UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Kenedy Felipe dos Santos da Silva

Teacher-Student aplicado a Diferentes Modelos de Recompensa do ambiente *Lunar Lander*

Recife

2021

Kenedy Felipe dos Santos da Silva

Teacher-Student aplicado a Diferentes Modelos de Recompensa do ambiente *Lunar Lander*

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Computação

Curso de Bacharelado em Ciência da Computação

Orientador: Pablo Azevedo Sampaio

Recife

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

S586t Silva, Kenedy Felipe dos Santos da
Teacher-Student aplicado a Diferentes Modelos de Recompensa do ambiente Lunar Lander / Kenedy Felipe dos Santos da Silva. - 2021.
37 f. : il.

Orientador: Pablo Azevedo Sampaio.
Inclui referências.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Ciência da Computação, Recife, 2021.

1. aprendizagem por reforço. 2. algoritmo teacher-student. 3. aprendizado por currículo. 4. modelagem de recompensa. I. Sampaio, Pablo Azevedo, orient. II. Título

CDD 004

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
http://www.bcc.ufrpe.br

FICHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

Trabalho defendido por Kenedy Felipe dos Santos da Silva às 14 horas do dia 20 de julho de 2021, no link https://meet.google.com/zav-uoza-dio, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado Teacher-Student aplicado a Diferentes Modelos de Recompensa do ambiente Lunar Lander, orientado pelo Prof. Pablo Azevedo Sampaio e aprovado pela seguinte banca examinadora:

Pablo Azevedo Sampaio

DC/UFRPE

George Gomes Cabral

DC/UFRPE

À Deus, que me deu toda sabedoria e a oportunidade de aprender cada dia mais, a quem está comigo todos os dias me apoiando nessa caminhada, minha esposa Rayane Gomes, ao meu pai, a minha mãe, a toda a minha família, a todos que contribuíram de forma direta e indireta no meu aprendizado e ao professor Pablo Sampaio.

Agradecimentos

Agradeço à Deus pela força, pelo aprendizado, pelo dia a dia, a minha amada e querida companheira Rayane Gomes por toda paciência e ajuda, ao meu pai Haroldo Pereira que me deu o dom de gostar de computação desde pequeno, a minha mãe Vânia Lúcia pelos ensinamentos e batalhas vencidas todos os dias, a minha avó Maria José que contribuiu muito na minha educação, a orientação do professor Pablo Sampaio com toda paciência e apoio, aos professores que passaram todo aprendizado no decorrer da minha formação, aos meus amigos de sala de aula e de projetos, Ikaro Alef, João Lucas, Tássia Barros, Jonatan Washington, Renilson Albuquerque, entre muitos outros que contribuíram e auxiliaram na minha jornada.

“Não podemos ensinar a outra pessoa diretamente;
só podemos facilitar sua aprendizagem.”
(Carl Rogers)

Resumo

As técnicas estudadas relacionadas à aprendizagem por reforço estão se tornando cada vez mais comuns em desafios do mundo real, porém um desafio é reduzir o tempo de aprendizado. Atualmente o tempo de aprendizado e/ou quantidade de interações realizadas pelo agente de aprendizagem por reforço podem resultar em altos custos nas aplicações, pois o treinamento dos modelos podem consumir bastante tempo, exigindo muitas interações do agente com o ambiente da tarefa. Este trabalho busca melhorar o aprendizado utilizando uma nova combinação de técnicas, a técnica *Teacher-Student* (Aluno-Professor) com a *Reward Shaping* (modelagem de recompensas). A técnica *Teacher-Student* visa escolher dentre um conjunto de tarefas similares que treinam para uma tarefa principal, de acordo com o aprendizado do aluno. A técnica *Reward Shaping*, altera a recompensa para tentar acelerar o aprendizado, fornece feedbacks mais frequentes sobre os comportamentos apropriados, ou seja, reporta recompensas com mais frequência. Adaptamos algoritmos de *Teacher-Student* para essa combinação de técnicas, e usamos o ambiente *Lunar Lander* como estudo de caso, usando quatro modelos de recompensa elaborados em (ALBUQUERQUE, 2021) para este ambiente. Foram realizados experimentos executando diferentes treinamentos para comparar essa abordagem com o treinamento apenas no *Lunar Lander* original (sem alteração das recompensas), e com os resultados obtidos em (ALBUQUERQUE, 2021) ao adotar cada um dos modelos de recompensas individualmente. A combinação das técnicas *Teacher-Student* com *Reward Shaping* contribuíram para uma nova experiência na área de aprendizagem por reforço, conseguindo acelerar o aprendizado do agente, considerando a duração de 600 mil passos de treinamento, atingindo o desempenho alvo em 2 de 5 propostas, além de conseguir aprender melhor que a abordagem original do *Lunar Lander* com algoritmo PPO.

Palavras-chave: aprendizagem por reforço, algoritmo teacher-student, aprendizado por currículo, modelagem de recompensas.

Abstract

The techniques studied related to learning by reinforcement are becoming more and more common in real world challenges, but one challenge is to reduce the learning time. Currently, the learning time and/or amount of interactions performed by the reinforcement learning agent can result in high costs in applications, as the training of models can consume a lot of time, requiring many interactions between the agent and the task environment. This work seeks to improve learning using a new combination of techniques, the Teacher-Student technique with Reward Shaping. The Teacher-Student technique aims to choose among a set of similar tasks that train for a main task, according to the student's learning. The Reward Shaping technique, altering the reward to try to accelerate learning, provides more frequent feedback on appropriate behaviors, that is, reports rewards more often. We adapted Teacher-Student algorithms for this combination of techniques, and used the Lunar Lander environment as a case study, using four reward models designed in (ALBUQUERQUE, 2021) for this environment. Experiments were performed running different trainings to compare this approach with training only on the original Lunar Lander (no rewards change), and with the results obtained in (ALBUQUERQUE, 2021) by adopting each of the rewards models individually. The combination of Teacher-Student techniques with Reward Shaping contributed to a new experience in the reinforcement learning area, managing to accelerate the agent's learning, considering the duration of 600 thousand training steps, reaching the target performance in 2 out of 5 proposals, in addition to of being able to learn better than the original Lunar Lander approach with PPO algorithm.

Keywords: reinforcement learning, teacher-student, reward shaping, curriculum learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AR	Aprendizagem por Reforço
DDPG	Deep Deterministic Policy Gradient
FIFO	First-in First-out
GR	Gradiente de Política
IA	Inteligência Artificial
MDP	Processo de Decisão de Markov
POMDP	Processo de decisão de Markov parcialmente observável
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
SAC	Soft Actor Critic
TD3	Twin Delayed DDPG
TS	Teacher Student
TSCL	Teacher-Student Curriculum Learning

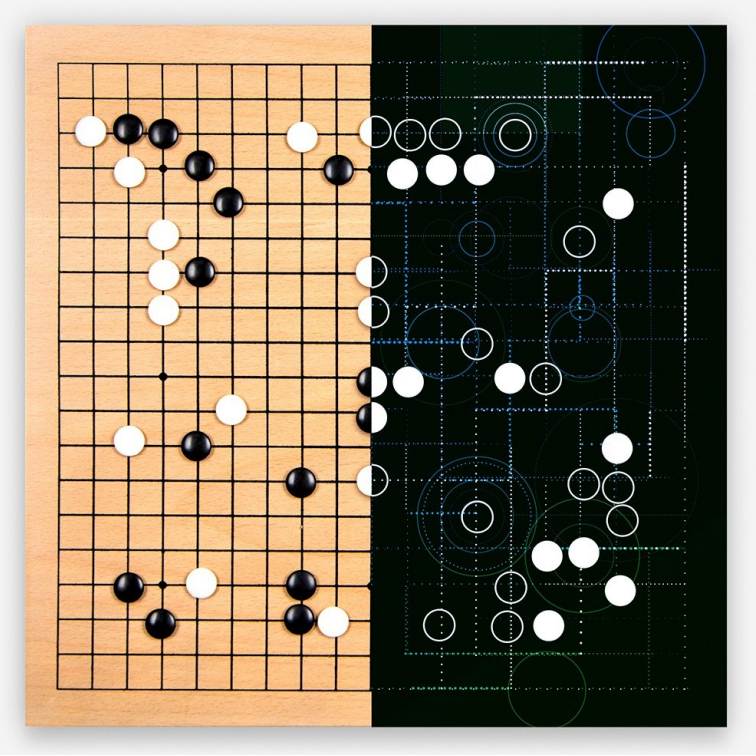
Sumário

1 Introdução

A inteligência artificial (IA) têm sido aplicada em todas as áreas e o *Reinforcement Learning* (Aprendizado por Reforço - AR) está provando cada vez mais que pode alcançar resultados inesperados e muito diferente dos humanos em diferentes ambientes. Pode-se considerar que o Aprendizado por Reforço abrange toda a IA: um agente é colocado em um ambiente e tem de aprender a se comportar com sucesso nesse ambiente. Destaque para AR hoje em dia são grandes resultados conquistados recentemente, tornando-se mundialmente famoso graças a desempenhos sobre-humano em vários videogames, destaque em jogos de Atari como *Pong*, *Enduro*, *Montezuma's Revenge* (BELLEMARE et al., 2016), e até mesmo batendo o campeão mundial do jogo *Go* (Figura 1) (SILVER et al., 2016).

