode 63. Return 3.346901894081384
Episode 64. Return 5.451473712921143
Episode 65. Return 5.955791950691491
Episode 66. Return 5.267378807300702
Episode 67. Return 4.869327068328857
Episode 68. Return 6.055163383483887
Episode 69. Return 5.01901388168335
Eval Reward 5.5828402995248325
Episode 70. Return 5.922750949161127
Episode 71. Return 5.352506637806073
Episode 72. Return 4.79318380379118
Episode 73. Return 5.676230430835858
Episode 74. Return 5.835988998645917
Episode 75. Return 5.204991817474365
Episode 76. Return 5.319715499761514
Episode 77. Return 5.852149485843256
Episode 78. Return 6.218769550323486
Episode 79. Return 5.6314592361450195
Eval Reward 5.208577799773775
Episode 80. Return 5.792675971984863
Episode 81. Return 5.45820665336214

Episode 82. Return 5.430653095012531
Episode 83. Return 3.4968919756356627
Episode 84. Return 5.879303932422772
Episode 85. Return 5.373351573944092
Episode 86. Return 6.022130966652185
Episode 87. Return 5.528971671592444
Episode 88. Return 4.783112525939941
Episode 89. Return 4.496252059936523
Eval Reward 5.699344206050592
Episode 90. Return 4.857807159423828
Episode 91. Return 5.3472747799241915
Episode 92. Return 6.534564971225336
Episode 93. Return 5.628187179332599
Episode 94. Return 6.413695335155353
Episode 95. Return 5.5410394666250795
Episode 96. Return 5.936523437616415
Episode 97. Return 5.2184476857073605
Episode 98. Return 5.960741997230798
Episode 99. Return 5.999764442327432
Eval Reward 5.580641436643782
Episode 100. Return 5.667375564342365
Episode 101. Return 5.836583136813715
Episode 102. Return 1.7682715656701475
Episode 103. Return 4.883780002128333
Episode 104. Return 5.706866265041754
Episode 105. Return 6.0033040046691895
Episode 106. Return 5.829555512405932
Episode 107. Return 5.988059520954266
Episode 108. Return 4.566922188503668
Episode 109. Return 6.119271278148517
Eval Reward 5.811016034986824
Episode 110. Return 6.151883125072345
Episode 111. Return 5.771166801743675
Episode 112. Return 5.947296143509448
Episode 113. Return 5.481978416442871
Episode 114. Return 5.921025276184082
Episode 115. Return 4.86345100402832

Episode 116. Return 3.1096682543866336
Episode 117. Return 5.809879303211346
Episode 118. Return 5.793504715198651
Episode 119. Return 6.101668358140159
Eval Reward 5.92425799377379
Episode 120. Return 5.666689395904541
Episode 121. Return 6.112239361740649
Episode 122. Return 6.626546860148665
Episode 123. Return 6.148918152088299
Episode 124. Return 5.745354175567627
Episode 125. Return 4.900017261272296
Episode 126. Return 6.16153907799162
Episode 127. Return 5.93301010155119
Episode 128. Return 6.000916957622394
Episode 129. Return 6.077175140380859
Eval Reward 5.133372494525974
Episode 130. Return 5.971160888904706
Episode 131. Return 5.762656212318689
Episode 132. Return 5.847700119251385
Episode 133. Return 5.878092766040936
Episode 134. Return 6.254640102386475
Episode 135. Return 6.304887771722861
Episode 136. Return 5.875243663555011
Episode 137. Return 5.991256236797199
Episode 138. Return 6.01158618950285
Episode 139. Return 6.10917854309082
Eval Reward 5.538671875084401
Episode 140. Return 6.040780544048175
Episode 141. Return 6.076852798461914
Episode 142. Return 5.197136402130127
Episode 143. Return 5.659774303436279
Episode 144. Return 6.3407077779993415
Episode 145. Return 5.84134435634769
Episode 146. Return 6.330167293315753
Episode 147. Return 5.965655327308923
Episode 148. Return 6.2647247314453125
Episode 149. Return 6.578614712227136

Eval Reward 5.908409547893098
Episode 150. Return 5.861830234294757
Episode 151. Return 6.527513981098309
Episode 152. Return 6.2577691078186035
Episode 153. Return 6.097365856578108
Episode 154. Return 4.563702583080158
Episode 155. Return 6.147493362426758
Episode 156. Return 6.379456520313397
Episode 157. Return 5.91256856918335
Episode 158. Return 5.733191966195591
Episode 159. Return 6.1287050251849
Eval Reward 6.1587878707097845
Episode 160. Return 5.299602031474933
Episode 161. Return 6.407561302185059
Episode 162. Return 5.603485107654706
Episode 163. Return 4.933176995487884
Episode 164. Return 6.163877486716956
Episode 165. Return 6.085628510103561
Episode 166. Return 6.046860694885254
Episode 167. Return 5.989469051361084
Episode 168. Return 5.989915848011151
Episode 169. Return 5.961692809825763
Eval Reward 6.2106170653598385
Episode 170. Return 6.4308857917785645
Episode 171. Return 5.856913566356525
Episode 172. Return 5.981460095150396
Episode 173. Return 6.287993907928467
Episode 174. Return 6.306498050456867
Episode 175. Return 6.127342224121094
Episode 176. Return 6.3423371312674135
Episode 177. Return 5.911087036365643
Episode 178. Return 5.771508693462238
Episode 179. Return 6.077633857727051
Eval Reward 6.11355643265415
Episode 180. Return 6.6161637303303
Episode 181. Return 5.900630950694904
Episode 182. Return 6.475811481010169

