UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

BRYAN LINCOLN MARQUES DE OLIVEIRA

Motivação intrínseca para aprendizado de manipulação robótica com recompensas esparsas

Goiânia
2019

BRYAN LINCOLN MARQUES DE OLIVEIRA

Motivação intrínseca para aprendizado de manipulação robótica com recompensas esparsas

Trabalho de Conclusão apresentado à Coordenação do Curso de Ciências da Computação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Bacharel em Ciências da Computação.

Área de concentração: Ciências da Computação, Inteligência Artificial.

Orientadora: Profa. Dra. Telma Woerle de Lima Soares

Goiânia
2019

BRYAN LINCOLN MARQUES DE OLIVEIRA

Motivação intrínseca para aprendizado de manipulação robótica com recompensas esparsas

Trabalho de Conclusão apresentado à Coordenação do Curso de Ciências da Computação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Bacharel em Ciências da Computação, aprovada em 12 de dezembro de 2019, pela Banca Examinadora constituída pelos professores:

Profa. Dra. Telma Woerle de Lima Soares
Instituto de Informática – UFG
Presidente da Banca

Prof. Dr. Anderson da Silva Soares
Instituto de Informática – UFG

Bryan Lincoln Marques de Oliveira

Graduando em Ciências da Computação pelo Instituto de Informática na Universidade Federal de Goiás. Durante a graduação foi monitor das disciplinas de Programação de Computadores e Matemática Discreta. Atualmente estuda e desenvolve na área de Aprendizado por Reforço pelo laboratório Deep Learning Brasil.

A todos aqueles que contribuíram direta ou indiretamente para meu crescimento pessoal e profissional.

Agradecimentos

Agradeço à minha família por todo o apoio durante minha jornada nesta graduação. Agradeço também à minha incrível namorada, Luana, por toda a motivação e companheirismo que permitiram visar objetivos ainda maiores. Aos meus amigos, pelos momentos de descontração e discussões construtivas. Ao Núcleo de Robótica Pequi Mecânico, pelas amizades, conhecimento técnico e ajuda no desenvolvimento de habilidades interpessoais. Agradeço também aos meus professores, pela excelência e qualidade técnica que me fizeram ter amor pela área. Em especial, agradeço à Profa. Dra. Telma Woerle de Lima Soares pela orientação neste trabalho e ao Prof. Dr. Anderson da Silva Soares pelo suporte fornecido. Agradeço também ao laboratório Deep Learning Brasil e aos membros do grupo de Aprendizado por Reforço, em especial ao MSc. Luckeciano Carvalho Melo, pelos valiosos *insights* e experiência compartilhada. Por fim, agradeço à COPEL e ao laboratório Deep Learning Brasil pelo auxílio financeiro que possibilitou a dedicação e a operacionalização do estudo.

If a machine is expected to be infallible, it cannot also be intelligent.

Alan Mathison Turing,
B. J. Copeland (2004). The Essential Turing, p.394, Oxford U. Press.

Resumo

Oliveira, Bryan Lincoln Marques de. **Motivação intrínseca para aprendizado de manipulação robótica com recompensas esparsas**. Goiânia, 2019. 44p. Relatório de Graduação. Instituto de Informática, Universidade Federal de Goiás.

Algoritmos de Aprendizado de Máquina têm se tornado cada vez mais eficientes em resolver problemas complexos do mundo real. Em especial, algoritmos de Aprendizado por Reforço são capazes de aprender comportamentos aplicáveis à robótica que podem substituir ou trabalhar em conjunto de modelos de controle clássico, aumentando assim sua robustez, aplicabilidade e viabilidade. No entanto, resta a dificuldade de se elaborar funções de recompensa que representem, para um agente de aprendizado por reforço, a tarefa que este deve executar. Pesquisas recentes nessa área propõem técnicas como curiosidade e motivação intrínseca como alternativa ao uso de recompensas extrínsecas do ambiente, se mostrando eficientes em guiar o agente a exploração satisfatória em ambientes de jogos como *VizDoom* e *Super Mario Bros*. Neste trabalho é analisado o impacto da técnica de motivação intrínseca no treinamento de agentes em ambientes de simulação robótica, assim como implicações gerais que a mesma tem sobre aspectos como generalização e eficiência de exploração e amostragem nos mesmos. Como resultado, são encontradas evidências de que a motivação intrínseca influencia positivamente na descoberta de comportamentos complexos por parte dos agentes, mantendo uma tendência para altos níveis de exploração mesmo após sua convergência.

Palavras–chave

Inteligência Artificial; Aprendizado por Reforço; Redes Neurais; Robótica.

Abstract

Oliveira, Bryan Lincoln Marques de. **Intrinsic motivation for robotic manipulation learning with sparse rewards**. Goiânia, 2019. 44p. Relatório de Graduação. Instituto de Informática, Universidade Federal de Goiás.

Machine Learning Algorithms have become increasingly efficient at solving complex real-world problems. In particular, Reinforcement Learning algorithms are capable of learning behaviors applicable to robotics that can replace or work together with classical control models, thereby increasing their robustness, applicability and viability. However, it remains difficult to design reward functions that represent, for a reinforcement learning agent, the task it must perform. Recent research in this area proposes techniques such as curiosity and intrinsic motivation as an alternative to the use of extrinsic environmental rewards, proving to be efficient in guiding the agent to satisfactory exploration in game environments such as *VizDoom* and *Super Mario Bros*. This paper analyzes the impact of the intrinsic motivation technique on agent training in robotic simulation environments, as well as its general implications for aspects such as generalization, exploration and sampling efficiency. As a result, evidence is found that intrinsic motivation positively influences the agents' discovery of complex behaviors, maintaining a tendency for high levels of exploitation even after it's convergence.

Keywords

Artificial Intelligence; Reinforcement Learning; Neural Networks; Robotics.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

CAPÍTULO 1

Introdução

Desde os últimos anos, algoritmos de Inteligência Artificial (IA) vêm sendo aplicados com grande sucesso em diversos setores da indústria e da sociedade. Suas soluções resolvem problemas que variam desde a detecção, classificação e acompanhamento de doenças [7, 16] a segmentação de ruas, carros e pedestres para navegação autônoma [28, 43]. Aplicações como essas não só permitem a solução de problemas antes inviáveis de se atacar como também permitem a substituição de processos caros e demorados.

Junto dos avanços da IA, avanços na robótica permitiram sua aplicação em tarefas de alta precisão que variam desde impressões 3D a execução de acrobacias [15]. Além disso, manipuladores robóticos têm se tornado quase omnipresentes em linhas de produção de indústrias em todo o mundo [14]. Em contrapartida, as técnicas de controle mais utilizadas nessas tarefas envolvem o desenvolvimento de modelos matemáticos complexos que devem ser precisamente construídos. Esses modelos levam em consideração, por exemplo, as características físicas do equipamento (e.g. distribuição de massa e torque máximo dos atuadores) e do ambiente (e.g. atrito com o chão, gravidade e rotação da Terra) [12]. A criação desses modelos pode se mostrar uma tarefa consideravelmente difícil e cara para ser executada em larga escala, principalmente quando o equipamento utilizado é formado por numerosas partes móveis e seu ambiente de trabalho é irregular.

Boa parte dos maiores desafios de implementação de robôs com comportamentos complexos são desafios de *software* [1]. Especialistas em controle e automação dedicam tempo considerável programando as ações dos atuadores robóticos para que estes executem suas tarefas de forma satisfatória. Outro problema é a falta de generalização de um comportamento entre diferentes equipamentos. Um comportamento programado para um hardware e estrutura específicos é dificilmente portado para outro sem diversos ajustes finos.

Pesquisas recentes em técnicas de IA aplicada a robótica, em especial de aprendizado por reforço, revelaram-se eficazes em substituir a necessidade de comportamentos estritamente programados para execução de tarefas [3, 23]. Essas técnicas são capazes de aprender comportamentos do zero em simulação de forma que seja possível portá-los para

o mundo real mantendo sua robustez a adversidades do ambiente [40]. Além disso, técnicas de aprendizado que trabalham em conjunto de técnicas clássicas a fim de permitir o porte de um comportamento de um equipamento para outro também foram desenvolvidas [18], permitindo a adaptação do agente para diferentes condições físicas.

Agentes de aprendizado por reforço trabalham com base nos sinais de retorno dados pelo ambiente em que são inseridos. Um problema que surge, com isso, é a necessidade de que os sinais, ou recompensas, guiem o aprendizado para o objetivo esperado. Mesmo sendo um processo menos trabalhoso que a construção de modelos de cinemática inversa, comportamentos complexos como locomoção e manipulação de objetos requerem a modelagem de sinais de recompensa que levam em consideração diversos fatores que podem ser difíceis de se descobrir e balancear [5, 26, 30]. Além disso, frequentemente agentes de aprendizado por reforço encontram problemas no sinal de recompensa modelado que os permitem maximizar seu valor acumulado sem executar a tarefa originalmente proposta [4, 13].

Naturalmente, o sinal de recompensa mais simples em tarefas do mundo real diz apenas se um objetivo foi alcançado ou não, ou seja, a recompensa é esparsa em relação as ações tomadas pelo algoritmo. Algoritmos convencionais de aprendizado por reforço tendem a ter dificuldades em aprender em configurações desse tipo [6]. Como alternativa, as noções da psicologia de motivação intrínseca [33] e curiosidade [34] foram aplicadas ao aprendizado por reforço com o objetivo de incentivar a exploração dos agentes em busca de recompensas extrínsecas [10, 29].

Este trabalho tem como objetivo a aplicação e a avaliação dos conceitos e técnicas de motivação intrínseca para exploração em ambientes de manipulação robótica com recompensa esparsa, dado o sucesso da aplicação destas técnicas em jogos de *Atari*, *Mario* e *VizDoom* [10, 29]. Para isso, este trabalho está organizado da seguinte maneira: no Capítulo 2 é apresentada a fundamentação teórica dos conceitos utilizados para a execução dos experimentos; no Capítulo 3 é descrito o ambiente utilizado, o algoritmo proposto e a definição dos testes; no Capítulo 4 são apresentados os resultados e no Capítulo 5 são apresentadas as conclusões deste trabalho.

CAPÍTULO 2

Fundamentação Teórica

Neste capítulo serão apresentados os conceitos teóricos que são base para este trabalho. Primeiramente será introduzido o funcionamento básico das redes neurais e das redes neurais profundas. Em seguida, será apresentado o conceito de aprendizado de máquina, onde serão definidas as técnicas de aprendizado supervisionado, não supervisionado e por reforço. Depois, será definido em mais detalhes a motivação e os tipos de aprendizado por reforço, assim como a Estimativa de Vantagem Generalizada e o algoritmo *Proximal Policy Optimization* que são utilizados como ferramenta neste trabalho. Por fim, serão apresentadas os principais problemas relacionados a ambientes com recompensa esparsa no contexto de aprendizado por reforço e será introduzida a técnica de exploração por curiosidade e motivação intrínseca.