Aprendizagem por reforço possui muitos parâmetros, e varia muito o seu desempenho devido os algoritmos serem muito sensível aos parâmetros, ou seja, alterar um parâmetro afeta o aprendizado. AR sofre com uma baixa eficiência de amostragem, ou seja ele requer um grande numero de interações com o ambiente para poder aprender

Figura 1 – Jogo Go



Fonte: AlphaGo and the future of go: interviews with top European players (KIM OUWELEEN, 2016).

de maneira eficaz. As técnicas atuais de Aprendizagem por Reforço ainda sofrem com a necessidade de uma grande quantidade de dados de interação, o que pode resultar em custos inviáveis em muitas aplicações.

A tomada de decisões é uma atividade básica em nossa vida cotidiana. É também uma característica essencial dos agentes inteligentes. Particularmente, a tomada de decisão para uma meta de longo prazo requer a inteligência de uma visão de longo prazo e comportamentos menos gananciosos; a tomada de decisão em um ambiente desconhecido requer a inteligência de adaptação do ambiente. Aprendizagem por reforço (SUTTON; BARTO, 2018) estuda a tomada de decisão para objetivos de longo prazo em ambientes desconhecidos, portanto, está no centro da Inteligência Artificial.

1.1 Aplicações Recentes

As tarefas do mundo real ainda são desafios que o Aprendizado por Reforço está desbravando, muitas pesquisas têm sido realizadas melhorando o Aprendizado por Reforço e tornando-o cada vez mais inteligente, seguro e eficiente. Alguns casos mais populares são sistemas de recomendações. A Netflix anunciou publicamente que está usando a AR para recomendar séries e filmes aos seus usuários e, os pesquisadores da Netflix publicam regularmente artigos usando AR. Outras marcas que também utilizaram foram Amazon, eBay, Mercado Livre e muitos outros e-commerce (ARGERICH, 2020).

Podemos citar algumas aplicações que atualmente utilizam o Aprendizado por Reforço, estas são abrangentes em muitas áreas. Podemos observar aplicações em robótica, carros autônomos, jogos, controladores de frotas, sistemas web, semáforos de trânsito, recomendações, automação industrial, planejamento de estratégia de negócios, processamento de dados entre outros. (SHOHAM; LEYTON-BROWN, 2008).

Várias empresas estão usando aprendizado por reforço para controlar robôs em indústrias. Um dos casos de uso desta área mais comuns citados na pesquisa é para controlar robôs ou peças de robôs, como a mão de um robô; a aplicação de AR à robótica inspirou cientistas ao longo dos anos e também oferece um ótimo cenário de validação para algoritmos de Aprendizagem por Reforço. Os robôs moldaram a indústria, reduzindo custos e, ao mesmo tempo, reduzindo os tempos de produção, e a AR promete levar isso ainda mais longe: a Aprendizagem por Reforço oferece uma maneira de controlar as peças do robô com um alto nível de precisão e com comportamentos complexos que são muito difíceis de programar. O crescimento está sendo notado em todas as áreas e fomentando cada vez mais as pesquisas dentro dos negócios (ARGERICH, 2020).

1.2 Desafios do Mundo Real

Sabendo que a IA está simplesmente progredindo em tempo recorde, o Aprendizado Supervisionado e não Supervisionado também tenham se espalhado por quase todos os setores, o Aprendizado por Reforço foi rejeitado até agora. Isso provavelmente ocorre porque o Aprendizado por Reforço apresenta vários desafios, como a ineficiência de amostragem (quando o algoritmo não consegue tirar o máximo proveito de cada amostra), o aprendizado seguro e a definição de uma boa função de recompensa que deve ser abordada antes de implementar soluções de Aprendizagem por Reforço para problemas do mundo real, ou seja, porque está área de pesquisa está no seu início.

Desafio é a palavra-chave de acordo com os pesquisadores atuais. Alguns desses desafios são descritos em *"Challenges of Real-World Reinforcement Learning"* (DULAC-ARNOLD; MANKOWITZ; HESTER, 2019), que mostra desafios que tornam a Aprendizagem por Reforço no mundo real mais difícil do que a Aprendizagem por Reforço na pesquisa. Na pesquisa *"Reinforcement Learning in Robotics: Applications and Real-World Challenges"* (KORMUSHEV; CALINON; CALDWELL, 2013), inúmeros desafios enfrentados pela representação de políticas em robótica são identificados e são exemplificados em aplicações do aprendizado por reforço para robôs do mundo real.

E os desafios de aplicar Aprendizagem por Reforço só cresce, com esse pensamento ficamos destinados a superar cada vez mais em desenvolvimento ações que consigam tratar destes problemas, buscando métodos que sejam implementados em diferentes soluções, realizando a configuração de sistemas e serviços, condução autônoma e vários outros métodos. Se pensarmos que esta é apenas a ponta do iceberg de todas as coisas que a Aprendizagem por Reforço pode fazer por nós, não há dúvida de que teremos muito trabalho e muitas pesquisas a realizar para superar esses desafios.

Desafios como falta de acesso aos dados, onde muitas empresas falham na coleta de dados pela deficiência na comunicação entre os setores, diferenças nos formatos e também variações na segurança e privacidade. Modelagem simples e barato que possa resolver um grande percentual das necessidades do cliente, podendo ser enviado em poucos meses, ele já é bem útil e terá impacto positivo na rentabilidade (DVB, 2019). Além de ajustes entre as áreas, como uma maior conexão entre TI, engenharia e ciência de dados, e também maior aceitação dos projetos pelos executivos, aumento da produtividade e otimização do tempo.

1.3 Técnicas e Abordagens

O Aprendizado por Currículo (*Curriculum Learning*) em estudos atuais obtiveram resultados satisfatórios quando se trata de Aprendizagem por Reforço. Os resultados com a utilização do currículo ajuda a otimização de recompensas esparsas ou de tarefas maiores e mais longas, podendo assim superar desafios como fases de jogos, procedimentos de serviços sequenciais. Os currículos possuem vários ambientes de treinamento para o mesmo agente. O aprendizado começa com atividades mais simples, aprendendo aspectos mais fáceis da tarefa ou sub-tarefas menores. Quando o conhecimento evolui, gradualmente é aumentado o grau de dificuldade ou são utilizadas novas sub-tarefas (BENGIO et al., 2009).

O *Teacher-Student* (TS) é uma classe de algoritmos que realiza a escolha das tarefas que serão treinadas pelo agente. A ideia principal é que o aluno pratique mais as tarefas nas quais está progredindo mais rapidamente, ou seja, a inclinação da curva de aprendizado é maior. Para combater o esquecimento, o aluno também deve praticar tarefas em que o desempenho esteja piorando, ou seja, a inclinação da curva de aprendizagem é negativa (MATIISEN et al., 2019).

O TS é uma versão dinâmica do aprendizado por currículo. Que possui vários ambientes, com variados níveis de dificuldade ou diferentes sub-tarefas. O TS poderá realizar a escolha dinâmica dos ambientes, visando assim melhorar o desempenho com a utilização de menos parâmetros, enquanto currículo tem uma ordem estática de apresentação dos ambientes. Em (MATIISEN et al., 2019), os autores formalizaram o *Teacher-Student* para um processo de decisão de Markov parcialmente observável (POMDP). A inspiração para os algoritmos de TS de Matiisen é que a melhor tarefa é aquela em que ele está aprendendo ou desaprendendo mais. Aumentando ou diminuindo o desempenho em termos de recompensa cumulativa.

1.4 Contribuições

A principal contribuição deste trabalho consiste na implementação da abordagem Teacher-Student para aplicá-la de formas diferentes com diferentes versões de recompensas, com objetivo de melhorar o desempenho com os modelos individuais. Buscando responder a nossa pergunta de pesquisa: *"Qual o efeito no aprendizado de um agente ao se criar currículos dinâmicos em ambientes que possuem vários modelos de recompensa?"*

Neste trabalho foi realizada a combinação de duas técnicas, *Teacher-Student* com *Reward Shaping* (Modelagem de Recompensa) de uma maneira nova, para tentar acelerar o treinamento de um agente para o ambiente *LunarLander-v2*, usando o algo-

ritmo de treinamento PPO (*Proximal Policy Optimization*). Adaptamos dois algoritmos da abordagem *Teacher-Student*, inspirados em (MATIISEN et al., 2019), criando assim uma nova variante do *Teacher-Student*.

