Insper

Instituto de Ensino e Pesquisa

Curso de Engenharia de Computação

Programa Institucional de Bolsas de Iniciação Científica (PIBIC)

RELATÓRIO PARCIAL

ESTUDO COMPARATIVO DE MODELOS DE OTIMIZAÇÃO DE AGENTES AUTÔNOMOS BASEADOS EM APRENDIZADO POR REFORÇO

JOÃO GABRIEL VALENTIM ROCHA

ORIENTADOR: Prof. Dr. Fábio José Ayres

São Paulo

2022, Fevereiro

João Gabriel Valentim Rocha

RELATÓRIO PARCIAL

Estudo comparativo de modelos de otimização de agentes autônomos baseados em aprendizado por reforço

Relatório de Projeto de Iniciação Científica para cumprir com os requisitos estabelecidos no Edital do Programa Institucional de Bolsas de Iniciação Científica do Insper.

São Paulo

2022, Fevereiro

RESUMO

O estudo comparativo de modelos de optimização de agente autônomos tem influência direta nas circunstâncias tecnológicas atuais. O crescimento exponencial do número de aplicações da área de Reinforcement Learning tem trazido a necessidade de estudos comparativos, haja vista a importância de selecionar e utilizar algoritmos cada vez mais atuais, mais rápidos e mais eficientes para aquela tarefa. Logo, nosso objetivo é trazer um estudo detalhado a respeito do desempenho, sob as métricas adequadas, que os principais algoritmos de optimização de agentes autônomos têm e compará-los. Nesse sentido, utilizamos um ambiente de simulação para entender cada algoritmo na essência enquanto fazemos cada estudo comparativo. Sob essa perspectiva, é de suma importância tomar como base os estudos mais recentes de algoritmos de optimização que são utilizados hoje. A utilização das métricas para avaliar um algoritmo vai variar de algoritmo para algoritmo, caberá a nós escolher e utilizar as mais adequadas para aquele tipo de processo de aprendizagem do agente. Além disso, alguns métodos podem ser inseridos junto ao algoritmo, com o intuito de melhorar a performance da aprendizagem do agente. Nesse sentido, utilizamos os métodos de Curriculum Learning e Transfer Learning, que consiste na ideia de transferência de aprendizado por meio de um curriculum. Em outras palavras, é possível que um agente possa aprender a realizar uma tarefa complexa pela secção dessa tarefa em tarefas menores que podem transferir o aprendizado de uma tarefa (menos complexa) para a próxima (mais complexa). Dessa forma, é possível que a curva de aprendizagem (ou a de recompensa acumulativa) venha a convergir em um tempo menor e com menos esforço. O processo de estabelecer um comparativo entre modelos de algoritmos de optimização de agente autônomos é, portanto, crucial para o embasamento necessário que permite que o número de aplicações e solução de problemas cresçam cada vez mais, sobretudo, aquelas que envolvem as técnicas de Reinforcement Learning.

Palavras-chaves: Reinforcement Learning. Genetic Algorithms. Neural networks policies. Steering Behaviors. Policy gradients, deep Q-networks.

SUMÁRIO

1 INTRODUÇÃO

Grande parte das técnicas utilizadas para traçar a caminhada de robôs ou carros que dirigem sem a necessidade de motorista são pautadas na otimização de agente autônomos. A área do conhecimento denominada "Otimização de agentes autônomos" (em inglês: Autonomous Agent Optimization) refere-se ao conjunto de técnicas utilizadas para aprimorar uma atividade feira por um agente autônomo, inserido em um ambiente, por meio de ferramentas computacionais, com o objetivo de buscar a solução ótima de uma tarefa determinística ou não (GéRON, 2019). Exemplos do uso de otimização de agentes autônomos incluem a otimização da caminhada de um robô, a trajetória de carros que dirigem sem a necessidade de um motorista, sistemas que podem aprender a jogar jogos eletrônicos (MNIH et al., 2013), otimização de um termostato para economia de energia e negociações financeiras automáticas (GéRON, 2019).

Nesse sentido, o mundo está cheio de situações em que um agente inteligente pode ser útil. Um agente é o responsável por interagir com o ambiente e recebe recompensas para realizar alguma tarefa. Nesse aspecto, se tomarmos como exemplo um sistema em que um drone precisa fazer uma entrega para uma empresa de delivery, o drone seria o agente, o ambiente seria o cenário em que ele se encontra e a recompensa seria ele levar a entrega para o endereço designado, no menor tempo possível. Dessa forma, é possível visualizar que as aplicações do Reinforcement Learning podem ser ampliadas para âmbitos diversos, como o exemplo da entrega do drone e o exemplo de sistemas que aprendem a jogar jogos eletrônicos, citados anteriormente.