2.1 Redes Neurais

Redes Neurais Artificiais (RNAs) são técnicas computacionais para modelagem matemática inspiradas pelo sistema nervoso central de organismos inteligentes. RNAs são tipicamente utilizadas como aproximadoras de funções universais e possuem estruturas que fazem um paralelo entre mecanismos biológicos como neurônios, sinapses e diferentes organizações estruturais (topologias) [32].

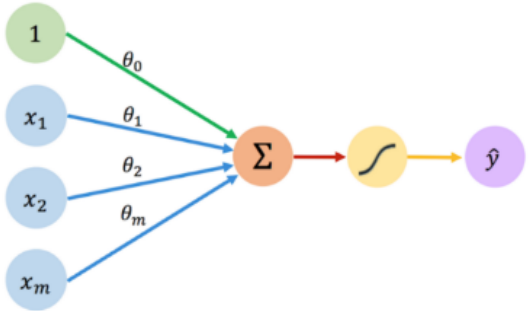


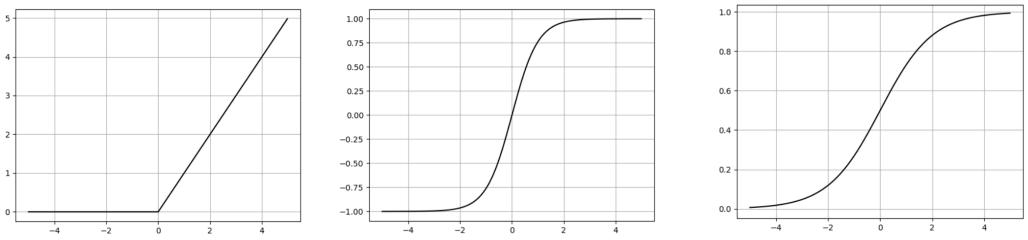
Figura 2.1: *Diagrama do funcionamento de um neurônio.*

A unidade mais básica de processamento de uma rede neural é o neurônio, que recebe um ou mais valores x_i como entrada, multiplica-os pelo peso sináptico correspondente θ_i e adiciona sua soma a um valor b conhecido como *bias* [17]. O valor final é alimentado a uma função de ativação que induz a não-linearidade da função aproximada pelo sistema. Este processo é demonstrado na Figura 2.1.

De uma forma mais didática, o parâmetro θ_i define quanto de um sinal x_i deverá ser considerado no processamento do neurônio e b permite que o neurônio ajuste o sinal de saída de uma forma independente dos sinais de entrada. A função de ativação permite o neurônio representar funções mais complexas que funções lineares [21]. A Equação 2-1 descreve o processamento de um neurônio.

$$\hat{y} = \sigma(w^T x + b) \tag{2-1}$$

Dentre as funções de ativação mais utilizadas na prática estão a *Rectified Linear Unit* (ReLU), a tangente hiperbólica (TanH) e a função sigmóide, utilizada na Equação 2-1. Cada função de ativação possui um comportamento específico, que pode ser observado na Figura 2.2.

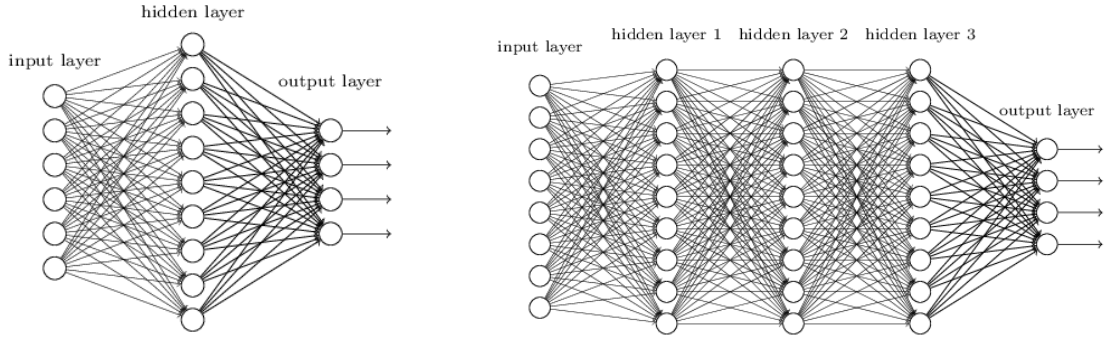


(a) *ReLU.* (b) *TanH.* (c) *Sigmóide.*

Figura 2.2: *(a), (b) e (c) representam o comportamento das funções de ativação ReLU, TanH e Sigmóide, respectivamente [21].*

Uma rede neural combina diversos neurônios em diversas camadas, formando um grafo direcionado (tipicamente acíclico) por onde os dados fluem. Os neurônios de uma camada processam os dados de entrada e passam os resultados como entrada dos neurônios da próxima camada, até encontrar a camada de saída [27]. Uma rede neural que possui muitas camadas é denominada uma Rede Neural Profunda. A Figura 2.3 ilustra o fluxo de processamento de redes neurais.

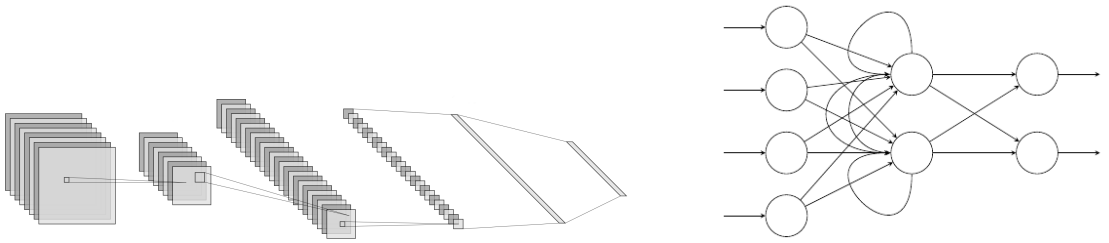
Existem diversas topologias para redes neurais que não seguem necessariamente a configuração apresentada. Redes Neurais Convolucionais, por exemplo, utilizam filtros convolucionais para extrair características em conjuntos de entradas onde a posição dos elementos é importante (e.g. imagens) [20] e Redes Neurais Recorrentes definem grafos cíclicos para modelagem de dados em domínios temporais [31]. As camadas de uma rede neural podem ter todos os seus neurônios conectados a todos os neurônios da camada



(a) *Rede neural.* (b) *Rede neural profunda.*

Figura 2.3: *(a) e (b) representam uma rede neural convencional e uma rede neural profunda, respectivamente [27].*

anterior - as camadas densas - ou podem ter conexões compartilhadas, como é o caso dos filtros convolucionais [20].



(a) *Topologia de uma Neural Convolucional.* (b) *Topologia de uma Rede Neural Recorrente.*

Figura 2.4: *(a) e (b) representam diferentes topologias que redes neurais podem assumir.*

A otimização, ou treinamento, de redes neurais é uma tarefa natural do aprendizado supervisionado, que será introduzido em seguida. Os algoritmos mais utilizados para isso são baseados na propagação do gradiente de uma função de custo através da rede a fim de atualizar seus parâmetros e minimizar esta função, um processo conhecido como *backpropagation* [17].

2.2 Aprendizado de Máquina

Aprendizado de máquina é um subcampo da Inteligência Artificial que estuda a capacidade de algoritmos aprenderem a executar tarefas baseadas em dados sem que sejam explicitamente programados para tal [37]. Na prática, seu objetivo é modelar teórica e matematicamente o processo de melhoria interativa das aplicações, baseando-se em padrões intrínsecos dos dados disponíveis. Existem três principais técnicas para alcançar este objetivo: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço.

No aprendizado supervisionado pares de exemplos no formato (X,Y), onde X são entradas e Y são saídas, são dados ao algoritmo para que este consiga aproximar uma função f que mapeie uma entrada $x \in X$ à saída correspondente $y \in Y$ [32]. Algoritmos de aprendizado supervisionado podem ainda ser classificados entre algoritmos de classificação e algoritmos de regressão, onde o primeiro mapeia uma entrada x a uma das classes em Y e o segundo mapeia uma entrada x em um valor contínuo no domínio de Y.

No aprendizado não supervisionado, por sua vez, o aprendizado é feito sobre dados não anotados, ou seja, sem o conjunto Y. O objetivo dessa classe de algoritmos é extrair padrões e regras, agrupar ou resumir os dados [32].

Já no aprendizado por reforço o aprendizado ocorre através da interação com um ambiente. Uma solução para um problema de aprendizado por reforço - o agente - deve interagir com o ambiente executando ações que geram sinais de recompensa. Dessa forma, o agente, em busca de maximizar o valor total da soma de suas recompensas futuras, deve aprender comportamentos (políticas) que ao mesmo tempo resolvem o problema inicialmente proposto [32]. A Figura 2.5 representa o processo descrito. Esta abordagem é descrita em mais detalhes na Seção 2.3.

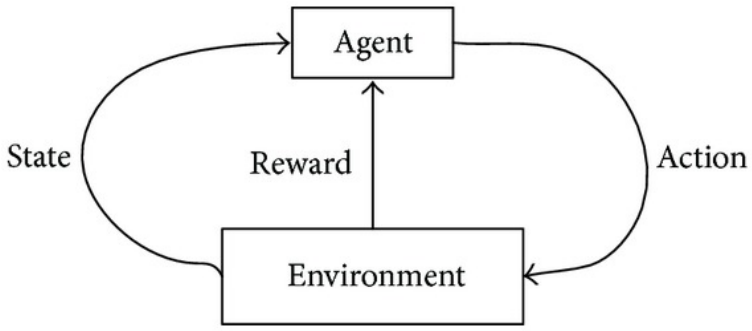


Figura 2.5: *Diagrama do processo de aprendizado por reforço [45]. Um agente (Agent), em um dado estado (State), seleciona ações (Action) que são aplicadas no ambiente (Environment) que, por sua vez, retorna um novo estado e um sinal de recompensa (Reward) para o agente.*

2.3 Aprendizado por Reforço

No aprendizado por reforço, um agente aprende através de sua interação com o ambiente em sequências de iterações, chamadas de episódios. Essa interação é tipicamente modelada como um Processo de Decisão de Markov (MDP), definido por um conjunto de estados S, um espaço de ações A, uma função de recompensa $r : S \times A \to [-\infty, \infty)$, um fator de desconto $\gamma \in [0,1)$ e uma função de transição $P : S \times A \to \Delta(S)$. Nessa configuração, $\Delta(S)$ é o espaço de distribuições de probabilidade $P(s'|s,a)$ de transição para um

estado s' a partir de um estado s e executando a ação a. O fator de desconto γ é utilizado para balancear quanto um agente deve priorizar recompensas imediatas ou recompensas futuras. A Figura 2.6 ilustra o sistema descrito.