Os ambientes que escolhemos para serem selecionadas são diferentes versões do ambiente *LunarLander-v2*, usando diferentes modelagens da recompensa. Os diferentes modelos foram propostos no trabalho de Renilson Albuquerque (ALBUQUERQUE, 2021) (estudo realizado em paralelo a este trabalho). Foram realizados experimentos para realizar a comparação do uso das abordagens com: O treinamento completo no ambiente original e o treinamento completo apenas usando os modelos de recompensa.

2 Fundamentação Teórica

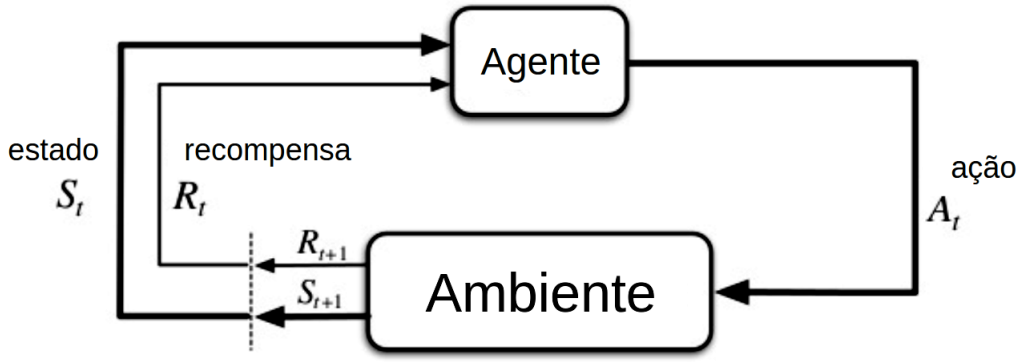
2.1 Introdução a Aprendizagem por Reforço

Atualmente as áreas da IA como Aprendizado Supervisionado e Aprendizado Não Supervisionado são as mais citadas, Aprendizado supervisionado é o conceito em que você tem um vetor contendo os dados de entrada com o valor-alvo correspondente (as saídas). Por outro lado, aprendizado não supervisionado é o quando a informação contém apenas os dados de entrada sem o valor-alvo correspondente. Em comparação com o Aprendizado por Reforço, o agente não tem informações preditivas, ele aprende conforme interage com o ambiente.

Aprendizagem por reforço é um paradigma de aprendizagem que busca aprender a controlar um sistema de forma a maximizar uma medida de desempenho numérica que expressa um objetivo de longo prazo. O agente de aprendizado não é informado sobre qual ação realizar, em vez disso, ele deve descobrir qual ação trará mais recompensa ao executá-las. As recompensas para as ações podem ser imediata ou na próxima ação, sendo duas características importantes do Aprendizado por Reforço: pesquisa por tentativa e erro e recompensa atrasada (SUTTON; BARTO, 2018).

A ideia básica é simplesmente capturar os aspectos mais importantes do problema real que um agente de aprendizado enfrenta interagindo com seu ambiente para atingir uma meta. Essa interação pode ser vista na figura 2. Além do agente e do ambiente, podem-se identificados quatro subelementos principais de um sistema de Aprendizagem por Reforço: uma política (que é o mapeamento de estados em

Figura 2 – Interação agente-ambiente na aprendizagem por reforço.



Fonte: Reinforcement learning: An introduction (SUTTON; BARTO, 2018).

probabilidades de escolher possíveis ações, buscando ser uma política que maximize a recompensa (ARULKUMARAN et al., 2017)), um sinal de recompensa (informação positiva, em caso de decisão correta ou negativa, em caso de decisão incorreta, para ação que o agente realizou), uma função de valor (especifica o que é bom no longo prazo. O valor de um estado é a quantidade total de recompensa que um agente pode esperar acumular no futuro) e, um modelo do ambiente (é algo que imita o comportamento do ambiente, ou mais geralmente, que permite fazer inferências sobre como o ambiente se comportará) (SUTTON; BARTO, 2018).

Em (PUTERMAN, 1990) é descrito o comportamento e significado de cada etapa que compõe o processo. A maioria das técnicas assume que o ambiente funciona como um Processo de Decisão de Markov (*Markov Decision Process* - MDP). A política é um mapeamento de estados percebidos do meio ambiente para ações a serem tomadas quando estiverem nesses estados. A política define a forma de comportamento do agente de aprendizagem em um determinado momento. A política ótima do MDP citada antes, é exatamente a política desejada para o agente.

Em geral, as políticas podem ser estocásticas, especificando probabilidades para cada ação. Um sinal de recompensa define o objetivo de um problema de Aprendizagem por Reforço. Em cada instante de tempo, o ambiente envia ao agente de AR um único número chamado de recompensa. O único objetivo do agente é maximizar a recompensa total que recebe a longo prazo. O sinal de recompensa, portanto, define quais são os eventos bons e ruins para o agente.

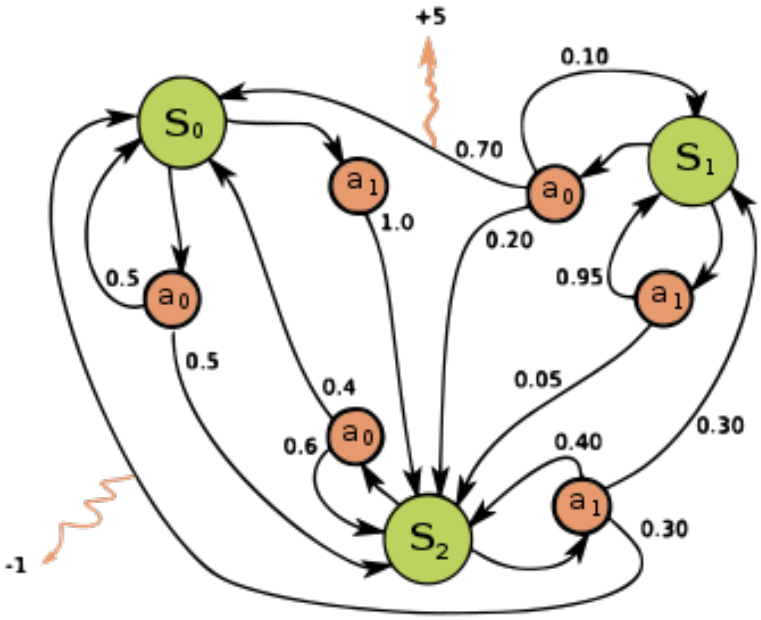
2.2 Processo de Decisão de Markov (MDP)

Os MDP, são baseados nas noções de estado, eles assumem número finito de estados e ações. O estado representa a situação atual do agente em um ambiente e a ação é um evento que afeta o processo. A cada momento o agente observa um estado e executa uma ação que gera uma recompensa que buscam serem maximizadas (PUTERMAN, 1990).

MDP é uma tupla (S, A, R, p), onde S é o conjunto de estados subjacentes, A é o conjunto de ações, R é a recompensa imediata e p é a probabilidade de que a ação a no estado s_1 desloque até o estado s_2. Enquanto não atinge um estado terminal, cada ação produz uma recompensa e leva a um novo estado. O objetivo é maximizar as recompensas acumuladas (esperadas) (SMITH; SIMMONS, 2012).

Uma solução para um MDP é fornecida em termos de uma política π, que mapeia cada estado para uma ação a ser executada neste estado. A política pode ser realizada em alguns casos através de uma tabela de pesquisa, em outros podem envolver cálculos. Aplicar a ação a ao estado s gera recompensas $R(s, a)$ como citado

Figura 3 – Exemplo de um processo de decisão de Markov



Fonte: Markov Decision Process (WIKIPEDIA, 2021).

anteriormente, buscando maximizá-las, um exemplo pode ser visto na figura 3.

2.3 Algoritmos de Gradiente de Política

Os métodos Gradiente de Política - GR visam otimizar e modelar diretamente a política. A modelagem da política geralmente é realizada com uma função parametrizada em relação ao θ na função $\pi_\theta(a|s)$. O valor da função objetivo, a recompensa, depende dessa política. Os parâmetros representados θ são ajustados usando um mecanismo de gradiente descendente para otimizar a função de recompensa total. Em busca de melhores resultados estes métodos possuem um grande desafio, pois eles estão ligados diretamente ao tamanho do episódio (são todos os estados que ocorrem do estado inicial até o estado terminal), por exemplo, um pouso de uma nave. Quando pequenos o mesmo possui um progresso extremamente lento, em caso de episódios grandes o sinal é sobrecarregado pelo ruído, e possíveis perdas de desempenho ao longo do episódio (SCHULMAN et al., 2017).

Detalhando temos θ para representar os parâmetros da política (que podem ser, por exemplo, os pesos da rede neural) e θ_t para retratar θ no iteração t. Podemos denominar a probabilidade de tomar uma ação a no estado s de $\pi_\theta(a|s)$, com o objetivo de atualização de θ à θ_{t+1}, de uma maneira que eventualmente atinja a política ideal.

Novas técnicas tem surgido, entre elas o uso de algoritmos de AR com redes neurais profundas. Em linhas gerais, os algoritmos criados seguem o padrão de AR. Alguns bastante utilizados e conhecidos são DDPG (Deep Deterministic Policy Gradient), SAC (*Soft Actor Critic*), PPO (*Proximal Policy Optimization*).