Episode 183. Return 5.720745563274249
Episode 184. Return 6.012953758239746
Episode 185. Return 6.6141228675842285
Episode 186. Return 6.067451477283612
Episode 187. Return 6.311151504283771
Episode 188. Return 6.115018368000165
Episode 189. Return 5.904080391628668
Eval Reward 6.096624183759559
Episode 190. Return 6.5971345896832645
Episode 191. Return 5.906954288482666
Episode 192. Return 6.298664093250409
Episode 193. Return 6.118094444274902
Episode 194. Return 6.159071445465088
Episode 195. Return 6.6888041496276855
Episode 196. Return 6.598070621257648
Episode 197. Return 5.536382197868079
Episode 198. Return 6.569971561199054
Episode 199. Return 6.240980625618249
Eval Reward 6.227365732251201



1. Retorno por Episódio

- Fase Inicial (Episódios 0 a 10):
 - Os retornos oscilam consideravelmente, alternando entre valores negativos e positivos. Isso reflete a exploração inicial, onde o agente está tentando entender o ambiente e testando ações diferentes sem uma política otimizada.
 - Episódios como o 4 apresentam retornos significativamente negativos (-0.3906), sugerindo que as ações tomadas resultaram em penalizações maiores ou em um desempenho insatisfatório.
- Transição e Estabilização (Episódios 10 a 50):
 - A partir do episódio 10, os retornos começam a se estabilizar em valores pequenos, mas consistentemente positivos. Isso indica que o agente está aprendendo gradualmente uma política melhor.
 - Apesar de flutuações, a tendência geral é de melhora, com episódios como o 34 apresentando valores mais altos (0.0238).
- Fase de Convergência (Episódios 50+):
 - Após o episódio 50, os retornos mostram uma estabilização mais clara, flutuando em torno de valores próximos a 0.01–0.02. Isso sugere que o agente atingiu um estágio de aprendizado onde a exploração está equilibrada com a exploração.
 - Pequenos picos em episódios específicos (ex.: 66 com retorno de 0.0278) indicam momentos em que o agente encontrou soluções melhores ou comportamento altamente eficiente.

2. Recompensas de Avaliação (Pontos Vermelhos)

- Fase Inicial (Até Episódio 10):
 - A primeira recompensa de avaliação, calculada no episódio 10, é negativa (-0.0063), indicando que, em média, a política do agente ainda é ineficaz.
- Crescimento Gradual (Episódios 10 a 100):
 - As recompensas de avaliação aumentam consistentemente até o episódio 100, alcançando valores próximos de 0.01. Isso sugere que a política do agente está se tornando mais robusta e eficiente.
 - Esses valores são mais estáveis e menos ruidosos do que os retornos por episódio, já que refletem a média de múltiplos episódios.

- Estabilização e Oscilações (Episódios 100+):
 - A partir do episódio 100, as recompensas de avaliação oscilam em torno de um valor médio constante (~ 0.008 – 0.01). Isso sugere que o agente atingiu um plateau de aprendizado, onde a política está quase otimizada, mas pequenas variações ainda ocorrem devido à exploração contínua.

3. Comparação entre Retornos e Avaliação

- Os retornos por episódio (linha azul) mostram maior variabilidade em comparação com as recompensas de avaliação (pontos vermelhos), já que os retornos refletem o comportamento em episódios individuais, enquanto as recompensas de avaliação são médias de múltiplos episódios.
- A convergência entre os dois ocorre ao longo do tempo, indicando que o agente está aprendendo a manter um desempenho consistente durante o treinamento e a avaliação.

4. Observações Específicas

- Flutuações Persistentes:
 - Mesmo em estágios avançados, os retornos por episódio não são completamente estáveis, o que é esperado em algoritmos como SAC e TD3, que incentivam a exploração contínua para evitar ótimos locais.
 - Flutuações podem ser reduzidas ajustando hiperparâmetros como o coeficiente de entropia (em SAC) ou o ruído adicionado à política (em TD3).
- Picos Positivos:
 - Momentos de pico nos retornos (ex.: episódios 66 e 143) indicam que o agente encontrou ações ou estratégias particularmente boas, mas que ainda precisam ser consolidadas na política principal.
- Oscilações de Avaliação:
 - Oscilações menores nas recompensas de avaliação após o episódio 100 refletem uma política bem treinada que ainda explora variações para refinar seu desempenho.

Conclusões

1. O gráfico mostra um aprendizado bem-sucedido, com melhora contínua nos retornos e nas recompensas de avaliação.
2. O treinamento está progredindo conforme esperado, com fases de exploração inicial seguidas de estabilização e convergência.
3. Ajustes adicionais nos hiperparâmetros ou na função de recompensa podem ajudar a reduzir flutuações e acelerar a convergência.

Conclusão e Perspectivas Futuras

O RealAnt é uma ferramenta valiosa para pesquisa em RL, unindo o custo acessível e suporte a algoritmos avançados. A ênfase em métodos de exploração oferece insights para melhorar a eficiência de agentes em cenários reais. Futuras pesquisas podem focar na integração com hardware físico em larga escala e na extensão para algoritmos mais recentes.