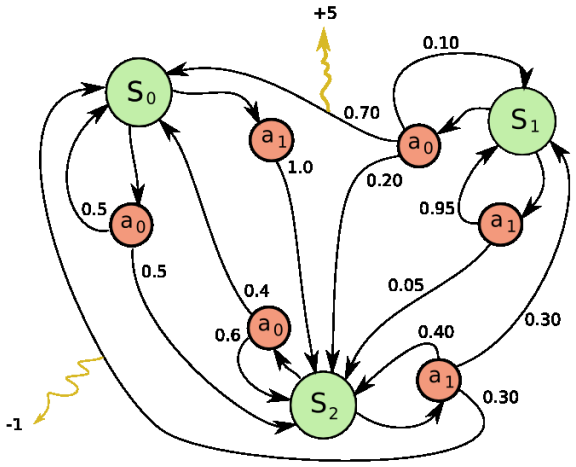


Figura 2.6: *Um Processo de Decisão de Markov [8]. Os estados $s_i \in S$ e as ações $a_i \in A$ são representadas pelos vértices verdes e vermelhos, respectivamente. Os valores das arestas são as probabilidades de transição P e os valores indicados pelas setas amarelas são as recompensas obtidas em cada transição.*

Sequências de transições em um MDP formam uma cadeia de Markov [19]. O valor total de recompensas, ou retorno, que um agente pode receber em uma cadeia a partir de um tempo t é definido pela Equação 2-2.

$$R_t = r_{t+1} + \gamma r_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2-2}$$

Na prática, diferentes cadeias de transições podem resultar no mesmo valor de R_t. Como o objetivo de um agente de aprendizado por reforço é maximizar o retorno para uma cadeia qualquer, definimos, para esse fim, a função de valor V para um estado s, como mostra a Equação 2-3. O valor de um estado, portanto, pode ser definido como o valor esperado da recompensa que pode ser acumulada a partir do estado atual, independentemente da sequência exata de transições que ocorreram.

$$V(s) = \mathbb{E}[R|S_t = s] \tag{2-3}$$

O agente seleciona suas ações baseando-se em sua política, que pode ser determinística, como na Equação 2-4, ou estocástica, como na Equação 2-5. Tipicamente, políticas determinísticas são utilizadas em ambientes determinísticos, como o jogo de xadrez, enquanto que políticas estocásticas são amplamente utilizadas em ambientes estocásticos, como em um jogo de poker, modelados como Processos de Decisão de Markov Parcial-

mente Observáveis (PDMPO) [19]. Em MDPs, a observação do estado atual é suficiente para que um agente obtenha uma política ótima, ou seja, é suficiente para que seja possível alcançar o objetivo de maximizar sua recompensa acumulada. PDMPOs, por outro lado, possuem variáveis escondidas que tornam essa tarefa significativamente difícil e, portanto, são abordados com estratégias mais sofisticadas.

$$\pi(s) = a \tag{2-4}$$

$$\pi(a|s) = P(A_t = a|S_t = s) \tag{2-5}$$

Um agente pode otimizar seu comportamento melhorando sua estimativa do valor de um par estado-ação (métodos baseados em valor) ou modificando sua política (métodos baseados em política). Além disso, um agente pode armazenar e otimizar um modelo do ambiente internamente (baseado em modelo) para consultá-lo na tomada de decisões ou trabalhar sem conhecer o funcionamento interno do ambiente (livre de modelo). Outro tipo de classificação especifica se o agente utiliza informações da própria política exploratória para otimização em direção à política ótima (*on-policy*) ou não (*off-policy*) [38].

Agentes modernos de aprendizado por reforço tipicamente utilizam redes neurais como forma de parametrizar seus componentes. Isso permite que o agente trabalhe com espaços de observação e ação contínuos e de alta dimensionalidade. Esses agentes são comumente denominados agentes de aprendizado por reforço profundo. Uma política parametrizada por parâmetros θ é denotada por π_θ.

Enxergando a otimização de agentes como um problema de maximização de recompensa acumulada, técnicas de exploração dão ao agente a possibilidade de sair de máximos locais ao explorar novos estados do ambiente. Uma das políticas exploratórias mais utilizadas é o ε-*greedy* [11]. Nessa técnica, o agente seleciona a ação de maior valor estimado com probabilidade $1-\varepsilon$ ou uma ação aleatória com probabilidade ε, como é feito em [25]. Ao decorrer do processo de treinamento, o parâmetro ε é reduzido até que somente a política do agente seja utilizada na seleção de ações.

Em métodos baseados em política, a exploração pode ser feita através da seleção da ação por amostragem em uma distribuição de probabilidades gerada pela política. Nesse método, a ação executada em um estado pode variar, mas de forma controlada. Outra forma de encorajar a exploração em métodos baseados em política é a adição de um termo de regularização baseado na entropia ($H(\pi(s_t;\theta))$) da distribuição gerada pelo agente na função de custo do problema de otimização da política [24, 42]. Isso faz com que o agente prefira distribuições mais uniformes a distribuições que encorajam demais as mesmas ações, evitando a convergência (ou seja, a estabilização em resultados

relativamente satisfatórios) para políticas sub-ótimas e mantendo um nível controlado de exploração [24, 42].

2.4 Estimativa de Vantagem Generalizada

Métodos baseados em política possuem a vantagem de aprender políticas estocásticas, ao contrário de métodos baseados em valor. Além disso, esses métodos conseguem trabalhar em espaços de ação de alta dimensionalidade, ou contínuos, por poderem ser representados por qualquer aproximador parametrizado que mapeie estados a ações [38]. Métodos baseados em política selecionam suas ações sem consultar diretamente uma função de valor, mas comumente as utilizam internamente no processo de otimização. Outra vantagem de utilizar métodos baseados em política é sua garantia de convergência pelo método de Gradiente de Política [39].

Métodos modernos de Gradiente de Política utilizam como objetivo, assim como os métodos tradicionais de aprendizado por reforço, a maximização da recompensa obtida pelo agente. Para isso, no entanto, é utilizada a Estimativa de Vantagem Generalizada (GAE) [35], definida pela Equação 2-6. Nesse contexto, definimos a vantagem A para um tempo t como a diferença entre a estimativa de valor para o estado atual e o real valor obtido empiricamente. Essa função nos diz se a recompensa obtida foi melhor ($A > 0$) ou pior ($A < 0$) que a recompensa esperada pelo agente. Se for melhor, devemos reforçar as ações executadas no episódio. Se for pior, devemos tornar essas ações menos prováveis.

$$A_t^{GAE(\gamma,\lambda)} = (1-\lambda)(A_t^{(1)} + \lambda A_t^{(2)} + \lambda^2 A_t^{(3)} + ...) \tag{2-6}$$

Aqui, $A_t^{GAE(\gamma,\lambda)}$ é um estimador para a função de vantagem generalizada, aplicado no tempo t e configurado pelos hiperparâmetros γ e λ. Os estimadores de vantagem com visão de n passos a partir do tempo t ($A^{(n)}$) são definidos pela Equação 2-7. O hiperparâmetro λ varia de 0 a 1 e controla quanto das estimativas de vantagem para passos futuros devem ser considerados, balanceando o viés e a variação das estimativas [35].

$$A_t^{(n)}(s,a) = r_t + \gamma V(s_{t+1}) + ... + \gamma^n V(s_{t+n}) - V(s_t) \tag{2-7}$$

O uso de estimadores de vantagem para a otimização nos métodos de Gradiente de Política torna o treinamento mais estável em relação ao uso dos retornos (Equação 2-2) para valores de λ entre 0 e 1 [35].

2.5 Proximal Policy Optimization

Um recorrente problema no treinamento de redes neurais é a escolha da taxa de aprendizado. Este parâmetro controla quanto o gradiente influencia na modificação dos parâmetros da rede e é bastante sensível: se for muito grande, a rede pode ser modificada drasticamente e perdemos qualquer conhecimento adquirido até o momento; se for muito pequeno, o treinamento é muito lento. Esse efeito é ainda maior no aprendizado por reforço, em especial em métodos baseados em política. Leves mudanças nos parâmetros podem levar a grandes mudanças no comportamento do agente, o que torna o treinamento um processo instável.

O algoritmo *Proximal Policy Optimization* (PPO) [36] foi introduzido com o propósito de mitigar justamente esse problema, limitando a mudança sofrida pela política entre as atualizações de seus parâmetros. A proporção da atualização da nova política em relação a política anterior é dada pela Equação 2-8.

$$d_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{2-8}$$

Nesse contexto, se $d_t(\theta) > 1$, então a ação a_t é mais provável na política atual do que na política antiga. Por outro lado, se $0 < d_t(\theta) < 1$, a ação a_t é menos provável na política atual do que na política antiga. Com base nisso, definimos a função de atualização limitada do PPO na Equação 2-9, onde *clip* é uma função de corte que restringe d_t ao intervalo $(1-\varepsilon, 1+\varepsilon)$.

$$L^{CLIP}(\theta) = \mathbb{E}_t[min(d_t(\theta)A_{\pi_\theta}, clip(d_t(\theta), 1-\varepsilon, 1+\varepsilon)A_{\pi_\theta})] \tag{2-9}$$

Essa função é calculada por N passos de otimização sobre um lote de iterações coletado durante um episódio. Como o objetivo do PPO é garantir pequenas atualizações nos parâmetros θ a cada episódio, a função de atualização busca o mínimo entre uma atualização para o passo atual com e sem o limite de proporção ε. Isso desencoraja a política de desviar muito do que era, uma vez que os gradientes zeram ao ultrapassar a região de corte, como mostra a Figura 2.7. Em outras palavras, se a ação foi boa ($A > 0$) e se tornou mais provável após o último passo de otimização do gradiente, é permitido aumentar sua probabilidade ainda mais até que d_t atinja o limite de corte $1+\varepsilon$, ou diminuí-la para reparar erros de passos anteriores. Da mesma forma, se a ação foi ruim ($A < 0$) e se tornou menos provável, é permitido diminuir sua probabilidade até que d_t atinja o limite de corte $1-\varepsilon$, ou aumentá-la para reparar erros de passos anteriores.