Os métodos *policy* utilizam um tipo de treinamento por gradiente descendente para otimizar diretamente a política, tiveram muitos avanços recentes. Podemos citar o DDPG que que é da família Gradiente de Política (*Policy Gradient*) e permite treinar políticas com ações contínuas, com bons resultados.

O SAC é um algoritmo que otimiza uma política estocástica. Uma característica central do SAC é a politica treinada para maximizar uma combinação entre a recompensa total esperada e a entropia, uma medida de aleatoriedade na política. A rede é forçada a não deixar uma ação com probabilidade muito mais alta que a outra, a não ser que seja muito melhor (em termos de recompensas). Isso faz com que a política seja forçada a explorar (explore) (HILL et al., 2018).

PPO é motivado pela pergunta: “Como podemos dar o maior passo de aprimoramento possível em uma política usando os dados que temos atualmente, sem avançar tão longe que causamos acidentalmente um colapso no desempenho?”. A ideia principal é que, após uma atualização, a nova política não fique muito longe da política antiga. Ou seja, que a probabilidade de uma ação não mude além de um certo percentual (dado como parâmetro) por atualização da rede da política (HILL et al., 2018).

Uma implementação foi realizada pela *OpenAI* (Instituição de pesquisa que tem como objetivo promover e desenvolver IA e que tem produzido vários avanços importantes) feita para GPU (Unidade de Processamento de Gráficos) chamada de PPO2. O PPO2 contém várias modificações do algoritmo original não documentadas pela *OpenAI*, com desempenho comparável ou melhor do que as abordagens de ponta, ao mesmo tempo em que é muito mais simples de implementar e ajustar. O PPO se tornou o algoritmo padrão de Aprendizagem por Reforço na *OpenAI* devido à sua facilidade de uso e bom desempenho (BROCKMAN et al., 2016).

3 Técnicas de Aprendizagem

3.1 Aprendizado por Currículo

O conceito de currículo de aprendizado, foi proposto por (BENGIO et al., 2009) fazendo uma analogia com a forma como nós, humanos, aprendemos. É preciso de décadas para realizar treinamentos junto a sociedade, para que os humanos sejam treinados como adultos totalmente funcionais. A formação é baseada em um sistema de ensino e um currículo que realiza a inclusão de diferentes conceitos e momentos, explorando informações previamente aprendidas para facilitar o novo aprendizado e novas abstrações. Ao escolher um currículo para apresentar exemplos e a ordem das informações para o sistema de aprendizagem, é possível observar e orientar treinamento aumentando a velocidade em que o aprendizado pode ocorrer.

A ideia da aprendizagem curricular é começar pequeno, aprender as tarefas mais fáceis ou sub-tarefas, aumentar posteriormente o nível de dificuldade de modo gradual (ELMAN, 1993). Os resultados obtidos são baseados no aprendizado de uma tarefa ou atividade simples com uma rede recorrente, a base de aprendizado influência diretamente na forma e qualidade de um aprendizado bem sucedido, pois a estrutura em que ela é apresentada para o aluno expande seus recursos gradualmente à medida que se aprende.

As informações descritas por (BENGIO et al., 2009; ELMAN, 1993) são importantes para a psicologia do desenvolvimento, porque ilustram o valor adaptativo de começar, como fazem os bebês humanos, com um estado inicial mais simples e, então, construir sobre isso para desenvolver representações cada vez mais sofisticadas da estrutura. Como (KRUEGER; DAYAN, 2009) cita, a questão de orientar a aprendizagem de uma rede neural recorrente para aprender uma linguagem simples e aumentar sua capacidade ao longo do caminho foi recentemente revisada a partir da perspectiva cognitiva.

No artigo *Google Research Football: A Novel Reinforcement Learning Environment* (KURACH et al., 2019), ele oferece vários ambientes simples que fazem parte de um jogo completo. Neste artigo que apresenta o ambiente *Football Academy*, só são reportados resultados para treinamentos em um único cenário por vez, ideia que leva a criação de currículo de aprendizado, o que foi realizado em (SILVA, 2019).

Com o objetivo de acelerar o treinamento, se comparado com os resultados reportados pelo Google (KURACH et al., 2019). Foi utilizado o aprendizado por currículo para o treinamento de um agente nos cenários *Empty Goal Close*, onde o jogador está

na área do gol vai até o gol vazio, *Empty Goal*, onde o jogador vai do meio campo até o gol vazio, *Run to Score*, no qual o jogador vai do meio do campo até o gol vazio porém com adversários em campo, e no cenário *Run to Score with Keeper* que é quando o jogador sai do meio do campo em direção ao gol com adversários em campo e um goleiro no gol adversário. Os resultados obtidos em (SILVA, 2019) leva a um melhor resultado quando comparado ao resultados reportados em (KURACH et al., 2019).

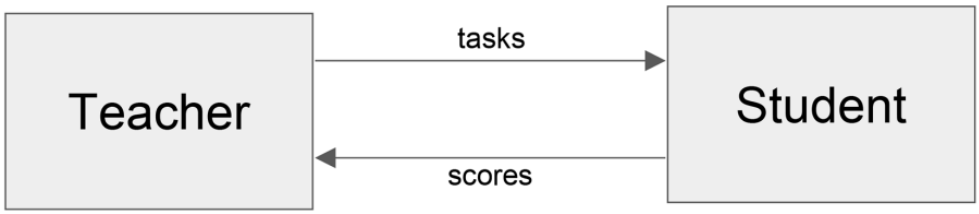
Um ponto importante a destacar é que treinamentos de agentes de aprendizado por reforço exige uma grande quantidade de tempo. Cada treinamento realizado por (SILVA, 2019) tem duração média de 18 horas, além desse destaque em outros trabalhos, identificamos que cada treinamento poderá ter seu tempo modificado de acordo com as tarefas escolhidas e a ordem do currículo. O tipo de recompensa influencia diretamente no tempo que o agente irá levar para aprender uma tarefa, recompensas esparsas (são recompensas que demoram a ser obtidas, requerendo muitas interações), permanecem desafiadoras de resolver com a aplicação direta desses algoritmos. Um dos motivos é que o número de amostras necessárias para resolver uma tarefa com exploração aleatória aumenta exponencialmente com o número de etapas para obter uma recompensa (LANGFORD, 2010).

3.2 *Teacher-Student*

No *Teacher-Student Curriculum Learning - TSCL* (MATIISEN et al., 2019), o aluno/agente (*Student*) é o modelo que está sendo treinado. O professor (*Teacher*) monitora o progresso do treinamento do aluno e determina as tarefas nas quais o aluno deve treinar em cada etapa do treinamento, a fim de maximizar a progressão do aluno ao longo do currículo, a configuração é descrita na Figura 4. O aluno pode ser qualquer modelo de aprendizado de máquina. O próprio professor está aprendendo sobre o aluno à medida que atribui tarefas, tudo como parte de uma única sessão de treinamento (MATIISEN et al., 2019).

(MATIISEN et al., 2019) formalizou uma estrutura para o Teacher-Student com aprendizado curricular, do ponto de vista do professor, como um processo de decisão Markov parcialmente observável (POMDP). Propondo uma família de algoritmos que utilizam como base a noção de progresso de aprendizado do aluno.

Figura 4 – Configuração *Teacher-Student*



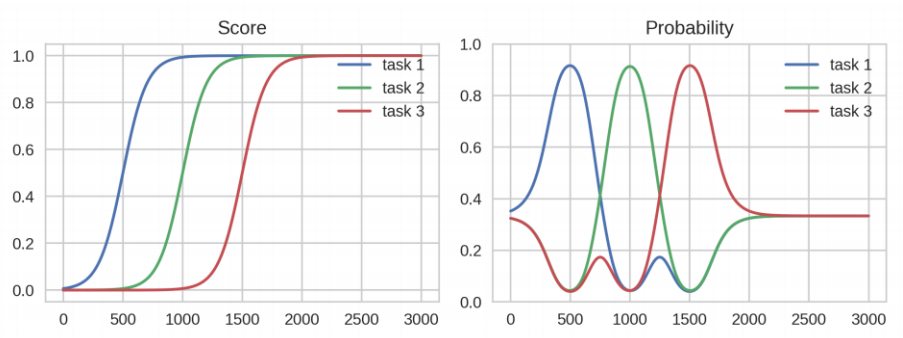
Fonte: *Teacher-Student Curriculum Learning* (MATIISEN et al., 2019).

POMDPs são normalmente resolvidos usando algoritmos de aprendizagem por reforço. No entanto, eles requerem muitos episódios de treinamento, enquanto utilizamos o objetivo de escolher uma tarefa para o Aluno usando heurísticas mais simples.

A sequência adotada por (MATIISEN et al., 2019) pode ser demonstrada na Figura 5. Representando no eixo x a quantidade de passos, no lado esquerdo, as recompensas de diferentes tarefas que melhoram com o tempo e, lado direito, a probabilidade de escolha de uma tarefa que depende da inclinação da curva de aprendizado. Exibindo o progresso de treinamento ideal em um ambiente de aprendizado de currículo, com as seguintes etapas:

1. Inicialmente o Professor não tem conhecimento então ele coleta as informações do aluno, conforme o aluno vai treinando em cada uma das tarefas;
2. Conforme o aluno realizar as tarefas e começa a fazer progresso o professor adiciona mais probabilidade para execução dessa tarefa;
3. Dominando a tarefa atual, a curva de aprendizado fica mais constante e o Professor começa a testar a tarefa com menos frequência. O aluno inicia também progresso em outra tarefa, de modo que o Professor tenha mais amostras;
4. Esse processo segue até que o Aluno domine todas as tarefas (iteração 2000). Com a curva de aprendizagem convergindo no final, o Professor retorna a à amostragem uniforme das tarefas.

Figura 5 – Aprendizagem de Currículo ideal



Fonte: *Teacher-Student Curriculum Learning* (MATIISEN et al., 2019).

O aluno pode ter um desempenho pior que o anterior na mesma tarefa, isto indica que o mesmo desaprendeu aquela tarefa específica. Se a mudança nas pontuações for negativa, esta tarefa deve ser praticada mais. Para tornar esse aprendizado eficiente, dado um currículo e o estado de aprendizado atual de um agente, precisamos descobrir quais são as próximas tarefas mais adequadas para treinar o agente, é esse problema que os algoritmos de *Teacher-Student* (que executam no professor) visam resolver. Os algoritmos TS assumem que as próximas tarefas boas são aquelas nas quais o agente está fazendo o progresso mais rápido, ou teve queda de desempenho maior (está desaprendendo).

(MATIISEN et al., 2019) apresenta quatro algoritmos, são eles: *Online*, *Naive*, *Window* e *Sampling*. Todos partem da ideia de que a atenção deve ser dada pelo valor absoluto de uma estimativa do progresso da aprendizagem ao longo das tarefas, ou seja, $a_c(t) := |\beta(t)|$ onde $\beta_c(t)$ é uma estimativa do progresso de aprendizagem do agente na tarefa c.

O aluno corresponde a um algoritmo de aprendizagem por reforço convencional, por exemplo o PPO sendo aplicado para aprender uma tarefa. Mas, como no currículo learning, existe um conjunto de tarefas (que podem ser sub-tarefas ou simplificações de uma tarefa principal) e o aluno vai ser treinado com cada uma delas. A grande diferença é que um algoritmo *Teacher* vai escolher dinamicamente qual tarefa aplicar. E uma tarefa pode ser reapresentada.

3.3 Modelagem de Recompensas

As recompensas modeladas são aplicadas como estratégia para resolver as dificuldades ocorridas pela atribuição de créditos e recompensas esparsas que retardam significativamente o aprendizado de um agente de Aprendizagem por Reforço (MAROM; ROSMAN, 2018).

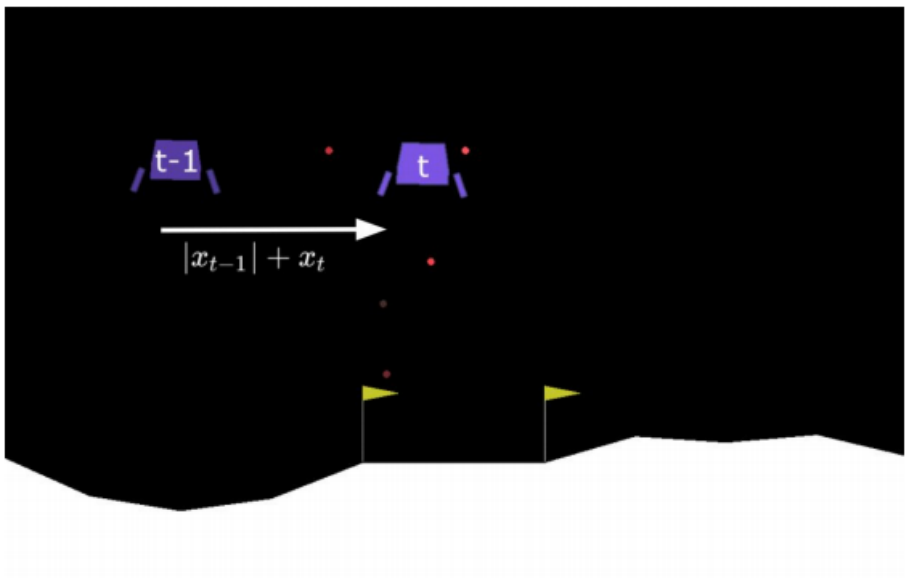
As recompensas são atribuídas aos agentes de aprendizagem por reforço unicamente por cada ação que o mesmo realiza em seu ambiente, aprendendo como agir e se comportar neste ambiente. Uma recompensa modelada é um mecanismo de exploração do agente que acrescenta uma recompensa adicional à recompensa natural obtida. Uma recompensa modelada pode ser descrita da seguinte forma:

$$R_0 = R + F,$$

onde R é a recompensa original do ambiente, F é a recompensa a ser adicionada a recompensa do agente e R_0 é a recompensa modelada (MANNION et al., 2017).

Um exemplo de modelagem pode ser visualizado na figura 6, na qual o agente, neste caso o módulo de pouso, recebe uma recompensa positiva ou negativa, quando se aproxima ou se afasta da posição $x = 0$ respectivamente. A posição $x = 0$ é onde está a área de pouso. O sistema de recompensas padrão do *Lunar Lander* não atribui nenhuma recompensa positiva ou negativa para o agente no ar, sinalizando que a sua coordenada x está longe do ponto de aterrizagem.

Figura 6 – Exemplo de um ambiente com modelagem de recompensa



Fonte: Um currículo de aprendizagem por reforço para o Lunar Lander (ALBUQUERQUE, 2021).

4 Uma Nova Combinação de Técnicas

Neste trabalho nos baseamos na pesquisa proposta por (MATIISEN et al., 2019) com objetivo implementar a abordagem de *Teacher-Student Curriculum Learning*, aplicando de uma maneira nova: em diferentes versões de um mesmo ambiente, modificando o modelo de recompensa. Não encontramos na literatura estudos sobre aplicação de *Teacher-Student* em sub-tarefas ou sub-modelos de um mesmo ambiente com recompensas modeladas. Vamos usar o ambiente *Lunar Lander* como estudo de caso.

Com base na pesquisas relacionadas ao *Teacher-Student*, observamos os resultados obtidos pelos algoritmos propostos. Decidimos utilizar os algoritmos do *Teacher-Student* que deram bons resultados nos estudos de (MATIISEN et al., 2019), sendo eles o *Window* e o *Online*, além de adotar como critério de decisão o método ε-*greedy*, que é um método simples para equilibrar exploração, ε se refere à probabilidade de escolher explorar.

4.1 Algoritmos

Adaptamos o algoritmo proposto por (MATIISEN et al., 2019), onde a principal alteração é o novo passo de avaliação que realiza o treinamento do aluno no ambiente original, alvo principal do aprendizado.

O algoritmo *Online* (algoritmo 1) é inspirado no algoritmo básico de bandido não-estacionário (*non-stationary bandit algorithm*) (SUTTON; BARTO, 2018), que é um problema de aprendizagem por reforço no qual um agente tem que fazer n escolhas entre k opções diferentes. Cada opção oferece uma recompensa (possivelmente) diferente de uma distribuição que muda ao longo do tempo (ou seja, não é estacionária). O problema fica mais difícil porque as observações anteriores são de pouca utilidade.

Neste algoritmo utilizamos a média móvel exponencialmente ponderada para estimar o retorno esperado Q das tarefas (ambientes):

$$Q_{t+1}(a_t) = \alpha r_t + (1 - \alpha)Q_t(a_t),$$

onde α é a taxa de aprendizagem, a é a ação realizada pelo agente e r a recompensa. (MATIISEN et al., 2019).

A recompensa r é uma medida do quanto o agente está aprendendo ou desaprendendo sobre ambiente na tarefa atual. O valor de r é dado pela diferença entre os

desempenhos obtidos pelo agente nesta mesma tarefa na seleção atual com a seleção anterior.

Algorithm 1 Algoritmo *Online*

1: **procedure** TS_Online($\alpha, \varepsilon, P, steps$)
2: Inicializar o algoritmo de aprendizagem (ALUNO)
3: Inicializar o retorno esperado $Q(a)$ = 0 para todos os N ambientes
4: $num\_updates$ = $steps \div P$
5: **for** *t = 1,...,*$num\_updates$ **do**
6: Escolher o ambiente a_t baseado em $|Q(a)|$ usando o método $\varepsilon - greedy$
7: Treinar o ALUNO usando o ambiente escolhido por P passos
8: Avaliar o aluno no ambiente original e guardar o desempenho médio $r_t = x_t^{(a_t)} - x_{t'}^{(a_t)}$
9: Atualizar retorno esperado $Q(a_t) = \alpha r_t + (1 - \alpha)Q(a_t)$

Inicialmente definimos os parâmetros para inicializar o algoritmo com a organização das tarefas que serão aprendidas pelo aluno, escolhemos os valores da exploração ε, a taxa de aprendizagem α. A execução do algoritmo é realizada por um número de *updates*, esse número é a quantidade de vezes que o professor vai escolher uma tarefa para o aluno. Esse valor é definido pela quantidade de passos (*steps*) que escolhemos para o treinamento completo do aluno. O valor é calculado através da divisão da quantidade de passos de treinamentos dividido pelo valor do total de passos de atualização no algoritmo de aprendizado do aluno.