Neste trabalho, a função de atualização do PPO é combinada com o termo de regularização baseado na entropia, definido na Seção 2.4. Além disso, um modelo parametrizado que estima a função de valor V também é otimizado, caracterizando

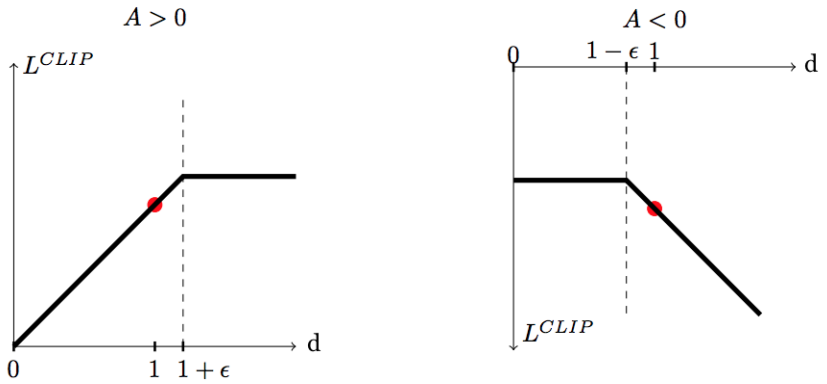


Figura 2.7: *Comportamento da função L^{CLIP} para valores de vantagem positivos (esquerda) e negativos (direita) [36], para atualizações em um único episódio.*

assim uma arquitetura Ator-Crítico [38]. A função de atualização toma então a forma da Equação 2-10, onde c é uma constante que regula a escala do termo de entropia e L^{VF} é a função de custo do modelo da função de valor.

$$L^{CLIP+VF+H}(\theta) = \mathbb{E}_t[L^{CLIP} - L^{VF} + cH(\pi(s_t;\theta))] \tag{2-10}$$

Por fim, o algoritmo PPO com função de atualização limitada é apresentado no Algoritmo 2.1 [2], onde N é o tamanho do lote de treinamento do algoritmo de otimização.

Algoritmo 2.1: PPO com função de atualização limitada

Entrada: parâmetros da política inicial θ_0, limiar de corte ε

para $k = 0, 1, 2, \ldots$ **faça**

Colete um conjunto de trajetórias D_k com política $\pi_k = \pi(\theta_k)$

Estime a função de vantagem $A_t^{GAE(\gamma,\lambda)}$

Calcule a atualização da política

$$\theta_{k+1} = arg\max_\theta L^{CLIP+VF+H}(\theta_k)$$

executando N passos do gradiente ascendente, onde

$$L^{CLIP+VF+H}(\theta_k) = \mathbb{E}_t[L^{CLIP} - L^{VF} + cH(\pi(s_t;\theta))]$$

2.6 Motivação Intrínseca

Um dos maiores desafios do aprendizado de reforço é aplicação de suas técnicas em ambientes reais. Dois principais problemas podem ser destacados: generalização e modelagem de recompensa. O primeiro diz respeito a capacidade de um agente treinado em uma configuração específica, ou um simulador, ser capaz de atuar de forma satisfatória em diferentes configurações, como em um manipulador robótico real. O segundo diz respeito à dificuldade de se modelar funções de recompensa que guiem um agente a agir de forma a completar uma determinada tarefa.

A sensibilidade dos algoritmos de aprendizado por reforço ao segundo problema faz com que, na maioria das vezes, o agente aprenda a explorar falhas do simulador ou falhas da modelagem da recompensa e simplesmente ignorar a tarefa que lhe foi dado. Além disso, a modelagem da função de recompensa para tarefas minimamente complexas é uma tarefa difícil [5, 26, 30]. A alternativa natural é o uso de recompensas esparsas, recompensando o agente apenas no momento em que ele completou a tarefa desejada, por exemplo. Porém, em ambientes onde a recompensa é esparsa ou inexistente, aprender boas políticas por tentativa e erro é uma tarefa extremamente difícil. Nessas situações, não existe um direcionamento claro de como o agente deve modificar sua política a fim de maximizar sua recompensa. Além disso, técnicas de exploração clássicas comumente utilizam distribuições de probabilidade sobre as ações que não levam em consideração guiar o agente a estados inexplorados, o que pode não ser suficiente para a descoberta de políticas que são capazes de alcançar o objetivo determinado [29].

Como saída para este problema é possível encorajar o agente a explorar melhor o ambiente e aprender novas habilidades recompensando-o de forma intrínseca, ou seja, independente da recompensa fornecida pelo ambiente (extrínseca). Uma forma de modelar a motivação intrínseca é através da geração um sinal de recompensa proporcional ao nível de surpresa do agente em relação aos estados que ele observa [22, 29]. Para isso, um modelo de futuro é responsável por prever o próximo estado s_{t+1} dados o estado atual s_t e a ação a executada, como mostra a Equação 2-11.

$$\hat{s}_{t+1} = f(s_t, a_t; \theta_F) \tag{2-11}$$

Dado o real estado futuro s_{t+1} é calculado o erro de predição, ou seja, a função de custo do modelo, como mostra a Equação 2-12.

$$L_F(s_{t+1}, \hat{s}_{t+1}) = \frac{1}{2}\|\hat{s}_{t+1} - s_{t+1}\|_2^2 \tag{2-12}$$

O erro de predição é então multiplicado por uma constante η que ajusta sua escala. Por fim, o valor é adicionado à recompensa do ambiente (r^e) como recompensa intrínseca, como mostra a Equação 2-13.

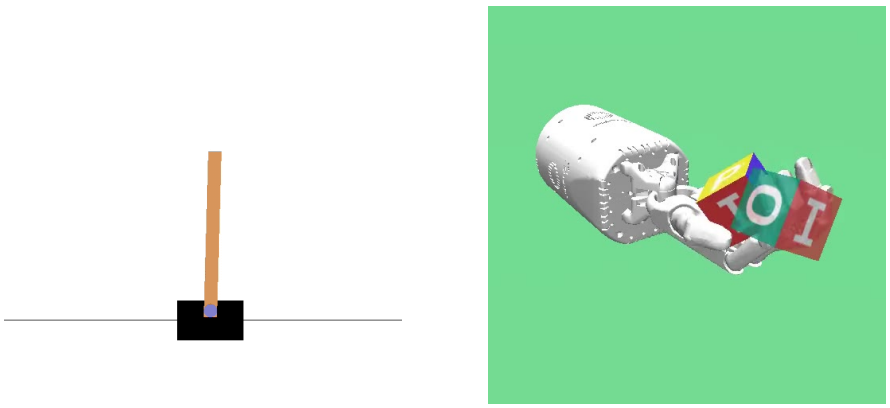
$$r_t = r_t^e(s_t, a, s_{t+1}) + \eta L_F \tag{2-13}$$

CAPÍTULO 3

Materiais e Métodos

Utilizando como base os conceitos previamente abordados, este Capítulo descreve os ambientes de *benchmark* utilizados para experimentação, o funcionamento do algoritmo utilizado e os testes executados. Primeiramente, são apresentados os objetivos do agente nos ambientes escolhidos e como é configurada a observação e a recompensa, assim como o espaço de ações e a duração do episódio. Em seguida, o algoritmo PPO é retomado e modificado para incluir o módulo de motivação intrínseca. O funcionamento geral dos passos do algoritmo são descritos em mais detalhes e a arquitetura das redes neurais utilizada é apresentada. Por fim, os testes são descritos e as hipóteses deste trabalho são estabelecidas, assim como o comportamento esperado de cada métrica a ser analisada.

3.1 OpenAI Gym

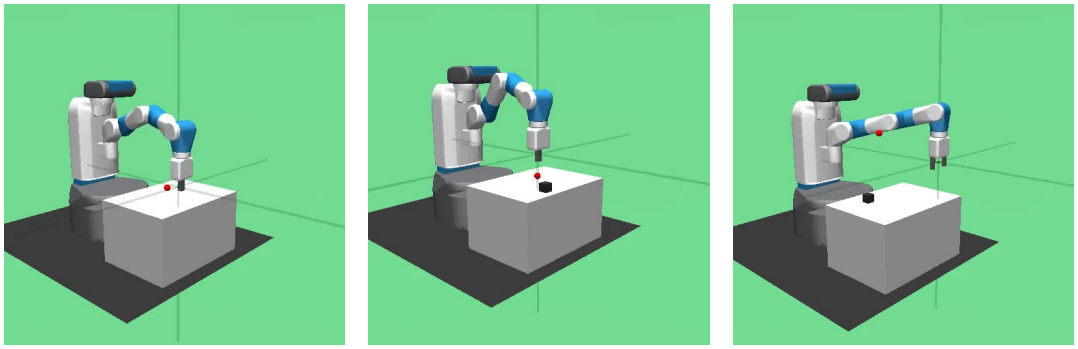


(a) *CartPole.* (b) *HandManipulateBlock.*

Figura 3.1: *Ambientes do Gym. Em (a), o agente deve equilibrar a estaca selecionando ações entre [esquerda, direita]. Em (b), o agente deve posicionar o bloco opaco como mostra o bloco translúcido de exemplo, selecionando valores de torque para cada junta da mão.*

O OpenAI Gym [9] é uma ferramenta de *benchmark* para algoritmos de aprendizado por reforço que tem como objetivo ser agnóstico em relação à estrutura dos agentes e manter uma interface amigável e comum a seus diversos ambientes. O Gym inclui ambientes que variam desde problemas simples, como o famoso *CartPole* (Figura 3.1(a)), até simulações de mãos robóticas, como o *HandManipulateBlock* (Figura 3.1(b)).

Para os testes deste trabalho, foram escolhidos ambientes de manipulação robótica *FetchReach*, *FetchPush*, *FetchPickAndPlace*, ilustrados pela Figura 3.2. A escolha desses ambientes possibilita a avaliação da aplicabilidade de técnicas similares a apresentada neste trabalho em manipuladores reais.



(a) *FetchReach.* (b) *FetchPush.* (c) *FetchPickAndPlace.*

Figura 3.2: *Ambientes de manipulação selecionados.*

Em *FetchReach* (Figura 3.2(a)) o agente deve posicionar a garra no alvo representado pela esfera vermelha. Em *FecthPickAndPlace* (Figura 3.2(c)) o agente deve pegar o disco com a garra e posicioná-lo no alvo. Em *FetchPush* (Figura 3.2(b)) o agente deve empurrar o disco para o alvo.