A primeira iteração no algoritmo *Teacher-Student* inicia com a escolha da tarefa que o aluno irá aprender, escolha realizada pelo ε-*greedy*, que consiste em escolher uma tarefa (por exemplo, uma versão do *Lunar Lander*) com probabilidade ε ou, com probabilidade (1-ε), escolher o ambiente a de maior valor $|Q(a)|$. Em seguida, realiza-se o treinamento no ambiente escolhido, avalia o aluno no ambiente original (este passo poderia ser feito direto com os resultados obtidos durante o treinamento do passo 6, mas implementamos como um passo separado), e a recompensa r obtida é salva e atualizada no retorno esperado $Q_{(a)}$.

O algoritmo *Window* (algoritmo 2) mantém um buffer do tipo *FIFO* (First-in First-Out) com os últimos K desempenhos médios (buffer D) e os registros dos seus respectivos passos de tempo (buffer E). É utilizada a regressão linear usando D e E para tentar prever a recompensa r_t com a inclinação da curva para cada ambiente, e o coeficiente de regressão é utilizado como recompensa no algoritmo *non-stationary bandit algorithm* citado anteriormente.

Implementamos os dois algoritmos *Teacher-Student* citados, que são os algoritmos que decidem, dado um currículo e o estado de aprendizagem do aluno, quais tarefas que o aluno deverá treinar em seguida. Os algoritmos possuem como parâmetros, a lista de tarefas a serem executadas pelo aluno, o valor *épsilon* (ε) para decisão de distribuição e o valor *alpha* (α) para ajuste das recompensas, esses parâmetros são

comuns para os dois algoritmos e o valor P que é quantidade de passos de treinamento por rodada, para o algoritmo *Window* temos um parâmetro a mais que é o *K*, valor referente a quantidade das últimas recompensas guardadas no buffer.

Algorithm 2 Algoritmo *Window*

procedure TS_Window($\alpha, \varepsilon, K, P, steps$)
    Inicializar o algoritmo de aprendizagem (ALUNO)
    Inicializar os buffers da fila $D(a)$ e $E(a)$ com tamanho K para todos os N ambientes
    Inicializar o retorno esperado $Q(a)$ = 0 para todas as N ambientes
    $num\_updates$ = $steps \div P$
    for *t = 1,...,*$num\_updates$ **do**
        Escolher o ambiente a_t baseado em $|Q(a)|$ usando o método $\varepsilon - greedy$
        Treinar o ALUNO usando o ambiente escolhido por P passos
        Avaliar o aluno no ambiente original e guardar o desempenho médio $o_t = x_t^{(a_t)}$
        Armazene a recompensa em $D(a_t)$ e os passos de tempo em $E(a_t)$
        Aplicar regressão linear p/ prever $D(a_t)$ a partir de $E(a_t)$ e usar o coeficiente como r_t
        Atualizar retorno esperado $Q(a_t) = \alpha r_t + (1 - \alpha)Q(a_t)$

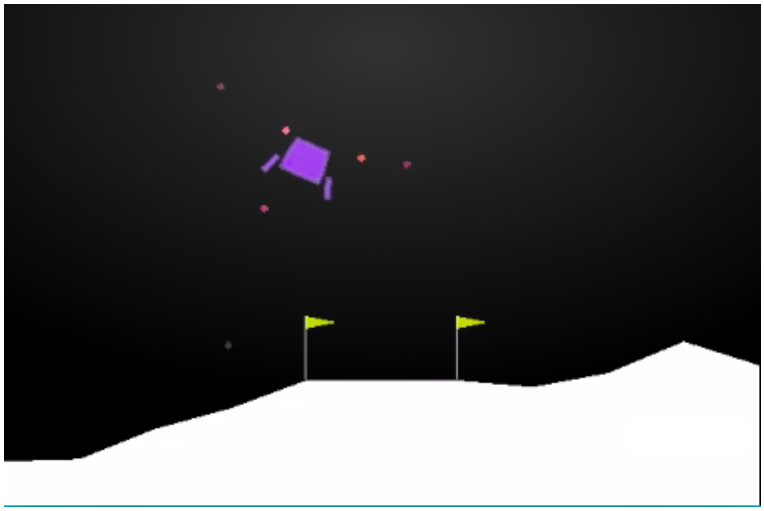
4.2 Tarefas Adotadas

Definimos como algoritmo de aprendizagem do aluno o PPO2, algoritmo de aprendizagem por reforço da biblioteca *Stable Baselines*. A biblioteca é um conjunto de implementações aprimoradas de algoritmos de aprendizagem por reforço baseados na biblioteca *OpenAI Baselines*, que são implementações de alta qualidade de algoritmos para melhorar a sua utilização e replicação.

O ambiente utilizado na proposta foi o *LunarLander-v2* (figura 7) da *Open AI Gym*. De acordo com (BROCKMAN et al., 2016), a nave é um módulo de pouso que possui uma plataforma de pouso. Originalmente o ambiente possui recompensa por mover-se do topo da tela para a área de pouso e parar com a velocidade zero é obtido recompensa entre 100 e 140 pontos. Se o módulo de pouso se afastar da plataforma de pouso, ele perde a recompensa conquistada. O episódio termina se o módulo de pouso bater ou parar, recebendo -100 ou $+100$ pontos adicionais. Cada contato de solo da perna é $+10$. O motor principal ao ser acionado gera $-0,3$ pontos por quadro. A recompensa alvo para solução são 200 pontos.

Além do ambiente original, que é o alvo principal do aprendizado, foi adotado modelos de recompensa diferentes criados para esse mesmo ambiente, esses modelos foram propostos por (ALBUQUERQUE, 2021). O ambiente possui três diferentes alterações que atribui recompensas por ações/tarefas que ajude o aprendizado do agente, as recompensas são baseadas nas movimentações que o agente realiza no ambiente. Os modelos de recompensas são esses:

Figura 7 – Ambiente *LunarLander-v2*



Fonte: OpenAI Gym (BROCKMAN et al., 2016)

1. *Identity*, representa o sistema de recompensas original.
2. *CenterLander*, que acrescenta recompensas para incentivar a centralização da nave em relação a área de pouso. Ele atribui recompensas positivas por mover a nave em direção ao centro (na horizontal);
3. *Downlander*, que acrescenta recompensas que incentivam a descida da nave até o chão, não necessariamente a área de pouso. Quanto mais próximo ao chão o agente recebe recompensas positivas, se ele se afastar recebe negativas;
4. *DistanceLander*, .que acrescenta um sistema de recompensas baseado na distância em linha reta da posição da nave até a área de pouso. Quanto mais próximo a área de pouso, recompensas positivas. Quanto mais distante, recompensas negativas serão reportadas.

As quatro diferentes percepções foram adotadas para reportar com mais frequência recompensas para o agente. A área de pouso do *Lunar Lander* corresponde à coordenada (0, 0). O objetivo principal da nave no *Lunar Lander* é pousar na área designada e para que isso ocorra, intuitivamente precisa-se que a nave seja encorajada a descer. Os vários modelos incentivam comportamentos úteis para um pouso com sucesso (descer, centralizar e pousar).

Ao aplicar o *Teacher-Student* com a modelagem de recompensa, criamos uma nova técnica para o aprendizado do agente, buscando entender o comportamento do agente e se a combinação poderá fazer com que o aprendizado seja mais rápido.

5 Experimentos e Resultados

5.1 Experimentos

Realizamos a comparação das três abordagens que possuem a modelagem de recompensas entre si, buscamos verificar o aprendizado individualmente de cada uma das tarefas. Posteriormente fizemos a comparação do agente no ambiente original.

O algoritmo que utilizamos para o treinamento do aluno foi o PPO2, os seus parâmetros foram definidos em um conjunto de experimentos para otimizar hiper-parâmetros, realizada como parte de um trabalho anterior, desenvolvido como iniciação científica. Os parâmetros para o agente são descritos conforme tabela 1.

Tabela 1 – Parâmetros do PPO2 (algoritmo usado pelo *Student*)

Parâmetro	**Valor**
n_steps	128
qtd_de_camadas	2
camada_1	128
camada_2	512
learning_rate	0.001
act_fun	$relu$

Fonte: Autor.