A recompensa nestes ambientes é considerada, neste trabalho, sendo -1 para todo passo de simulação onde o agente não resolveu a tarefa e 0 quando a tarefa é resolvida. O episódio é limitado a 2048 passos de simulação e é finalizado assim que o agente alcança o objetivo. A observação do ambiente é composta pela posição, rotação e velocidade das juntas do manipulador, assim como posição, rotação e velocidade do disco e a posição do alvo. A ação de um agente neste ambiente é um vetor composto por 4 componentes que representam o torque em cada atuador do manipulador, desde cada junta até a abertura da garra.

3.2 Algoritmo PPO com Motivação Intrínseca

O algoritmo utilizado adiciona um passo adicional no funcionamento padrão do PPO descrito em [2] e é representado pelo Algoritmo 3.1 e pela Figura 3.3. Os passos são descritos com maiores detalhes logo em seguida.

Algoritmo 3.1: PPO com motivação intrínseca

Entrada: parâmetros da política inicial θ_0, limiar de corte ε

para $k = 0, 1, 2, ...$ **faça**

Colete um conjunto de trajetórias D_k com política $\pi_k = \pi(\theta_k)$

para *cada tupla* (s_t, a_t, r_t, s_{t+1}) *em* D_k **faça**

Calcule o erro do modelo de futuro como mostra a Equação 2-12

Calcule a recompensa total r_t, como mostra a Equação 2-13

Estime a função de vantagem $A_t^{GAE(\gamma,\lambda)}$

Calcule a atualização da política

$$\theta_{k+1} = arg\max_{\theta} L^{CLIP+VF+H}(\theta_k)$$

executando N passos do gradiente ascendente, onde

$$L^{CLIP+VF+H}(\theta_k) = \mathbb{E}_t[L^{CLIP} - L^{VF} + cH(\pi(s_t;\theta))]$$

Durante a coleta de trajetórias, uma observação s_t do ambiente é passada para o agente a cada passo t de simulação. Essa observação é então alimentada em duas redes neurais: uma rede que calcula a ação a_t a ser tomada, chamada de ator, e uma rede que calcula o valor do estado atual, chamada de crítico. Após a seleção da ação, esta é executada e obtém-se a observação do estado s_{t+1} e a recompensa extrínseca do ambiente. Após o passo de coleta de trajetórias, a recompensa intrínseca é calculada alimentando o modelo de futuro com (s_t, a_t) e calculando o erro $L_F(s_t + 1, \hat{s}_{t+1})$. Ao final do episódio o valor da função de vantagem é calculado, os parâmetros do ator são atualizados de acordo com a Equação 2-10, os parâmetros do modelo de futuro são atualizado de acordo com a Equação 2-12 e os parâmetros do crítico são atualizados de acordo com o erro quadrático médio entre os valores dos estados previstos pela rede e os reais valores obtidos no episódio.

A rede neural do ator possui duas camadas densas com 128 neurônios cada e função de ativação ReLU, uma terceira camada densa com 15 neurônios e função de ativação TanH e uma camada densa final que tem por objetivo determinar a média μ da distribuição de probabilidades da saída. O desvio padrão σ da distribuição é determinada por parâmetros aprendidos pelo próprio modelo e tende a ser controlado a fim de manter a entropia da distribuição alta enquanto for possível [42]. A ação do agente é selecionada fazendo a amostragem da distribuição $N(\mu, \sigma^2)$. A rede neural do crítico possui uma topologia semelhante a do ator nas duas primeiras camadas mas sua camada de saída possui apenas um neurônio. A configuração detalhada dos hiperparâmetros do algoritmo utilizado nos experimentos pode ser conferida no Apêndice A.

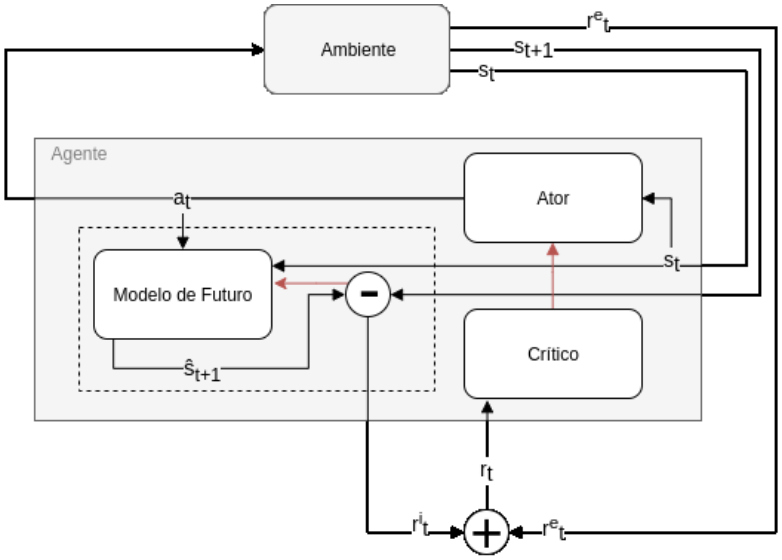


Figura 3.3: *Diagrama do funcionamento do algoritmo PPO com motivação intrínseca com modelo Ator-Crítico. As setas pretas representam os fluxos dos dados no passo de coleta de trajetórias e as setas vermelhas representam o fluxo dos dados no passo de otimização.*

3.3 Testes

Os testes são executados de forma a analisar o impacto da motivação intrínseca no aprendizado por reforço em ambientes de manipulação robótica com recompensa esparsa. Para isso, o treinamento do agente é executado nas mesmas condições em dois testes, onde no primeiro a recompensa intrínseca é adicionada à recompensa total e na segunda somente a recompensa extrínseca é transmitida. Para cada teste, o agente foi treinado no ambiente *FetchReach* por 10 milhões de iterações iterações e nos ambientes *FetchPush* e *FetchPickAndPlace* por 30 milhões de iterações. Os testes foram executados paralelamente em seis máquinas com dez núcleos de processamento cada e sem GPUs, resultando em um tempo de treinamento total de aproximadamente oito horas. Os resultados são obtidos durante o processo de treinamento através da ferramenta Tensorboard[1].

Como métrica de desempenho é analisada a porcentagem de sucessos em cada lote de avaliações durante o treinamento de cada tarefa, assim como a recompensa intrínseca média, a entropia da política e a divergência KL entre as políticas de antes e depois de uma atualização de parâmetros. A taxa de sucesso mostra a eficiência do algoritmo em alcançar o objetivo e se isso ocorre de forma consistente. A recompensa intrínseca mostra como o modelo de futuro se comporta ao se deparar com os diversos estados em que o agente se encontra durante o treino e testa sua capacidade de predição, ou seja, o entendimento da dinâmica do ambiente. Picos no gráfico de recompensa intrínseca

[1] Disponível em `https://www.tensorflow.org/tensorboard/get_started`

podem indicar aumento do interesse do agente sobre alguma situação específica que modificou o padrão das observações do ambiente de forma significativa. A entropia da política indica quão conservador o agente é em relação a probabilidade de escolha entre as ações, ou seja, se o mesmo permite que diferentes sequências de ações sejam tomadas para atingir seus objetivos. Altos valores de entropia da política podem indicar boas capacidades de generalização, uma vez que mostra que o agente não só simplesmente "decorou"uma sequência específica de ações que resolve o objetivo [42]. A divergência KL mostra o ritmo no qual a política se modifica entre as atualizações. Esse ritmo tende a ser alto em fases do treinamento em que o agente está modificando significativamente sua política e pode estar relacionado com quedas ou aumentos em sua taxa de sucesso.

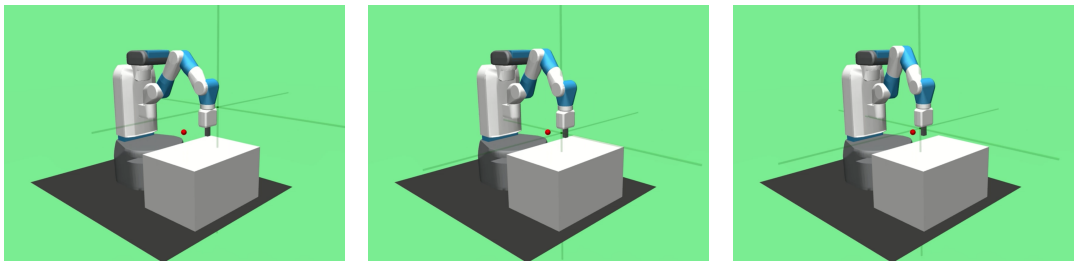
Como hipótese, é esperado que o agente com recompensa intrínseca se saia melhor que o algoritmo sem a mesma, refletindo na razão de sucessos máxima de ambos e na velocidade em que se estabilizam nesta métrica. Além disso, espera-se picos de recompensa intrínseca quando mudanças de comportamento do agente influenciam diferentes aspectos da observação no ambiente.

CAPÍTULO 4

Resultados

Neste Capítulo são descritos os resultados obtidos da execução dos testes descritos no capítulo anterior. Para cada ambiente de teste, o agente foi treinado uma vez utilizando o módulo de motivação intrínseca e uma vez sem utilizá-lo pelo mesmo número de iterações. Na análise, nos referimos ao agente do segundo treinamento como algoritmo, ou agente, *baseline*. Dizemos também que o algoritmo "convergiu"quando sua razão de sucessos se torna satisfatória. Em todos os gráficos, o eixo horizontal representa o número de iterações, as linhas em vermelho indicam o uso do algoritmo PPO com motivação intrínseca, ou curiosidade, e as linhas azuis indicam o uso algoritmo PPO *baseline*. Nos gráficos de recompensa intrínseca e de divergência KL as linhas foram suavizadas[1] para melhor compreensão. Os resultados foram agrupados de acordo com os ambientes treinados e uma análise geral é feita ao final deste Capítulo.

4.1 *FetchReach*



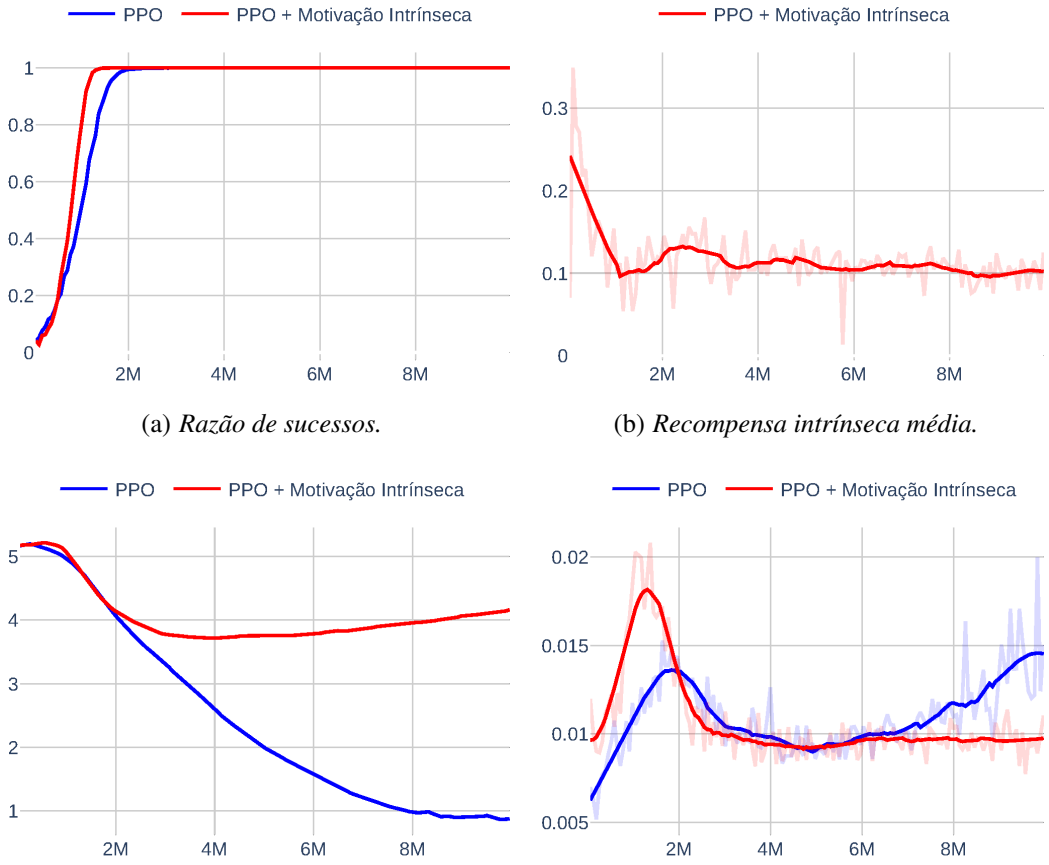
(a) *Início do episódio.* (b) *Episódio em progresso.* (c) *Próximo do fim do episódio.*

Figura 4.1: *Progresso do agente treinado com curiosidade no ambiente FetchReach.*

No ambiente *FetchReach*, ambos os algoritmos alcançaram 100% de sucesso em menos de dois milhões de passos de simulação. A Figura 4.1 mostra o agente treinado com módulo de curiosidade em alguns passos de simulação. É interessante notar o decaimento

[1] O filtro de Savitzky-Golay, implementado pela biblioteca SciPy [41], foi utilizado com uma janela de tamanho 31 e ordem polinomial 3.

do valor da recompensa intrínseca (Figura 4.2(b)) em um ritmo semelhante ao aumento da razão de sucessos do algoritmo PPO com módulo de curiosidade (Figura 4.2(a)). Este fenômeno pode indicar o aprendizado da dinâmica do ambiente por parte do agente, ou seja, a medida em que o agente descobre como suas ações influenciam o ambiente e, consequentemente, mais próximo está de seu objetivo, menos surpreso com os novos estados ele fica.



(a) *Razão de sucessos.* (b) *Recompensa intrínseca média.*

(c) *Entropia da política.* (d) *Divergência KL entre as políticas.*

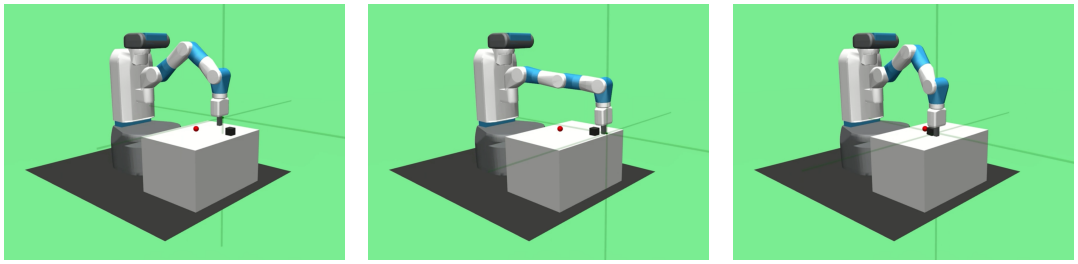
Figura 4.2: *Estatísticas para o ambiente FetchReach.*

Outra análise que pode ser feita neste ambiente é em relação aos gráficos de entropia (Figura 4.2(c)) e divergência KL (Figura 4.2(d)). A entropia da política do agente com módulo de curiosidade tende a se manter alta mesmo após a convergência do algoritmo, enquanto que o agente *baseline* diminui sua entropia se tornando cada vez mais determinístico. Na prática, vemos que o agente com curiosidade se mantém com altos níveis de exploração, possibilitando que melhores sequências de ações para alcançar o objetivo proposto sejam encontradas. Pode-se observar também pelo gráfico de divergência KL que após a convergência a política do agente com curiosidade tende sofrer poucas modificações em relação à política do agente *baseline*. Este comportamento reforça ainda mais a hipótese de que a dinâmica do ambiente foi aprendida pelo agente

com curiosidade, fazendo com que a recompensa intrínseca seja muito menor e as alterações na política sejam mínimas.

4.2 *FetchPush*

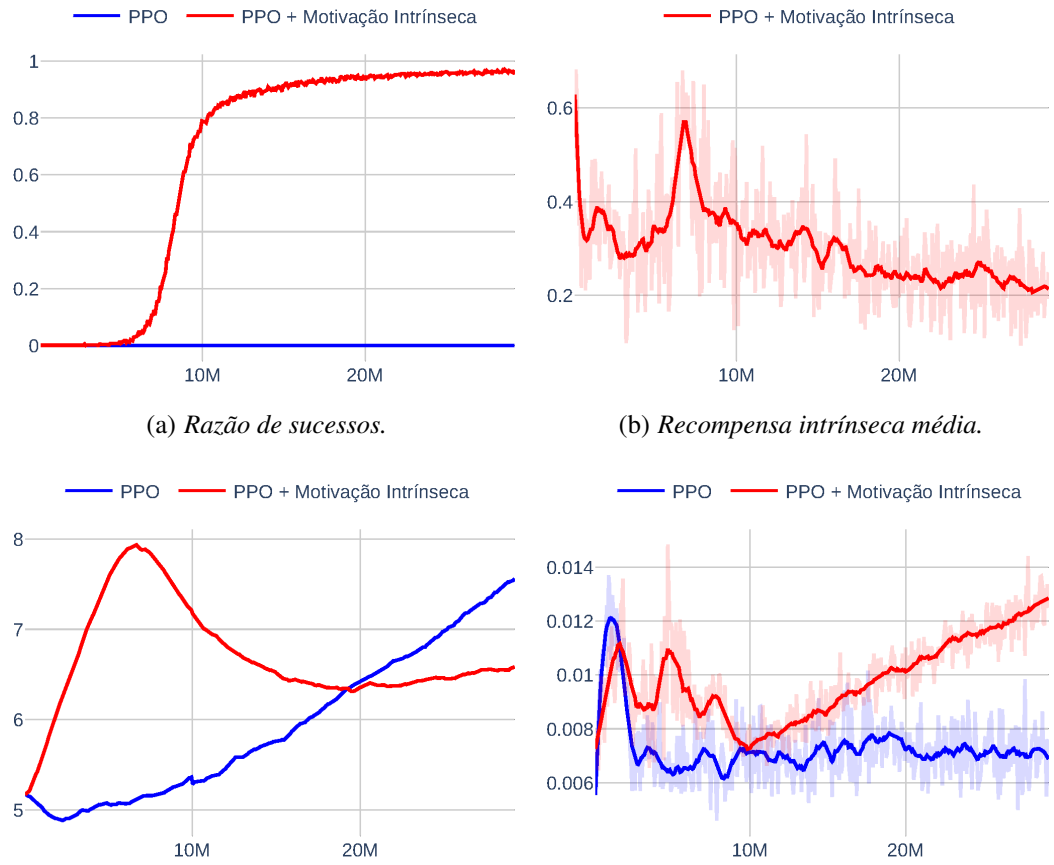
No ambiente *FetchPush* o algoritmo PPO original não conseguiu resolver a tarefa durante o treinamento, como mostra o gráfico da Figura 4.4(a). Uma explicação é a quantidade de ações em sequência necessárias para alcançar o objetivo, tornando baixa a chance disso ocorrer por acaso. O PPO com módulo de curiosidade, no entanto, pode aproveitar o incentivo que recebe ao explorar as posições do próprio manipulador e, o mais impactante, o comportamento do bloco quando o atinge por acaso. Como a posição, rotação e velocidades do bloco fazem parte da observação do agente e o modelo de futuro não conhece a dinâmica do bloco no início do treinamento, esbarrar no mesmo faz com que o erro de predição aumente de forma significativa. Como o erro de predição é diretamente proporcional à recompensa intrínseca do agente, este se torna consideravelmente interessado em jogar o bloco de um lado para o outro, eventualmente alcançando o objetivo. Quando este processo ocorre, observa-se um pico no gráfico de recompensa intrínseca (Figura 4.4(b)) e logo em seguida um aumento na taxa de sucesso (Figura 4.4(a)).



(a) *Início do episódio.* (b) *Episódio em progresso.* (c) *Fim do episódio.*

Figura 4.3: *Progresso do agente treinado com curiosidade no ambiente FetchPush.*

O gráfico de divergência KL (Figura 4.4(d)) mostra como a política do agente com curiosidade é modificada gradativamente durante o treino, mesmo após a convergência. Como a razão de sucessos ao final do treinamento foi de 96%, como mostra a Tabela 4.1, e a recompensa intrínseca possuía um valor considerável, acredita-se que o agente poderia melhorar sua política ainda mais. Além disso, o gráfico da Figura 4.4(c) mostra como a entropia da política do agente com curiosidade possui um pico logo no início do treinamento, possivelmente motivado pela recompensa intrínseca, e como a mesma se estabiliza em valores altos após a convergência. O agente *baseline*, por outro lado, não recebendo sinais de como modificar sua política opta por aumentar a entropia das distri-



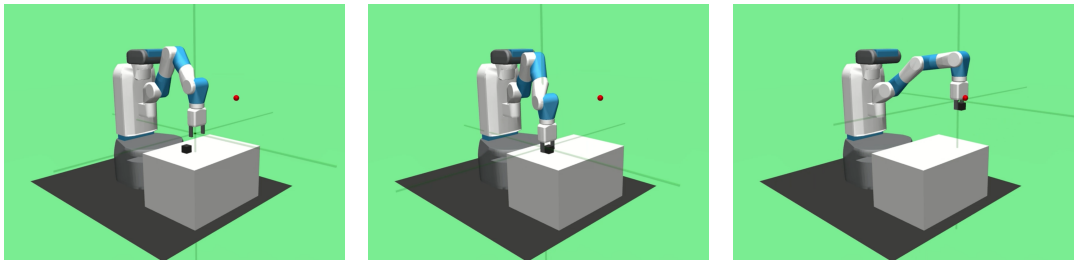
(a) *Razão de sucessos.* (b) *Recompensa intrínseca média.*

(c) *Entropia da política.* (d) *Divergência KL entre as políticas.*

Figura 4.4: *Estatísticas para o ambiente FetchPush.*

buições das ações, reforçando seu comportamento exploratório. Além disso, é possível ver que a divergência KL entre as políticas do agente *baseline* se mantém relativamente constante. Um exemplo de episódio em que o agente treinado com curiosidade consegue resolver a tarefa é mostrado na Figura 4.3.