- **n_steps**: números de passos para serem executadas no ambiente, antes de cada atualização dos pesos da rede neural da política;
- **camadas**: quantidade de neurônios nas camadas intermediárias da rede da política;
- **learning_rate**: taxa de aprendizagem do agente;

Para inicialização das modelagens de recompensa, cada ambiente possui um ou mais parâmetros de entrada definidos por (ALBUQUERQUE, 2021), que influenciam diretamente no modo como será calculada a recompensa. Utilizamos os parâmetros descritos na tabela 2.

Tabela 2 – Parâmetros *Lunar Lander* com recompensas modeladas

Modelagem	**Parâmetro**	**Valor**
Identity	N/A	N/A
CenterLander	fator_delta	0.15
DownLander	fator_checkpoint qtd_checkpoint	0.15 10
DistanceLander	fator_delta	0.15

Fonte: (ALBUQUERQUE, 2021).

Para o algoritmo *Teacher-Student Online* definimos os parâmetros baseado nos valores utilizados por (MATIISEN et al., 2019) como melhores resultados, conforme a tabela 3. realizamos também uma variação para mais e para menos, para realizar novos testes, no entanto os valores padrões (0.3 e 0.5) obtiveram resultados mais satisfatórios.

Tabela 3 – Parâmetros *Teacher-Student*

N	ε	α	P	$steps$
4	0.3 ± 0.2	0.5 ± 0.2	128	600000

Fonte: Autor.

Onde N é a quantidade de ambientes, ε é a taxa de exploração, α é a taxa de aprendizagem, P é a quantidade de passos para avaliação *Teacher* e $steps$ a quantidade de passos total para treinamento do *Student*.

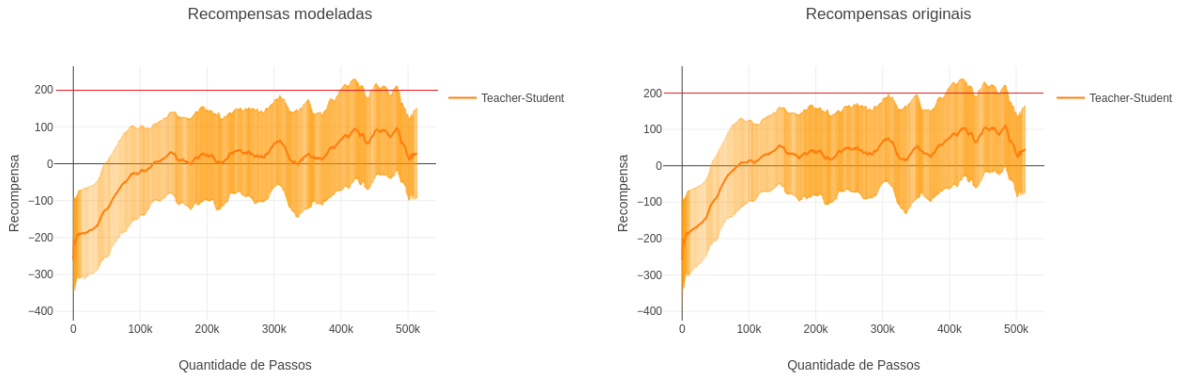
Foram realizadas cinco execuções de treinamento com o algoritmo PPO para cada uma das seguintes abordagens:

1. Treinamento apenas no ambiente original *Lunar Lander*
2. Treinamento apenas no ambiente modelado *Center Lander*
3. Treinamento com o *Teacher-Student* nos dois ambientes (original *Lunar Lander* e *Center Lander* modelado)

5.2 Resultados

Como adaptamos a abordagem *Teacher-Student* com a modelagem de recompensa, consideramos o resultado do treinamento por completo, sem a separação de ambientes nos resultados. Podemos observar o resultado no gráfico da figura 8 que contém a média (ao longo do tempo de treinamento) com valores de recompensa modelada (a recompensa retornada por cada versão do ambiente) e recompensa original (a recompensa do ambiente original, desconsiderando as recompensas acrescentadas em cada versão). Vemos que existe pouca diferença na forma do gráfico. Por isso, nossa comparação focou no gráfico de recompensas originais, para melhorar comparação com as outras abordagens consideradas (1 e 2).

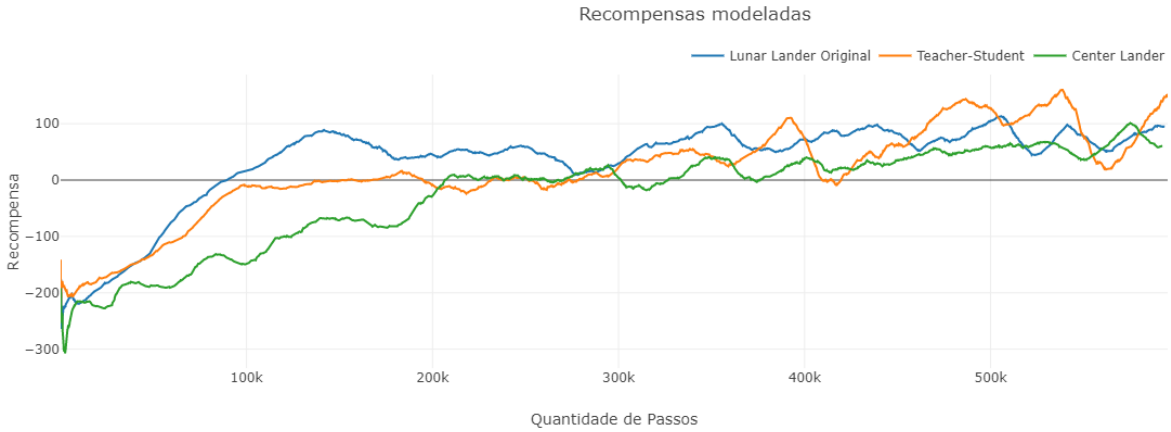
Figura 8 – Média dos treinamentos com algoritmo *Teacher-Student*



Fonte: Autor

Comparamos o treinamento realizado pelo *Teacher-Student* com o treinamento de um agente no ambiente original e na versão modelada *CenterLander*, os resultados são descritos no gráfico da figura 9.

Figura 9 – Média dos treinamentos de um agente em cada ambiente



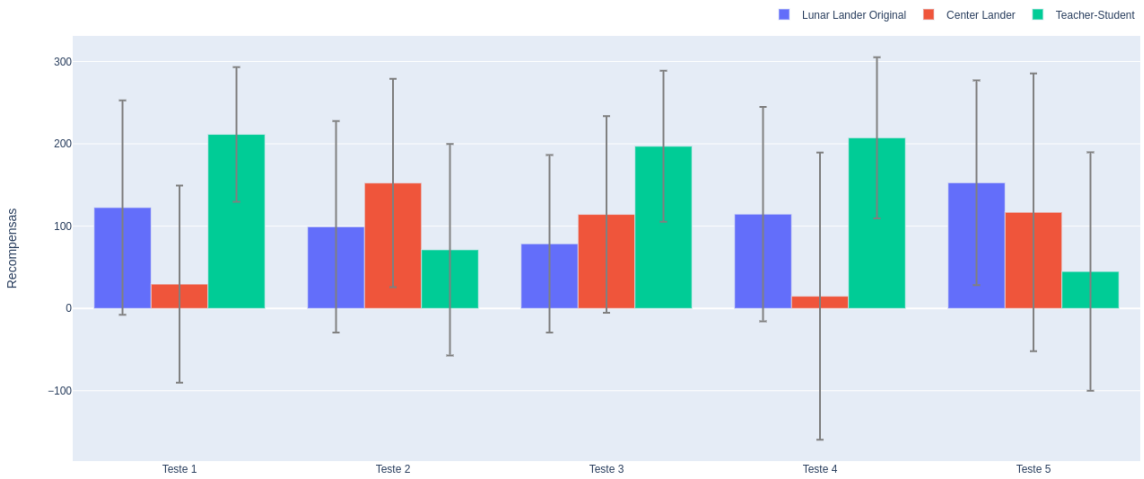
Fonte: Autor

Após a realização de 5 treinamentos, o agente treinado foi adicionado ao ambiente original e executado por 20 episódios. Verificamos o comportamento do agente e observamos as recompensas obtida, realizando uma comparação entre os testes.

Nossa abordagem (*Teacher-Student + Reward Shaping*) *acelera o aprendizado*, considerando quantas vezes cada algoritmo conseguiu treinar com sucesso, ou seja, atingindo o desempenho alvo do ambiente (200 pontos), executando 5 treinamentos de duração relativamente curta (600 mil passos).

Conseguimos visualizar essa informação no gráfico da figura 10, onde das 5 execuções com a nossa abordagem, 2 delas fizeram o algoritmo PPO atingir o desempenho de 200 e 1 execução ficando bem próximo ao valor alvo, obtendo recompensa média de 196 pontos, já nas outras abordagens, em nenhuma delas o agente PPO chegou a aprender com sucesso a tarefa.

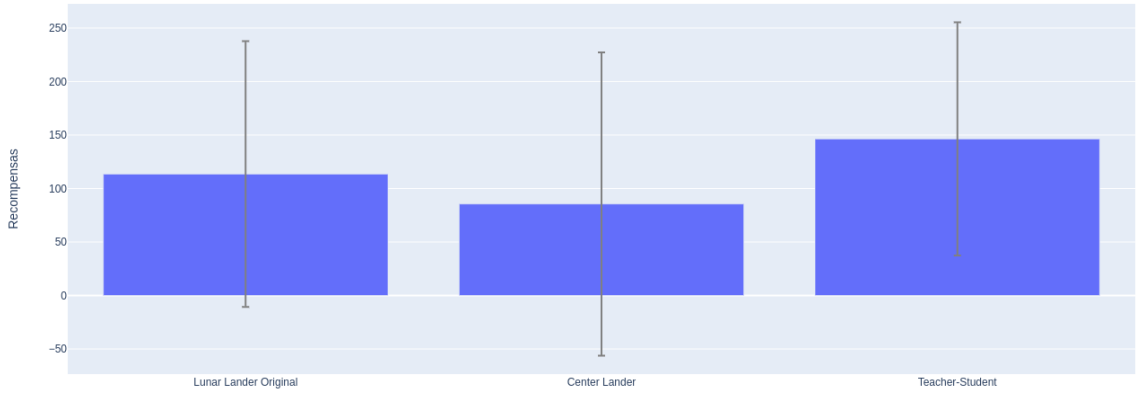
Figura 10 – Média por proposta das execuções do modelo treinado no ambiente original



Fonte: Autor

Avaliando a qualidade nos treinamentos com duração de 600 mil passos, nossa abordagem consegue obter *uma melhor qualidade nas recompensas obtidas*, esse resultado é sobreposto com o resultado reportado acima, porém avaliamos a qualidade do agente independentemente de atingir o valor alvo (200 pontos) do ambiente. O resultado obtidos pelo *Teacher-Student* foi de 147 pontos de média de recompensas, a avaliação pode ser vista no gráfico de barras individual (figura 11), com a média geral de cada abordagem.

Figura 11 – Média geral das execuções do modelo treinado no ambiente original



Fonte: Autor

6 Conclusão e Trabalhos Futuros

O desenvolvimento do presente estudo fez uma análise de como a combinação das técnicas pode melhorar o aprendizado de um agente inteligente. Atingindo médias melhores que as outras abordagens nos treinamentos e nos testes posteriores ao treinamento com o agente já treinado, conseguindo em pouco tempo de treinamento atingir em 2 de 5 propostas o desempenho necessário para solucionar o problema do ambiente Lunar Lander.

Os resultados se apresentam de forma promissoras, tendo em vista que contribuição proposta com a combinação da técnica *Teacher-Student* e *Reward Shaping* conseguem aprender com uma melhor qualidade ao observar a média geral de todas as execuções realizadas após o treinamento do agente.

Concluímos que este estudo foi uma investigação inicial sobre a combinação das técnicas, podendo ser investigada mais a fundo posteriormente, adotando mais tempo de treinamento, novos ajustes de parâmetros, mais testes em busca de novos resultados e comparações com outras abordagens.

6.1 Considerações Finais

O estudo contou com a implementação do algoritmo *Window*, porém não foram realizados testes mais detalhados com o mesmo devido a questão do tempo de treinamento dos algoritmos. Podendo ser um trabalho posterior a este estudo.

O desenvolvimento de ideias que possam otimizar treinamentos de agentes, é cada vez mais necessário com o objetivo de torná-las fáceis de serem utilizadas ou implementadas. Nesse sentido, a utilização de bibliotecas, como por exemplo o Optuna (AKIBA et al., 2019), que visem o aperfeiçoamento dos parâmetros de entrada dos algoritmos, poderá permitir o aprendizado mais aprimorado. Além disso, diminui a quantidade de implementações e de execuções de algoritmos para solucionar mais e novos desafios.

6.2 Trabalhos Futuros

- Utilizar o algoritmo *Window* para testes e comparação com a abordagem *Online*.
- Aplicar a combinação das técnicas aqui proposta em outros ambientes, por exemplo o *Google Research Football*.
- Aprimorar os parâmetros de entrada dos algoritmos e modelos de recompensa através de hiperparâmetros.
- Fazer experimentos mais profundos.
- Testar novas variações dessa combinação de TS com modelagem das recompensas.

Referências

AKIBA, T. et al. Optuna: A next-generation hyperparameter optimization framework. In: *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.: s.n.], 2019. Citado na página 34.

ALBUQUERQUE, R. S. *Um currículo de aprendizagem por reforço para recompensas modeladas no Lunar Lander*. Monografia (TCC) — Universidade Federal Rural de Pernambuco, Recife, 2021. Citado 7 vezes nas páginas 5, 6, 15, 24, 27, 29 e 30.

ARGERICH, M. F. How is reinforcement learning used in business? *Medium*, 2020. Citado na página 12.

ARULKUMARAN, K. et al. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017. Citado na página 17.

BELLEMARE, M. G. et al. Unifying count-based exploration and intrinsic motivation. *arXiv preprint arXiv:1606.01868*, 2016. Citado na página 11.

BENGIO, Y. et al. Curriculum learning. In: *Proceedings of the 26th annual international conference on machine learning*. [S.l.: s.n.], 2009. p. 41–48. Citado 2 vezes nas páginas 14 e 20.

BROCKMAN, G. et al. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. Citado 3 vezes nas páginas 19, 27 e 28.

DULAC-ARNOLD, G.; MANKOWITZ, D.; HESTER, T. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019. Citado na página 13.

DVB. *Why do 87% of data science projects never make it into production?* 2019. Disponível em: <https://venturebeat.com/2019/07/19/why-do-87-of-data-science-projects-never-make-it-into-production/>. Acesso em: 20 julho 2021. Citado na página 13.

ELMAN, J. L. Learning and development in neural networks: The importance of starting small. *Cognition*, Elsevier, v. 48, n. 1, p. 71–99, 1993. Citado na página 20.

HILL, A. et al. *Stable Baselines*. [S.l.]: GitHub, 2018. <https://github.com/hill-a/stable-baselines>. Citado na página 19.

KIM OUWELEEN. *AlphaGo and the future of go: interviews with top European players*. 2016. Disponível em: <https://www.eurogofed.org/index.html?id=67>. Acesso em: 18 junho 2021. Citado na página 11.

KORMUSHEV, P.; CALINON, S.; CALDWELL, D. G. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, v. 2, n. 3, p. 122–148, 2013. ISSN 2218-6581. Disponível em: <https://www.mdpi.com/2218-6581/2/3/122>. Citado na página 13.

KRUEGER, K. A.; DAYAN, P. Flexible shaping: How learning in small steps helps. *Cognition*, Elsevier, v. 110, n. 3, p. 380–394, 2009. Citado na página 20.

KURACH, K. et al. Google research football: A novel reinforcement learning environment. *arXiv preprint arXiv:1907.11180*, 2019. Citado 2 vezes nas páginas 20 e 21.

LANGFORD, J. Efficient exploration in reinforcement learning. In: ____. *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010. p. 309–311. ISBN 978-0-387-30164-8. Citado na página 21.

MANNION, P. et al. Policy invariance under reward transformations for multi-objective reinforcement learning. *Neurocomputing*, Elsevier, v. 263, p. 60–73, 2017. Citado na página 24.

MAROM, O.; ROSMAN, B. Belief reward shaping in reinforcement learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2018. v. 32, n. 1. Citado na página 23.

MATIISEN, T. et al. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, IEEE, v. 31, n. 9, p. 3732–3740, 2019. Citado 7 vezes nas páginas 14, 15, 21, 22, 23, 25 e 30.

PUTERMAN, M. L. Markov decision processes. *Handbooks in operations research and management science*, Elsevier, v. 2, p. 331–434, 1990. Citado na página 17.

SCHULMAN, J. et al. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. Citado na página 18.

SHOHAM, Y.; LEYTON-BROWN, K. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. [S.l.]: Cambridge University Press, 2008. Citado na página 12.

SILVA, J. W. P. *Um Currículo de Aprendizado por Reforço para o Cenário "Run to Score with Keeper"do Google Research Football Environment*. 49 f. Monografia (TCC) — Universidade Federal Rural de Pernambuco, Recife, 2019. Citado 2 vezes nas páginas 20 e 21.

SILVER, D. et al. Mastering the game of go with deep neural networks and tree search. *nature*, Nature Publishing Group, v. 529, n. 7587, p. 484–489, 2016. Citado na página 11.

SMITH, T.; SIMMONS, R. Heuristic search value iteration for pomdps. *arXiv preprint arXiv:1207.4166*, 2012. Citado na página 17.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018. Citado 4 vezes nas páginas 12, 16, 17 e 25.

WIKIPEDIA. *Markov decision process*. 2021. Disponível em: <https://en.wikipedia.org/wiki/Markov_decision_process>. Acesso em: 27 junho 2021. Citado na página 18.