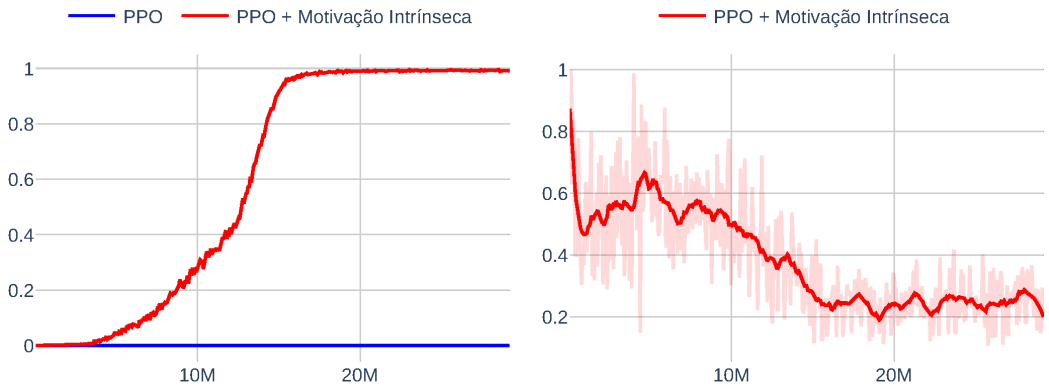
4.3 *FetchPickAndPlace*



(a) *Início do episódio.* (b) *Episódio em progresso.* (c) *Fim do episódio.*

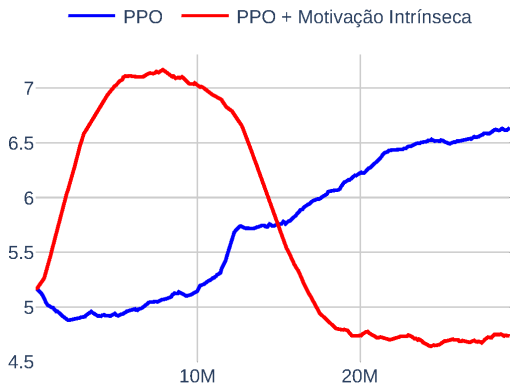
Figura 4.5: *Progresso do agente treinado com curiosidade no ambiente FetchPickAndPlace.*

Neste ambiente, assim como em *FetchPush*, o algoritmo PPO original não é capaz de realizar a tarefa durante o treinamento (Figura 4.6(a)) pelos mesmos motivos. Neste ambiente o desafio é ainda maior, pois uma grande quantidade de ações extremamente específicas devem ser executadas para alcançar o objetivo. Para o PPO com módulo de curiosidade, porém, a tarefa ainda é factível. O agente é recompensado já em poucas iterações após o início do treino ao interagir com o bloco, sendo encorajado a manipulá-lo uma vez que sua posição, rotação e velocidades fazem parte de sua observação, o que influencia em sua recompensa intrínseca (Figura 4.6(b)). Neste caso nota-se também que sua evolução é um processo mais lento, cerca de 15 milhões de iterações para que o agente alcance mais de 85% de sucesso.

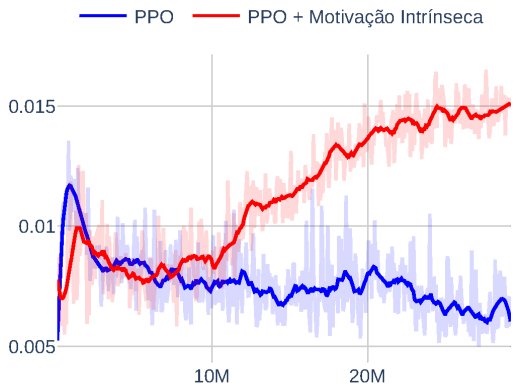


(a) *Razão de sucessos.*

(b) *Recompensa intrínseca média.*



(c) *Entropia da política.*



(d) *Divergência KL entre as políticas.*

Figura 4.6: *Estatísticas para o ambiente FetchPickAndPlace.*

Mais uma vez, há um pico no gráfico de recompensa intrínseca pouco antes da taxa de sucesso do agente começar a subir. A curiosidade do agente diminui poucas iterações após a convergência, uma vez que a dinâmica do ambiente não trás mais tantas novidades. Além disso, observa-se o mesmo comportamento nos gráficos de entropia (Figura 4.6(c)) e divergência KL entre políticas (Figura 4.6(d)) se comparado com o ambiente *FetchPush*. O interessante a se notar, porém, é quão conservadora em relação a

uniformidade das probabilidades de ações o agente se tornou neste ambiente, como mostra o gráfico de entropia. Esse comportamento é esperado pois o agente deve sempre abrir a garra para segurar o bloco e evitar soltá-lo, uma vez que o mesmo pode cair da mesa, fora de seu alcance. A Figura 4.5 mostra o agente treinado com curiosidade resolvendo a tarefa proposta.

4.4 Considerações Gerais

Em geral, o algoritmo com módulo de curiosidade foi capaz de resolver as tarefas de todos os ambientes, como mostra a Tabela 4.1. É interessante notar que a taxa de sucesso no ambiente *FetchPickAndPlace* é maior que a taxa de sucesso no ambiente *FetchPush*. Isso acontece pois em algumas avaliações no ambiente *FetchPush* o manipulador pode jogar o bloco para fora da mesa, impossibilitando o agente de completar sua tarefa. Em *FetchPickAndPlace* a garra do manipulador robótico raramente solta o bloco uma vez que conseguiu agarrá-lo. Após segurar o bloco, basta que o agente se comporte como em *FetchReach*, o que é uma tarefa simples.

Tabela 4.1: *Razão de sucesso dos algoritmos em cada ambiente de teste.*

Algoritmo	FetchReach	FetchPush	FetchPickAndPlace
PPO	100%	0%	0%
PPO + Motivação Intrínseca	100%	96%	99%

CAPÍTULO 5

Conclusões

Neste trabalho, foi apresentada uma alternativa para o treinamento de agentes de aprendizado por reforço em ambientes com recompensa esparsa ou inexistente. Foi demonstrado que agentes utilizando a técnica de motivação intrínseca, ou curiosidade, conseguem resolver tarefas consideravelmente difíceis do domínio da robótica.

No Capítulo 1 foi exposto o problema da programação de comportamentos complexos em robôs e algumas soluções utilizando técnicas de aprendizado por reforço. Foi apresentado também como comumente estes problemas são formulados para o treinamento de agentes e o desafio do aprendizado em ambientes com recompensa esparsa, que é a motivação deste trabalho. A proposta do uso de um módulo de motivação intrínseca, inspirada por trabalhos recentes na área de aprendizado por reforço, também foi descrita.

No Capítulo 2 foram apresentados os fundamentos teóricos que foram base para a preparação e execução dos experimentos. Primeiramente, foram definidos os conceitos de aprendizado de máquina, redes neurais e aprendizado por reforço. Os fundamentos do algoritmo utilizado como base para este trabalho, o PPO, foram apresentados em seguida, assim como o funcionamento do módulo de motivação intrínseca para exploração, que foi o foco deste estudo.

No Capítulo 3 foram definidos os ambientes de *benchmark* selecionados e seu funcionamento foi descrito. O funcionamento completo do algoritmo utilizado e a comunicação entre os modelos de predição de valor do estado, da política e de predição do estado futuro também foram apresentados. Em seguida, a metodologia para os experimentos executados foi definida. Para a demonstração do impacto do módulo de curiosidade no aprendizado, o mesmo algoritmo com os mesmos hiperparâmetros foi executado duas vezes em cada ambiente - uma vez com o módulo de curiosidade e outra vez sem o mesmo.

Os resultados, apresentados no Capítulo 4, demonstram um comportamento exploratório capaz de executar complexas sequências de ações por parte do algoritmo proposto, permitindo-o alcançar o objetivo em todos os ambientes em que foi testado. Os gráficos apresentados demonstram também a tendência do agente a manter políticas exploratórias mesmo alcançando o objetivo geral de cada tarefa proposta. Essas tendências

indicam um comportamento possivelmente generalista que pode ser útil para aplicações em problemas que requerem uma abordagem multitarefa ou até de meta-aprendizado.

Nesse contexto, são propostas algumas direções para trabalhos futuros. Primeiramente, o impacto da motivação intrínseca em ambientes multitarefa e de meta-aprendizado pode ser analisado. *Benchmarks* propostos recentemente [44] se mostraram robustos para a avaliação dessas capacidades. Além disso, o estudo da capacidade de generalização do algoritmo para ambientes de teste diferentes dos quais o agente foi treinado é uma importante métrica a ser avaliada. Outras ideias para futuros trabalhos envolvem a aplicação de motivação intrínseca para controle em sistemas complexos e de característica hierárquica, assim como sua aplicação em sistemas de aprendizado multi-agente. Por fim, o uso de curiosidade para aprendizado de controle a partir de píxeis pode ser explorado, dado o sucesso da aplicação desta configuração em jogos [10] e pela facilidade e viabilidade do uso de câmeras em robôs reais.

Referências Bibliográficas

[1] ABBEEL, P. **Deep learning for robotics.**, 2019.

[2] ACHIAM, J. **Advanced policy gradient methods**. Disponível para visualização em: http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_13_advanced_pg.pdf . Acessado em 15 nov. 2019.

[3] AKKAYA, I.; ANDRYCHOWICZ, M.; CHOCIEJ, M.; LITWIN, M.; MCGREW, B.; PETRON, A.; PAINO, A.; PLAPPERT, M.; POWELL, G.; RIBAS, R.; SCHNEIDER, J.; TEZAK, N.; TWOREK, J.; WELINDER, P.; WENG, L.; YUAN, Q.; ZAREMBA, W.; ZHANG, L. **Solving rubik's cube with a robot hand**, 2019.

[4] AMODEI, D.; OLAH, C.; STEINHARDT, J.; CHRISTIANO, P.; SCHULMAN, J.; MANÉ, D. **Concrete problems in ai safety**, 2016.

[5] ANDRYCHOWICZ, M.; BAKER, B.; CHOCIEJ, M.; JÓZEFOWICZ, R.; MCGREW, B.; PACHOCKI, J.; PETRON, A.; PLAPPERT, M.; POWELL, G.; RAY, A.; ET AL.. **Learning dexterous in-hand manipulation**. *The International Journal of Robotics Research*, p. 027836491988744, Nov 2019.

[6] ANDRYCHOWICZ, M.; WOLSKI, F.; RAY, A.; SCHNEIDER, J.; FONG, R.; WELINDER, P.; MCGREW, B.; TOBIN, J.; ABBEEL, P.; ZAREMBA, W. **Hindsight experience replay**, 2017.

[7] ARCADU, F.; BENMANSOUR, F.; MAUNZ, A.; WILLIS, J.; HASKOVA, Z.; PRUNOTTO, M. **Deep learning algorithm predicts diabetic retinopathy progression in individual patients**. *npj Digital Medicine*, 2(1):92, 2019.

[8] ASHRAF, M. **Reinforcement learning demystified: Markov decision processes**. Disponível para visualização em: https://towardsdatascience.com/reinforcement-learning-demystified-markov-decision-processes-part-1-bf00dda41690 2018. Acessado em 13 nov. 2019.

[9] BROCKMAN, G.; CHEUNG, V.; PETTERSSON, L.; SCHNEIDER, J.; SCHULMAN, J.; TANG, J.; ZAREMBA, W. **Openai gym**, 2016.

[10] BURDA, Y.; EDWARDS, H.; PATHAK, D.; STORKEY, A.; DARRELL, T.; EFROS, A. A. **Large-scale study of curiosity-driven learning**, 2018.

[11] C. J. C. H. WATKINS. **Learning from Delayed Rewards**. PhD thesis, King's College, 1989.

[12] CARDOSO, F. C.; GUIMARÂES, T. A.; DE SOUSA DÂMASO, R.; DE SOUSA COSTA, E. **Kinematic and dynamic behavior of articulated robot manipulators by two bars**. ABCM Symposium Series in Mechatronics - Vol. 5, 2012.

[13] CLARK, J.; AMODEI, D. **Faulty reward functions in the wild**. Disponível para visualização em: https://openai.com/blog/faulty-reward-functions/ . Acessado em 20 nov. 2019.

[14] DOGGIES, S. **Automated assembly line**. Disponível para visualização em: https://iptmajor.weebly.com/automated-assembly-line-robotic-arms.html , 2019. Acessado em 17 nov. 2019.

[15] FENG, S.; WHITMAN, E.; XINJILEFU, X.; ATKESON, C. **Optimization based full body control for the atlas robot**. 2015:120–127, 02 2015.

[16] FISHER, C. K.; SMITH, A. M.; WALSH, J. R.; SIMON, A. J.; EDGAR, C.; JACK, C. R.; HOLTZMAN, D.; RUSSELL, D.; HILL, D.; GROSSET, D.; WOOD, F.; VANDERSTICHELE, H.; MORRIS, J.; BLENNOW, K.; MAREK, K.; SHAW, L. M.; ALBERT, M.; WEINER, M.; FOX, N.; AISEN, P.; COLE, P. E.; PETERSEN, R.; SHERER, T.; KUBICK, W. **Machine learning for comprehensive forecasting of alzheimer's disease progression**. *Scientific Reports*, 9(1):13622, 2019.

[17] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. http://www.deeplearningbook.org .

[18] JOHANNINK, T.; BAHL, S.; NAIR, A.; LUO, J.; KUMAR, A.; LOSKYLL, M.; OJEA, J. A.; SOLOWJOW, E.; LEVINE, S. **Residual reinforcement learning for robot control**. In: *2019 International Conference on Robotics and Automation (ICRA)*, p. 6023–6029, May 2019.

[19] LAPAN, M. **Deep Reinforcement Learning Hands-On**. Packt Publishing, Birmingham, UK, 2018.

[20] LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. **Backpropagation applied to handwritten zip code recognition**. *Neural Comput.*, 1(4):541–551, Dec. 1989.

[21] MELO, L. C. **A deep reinforcement learning method for humanoid kick motion**. Instituto Tecnológico de Aeronáutica, São José dos Campos, 2018.

[22] MEYER, J.; WILSON, S. W. **A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers**, p. 222–227. MITP, 1991.

[23] MEYES, R.; TERCAN, H.; ROGGENDORF, S.; THIELE, T.; BÜSCHER, C.; OBDENBUSCH, M.; BRECHER, C.; JESCHKE, S.; MEISEN, T. **Motion planning for industrial robots using reinforcement learning**. *Procedia CIRP*, 63:107–112, 12 2017.

[24] MNIH, V.; BADIA, A. P.; MIRZA, M.; GRAVES, A.; LILLICRAP, T. P.; HARLEY, T.; SILVER, D.; KAVUKCUOGLU, K. **Asynchronous methods for deep reinforcement learning**, 2016.

[25] MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G.; PETERSEN, S.; BEATTIE, C.; SADIK, A.; ANTONOGLOU, I.; KING, H.; KUMARAN, D.; WIERSTRA, D.; LEGG, S.; HASSABIS, D. **Human-level control through deep reinforcement learning**. *Nature*, 518(7540):529–533, 2015.

[26] NG, A. Y.; HARADA, D.; RUSSELL, S. J. **Policy invariance under reward transformations: Theory and application to reward shaping**. In: *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, p. 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[27] NIELSEN, M. A. **Neural Networks and Deep Learning**. Determination Press, 2015.

[28] PAN, Y.; CHENG, C.-A.; SAIGOL, K.; LEE, K.; YAN, X.; THEODOROU, E.; BOOTS, B. **Agile autonomous driving using end-to-end deep imitation learning**, 2017.

[29] PATHAK, D.; AGRAWAL, P.; EFROS, A. A.; DARRELL, T. **Curiosity-driven exploration by self-supervised prediction**. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jul 2017.

[30] PENG, X. B.; BERSETH, G.; YIN, K.; VAN DE PANNE, M. **Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning**. *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)*, 36(4), 2017.

[31] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1**. chapter Learning Internal Representations by Error Propagation, p. 318–362. MIT Press, Cambridge, MA, USA, 1986.

[32] RUSSEL, S.; NORVIG, P. **Artificial Intelligence: a Modern Approach**. Prentice-Hall, 1995.

[33] RYAN, R.; DECI, E. **Intrinsic and extrinsic motivations: Classic definition and new directions**. *Contemporary Educational Psychology*, 25:54–67, 02 2000.

[34] RYAN, R. M.; SILVIA, P. J. **Curiosity and motivation**, 09 2012.

[35] SCHULMAN, J.; MORITZ, P.; LEVINE, S.; JORDAN, M.; ABBEEL, P. **High-dimensional continuous control using generalized advantage estimation**, 2015.

[36] SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. **Proximal policy optimization algorithms**, 2017.

[37] SIMON, P. **Too big to ignore: The business case for big data**. 2013.

[38] SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. The MIT Press, Cambridge, MA, 1998.

[39] SUTTON, R. S.; MCALLESTER, D.; SINGH, S.; MANSOUR, Y. **Policy gradient methods for reinforcement learning with function approximation**. In: *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, p. 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

[40] TOBIN, J.; FONG, R.; RAY, A.; SCHNEIDER, J.; ZAREMBA, W.; ABBEEL, P. **Domain randomization for transferring deep neural networks from simulation to the real world**. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep 2017.

[41] VIRTANEN, P.; GOMMERS, R.; OLIPHANT, T. E.; HABERLAND, M.; REDDY, T.; COURNAPEAU, D.; BUROVSKI, E.; PETERSON, P.; WECKESSER, W.; BRIGHT, J.; VAN DER WALT, S. J.; BRETT, M.; WILSON, J.; JARROD MILLMAN, K.; MAYOROV, N.; NELSON, A. R. J.; JONES, E.; KERN, R.; LARSON, E.; CAREY, C.; POLAT, İ.; FENG, Y.; MOORE, E. W.; VAND ERPLAS, J.; LAXALDE, D.; PERKTOLD, J.; CIMRMAN, R.; HENRIKSEN, I.; QUINTERO, E. A.; HARRIS, C. R.; ARCHIBALD, A. M.; RIBEIRO, A. H.; PEDREGOSA, F.; VAN MULBREGT, P.; CONTRIBUTORS, S. . . **SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python**. *arXiv e-prints*, p. arXiv:1907.10121, Jul 2019.

[42] WILLIAMS, R. J.; PENG, J. **Function optimization using connectionist reinforcement learning algorithms**. 1991.

[43] YANG, S.; WANG, W.; LIU, C.; DENG, W.; HEDRICK, J. K. **Feature analysis and selection for training an end-to-end autonomous vehicle controller using deep learning approach**. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*, p. 1033–1038, June 2017.

[44] YU, T.; QUILLEN, D.; HE, Z.; JULIAN, R.; HAUSMAN, K.; FINN, C.; LEVINE, S. **Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning**. In: *Conference on Robot Learning (CoRL)*, 2019.

[45] ZHOU, X.; ZHU, F.; LIU, Q.; FU, Y.; HUANG, W. **A sarsa()-based control model for real-time traffic light coordination**. *TheScientificWorldJournal*, 2014:759097, 01 2014.

APÊNDICE A

Hiperparâmetros

Tabela A.1: *Hiperparâmetros para o algoritmo PPO e módulo de motivação intrínseca.*

Hiperparâmetro	Valor
Iterações (em cada ambiente)	$3x10^7$
Neurônios nas camadas escondidas	128
Taxa de aprendizado do modelo de valor	10^{-4}
Taxa de aprendizado do modelo da política	10^{-5}
Taxa de aprendizado do modelo de futuro	10^{-5}
Coeficiente de entropia	0.01
Recompensa intrínseca máxima	1
Tamanho do episódio	2048
Tamanho do lote	32
Épocas	10
γ	0.95
ε (PPO)	0.3
λ (GAE)	0.95
η (recompensa intrínseca)	1
Otimizador	Adam