Nesse contexto, é preciso definir o que se chama de Policy, que é a política de tomadas de decisão, em alguns casos, é possível se referir a policy com o sendo o agente. Para desenvolver a política de tomada de ações do agente usamos técnicas de aprendizado conhecidas como aprendizado por reforço. Dessa forma, podemos estabelecer mais a frente o problema central do Reinforcement Learning, que é encontrar a policy optima, ou seja, a política de tomada de ações que traz a maior recompensa para o agente.

2 REINFORCEMENT LEARNING

Aprendizado por reforço (*Reinforcement Learning*) refere-se ao conjunto de técnicas para o treinamento de agentes autônomos através de incentivos que chamamos de **recompensa**. Um agente pode ser qualquer entidade que se encontra em um determinado ambiente, com o intuito de realizar alguma tarefa. Nesse sentido, o ambiente interage com o agente e isso caracterizar a essência do sistema que é o foco do estudo do *Reinforcement Learning*, tal como o gráfico a seguir.

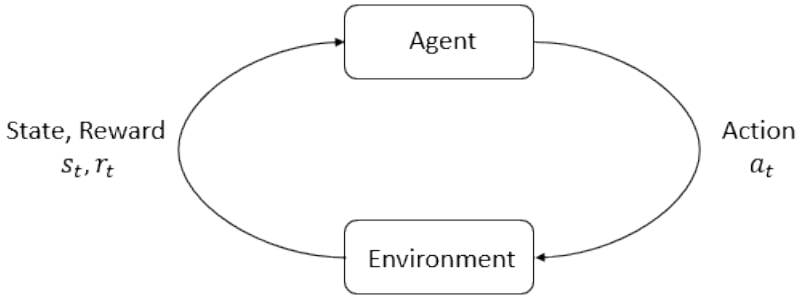

Figura 1 – Interação agente-ambiente.

Diante disso, é possível descrever o comportamento e o andamento do processo com a descrição de algumas variáveis que são pertinentes à condição do Agente: a ação a_t a ser tomada no tempo t, o estado s_t do tempo t, o próximo estado s_{t+1}, alcançado após a tomada da ação a_t e a recompensa r_t devido a troca de estado.

A exemplo disso, podemos pensar num robô aspirador de pó que traça trajetórias com o intuito de limpar uma sala. Nesse contexto, o agente é o robô e o ambiente é sala. Nosso sistema buscará resolver o problema central do *Reinforcement Learning*, que é maximizar a recompensa acumulada, que nesse caso, é fazer com que o robô limpe a sala no menor tempo possível, traçando trajetórias que são optimizadas para que isso ocorra.

Dessa forma, vamos definir alguns aspectos importantes e necessários para que possamos descrever e entender os sistemas de *Reinforcement Learning* de uma maneira mais consistente e estruturada, sobretudo, fazendo uso das ferramentas matemáticas para descrever como funciona alguns conceitos e abordagens.

2.1 Policy

A policy é a regra utilizada pelo agente para decidir quais ações tomar, que pode ser determinística ou estocástica. Aqui trabalharemos com policies estocásticas. As policies estocásticas são usualmente denotadas por π:

$$a_t \sim \pi(\cdot|s_t).$$

É comum que a policy seja mencionada de forma intercambiável com o agente, pelo fato da policy poder ser interpretada como o cérebro do agente. Em deep RL, lidamos com policies parametrizadas, comumente denotamos os parâmetros de tal policy por θ ou ϕ e, em seguida escrevemos isso como um subscrito no símbolo da policy:

$$a_t \sim \pi_\theta(\cdot|s_t).$$

2.2 Trajetória

Uma trajetória é uma sequência de estados e ações de um agente que atua em um determinado ambiente, para realizar uma determinada tarefa.

$$\tau = (s_0, a_0, s_1, a_1, ...).$$

O primeiro estado é aleatoriamente inserido seguindo uma distribuição de estado inicial denotada por ρ_0 :

$$s_0 \sim \rho_0(\cdot).$$

Além disso, para o estado de transição, ou seja, o que ocorre entre um estado s_t e o estado s_{t+1}:

$$s_{t+1} \sim P(\cdot|s_t, a_t).$$

2.3 Recompensa e Retorno

A função de recompensa R é criticamente importante para o Reinforcement Learning. Isso depende do estado atual (s_t), da ação tomada (a_t), e o próximo estado (s_{t+1}):

$$r_t = R(s_t, a_t, s_{t+1})$$

Em conceitos mais palpáveis, a função de recompensa nos traz a influência da transição entre os estados (tomada a ação a_t) de forma quantitativa. Sobretudo, isso tem forte relação com a tarefa que deve ser realizada pelo agente, naquele ambiente. Nesse sentido, vale ressaltar que existe alguns tipos de cálculo do retorno.

Um tipo de retorno é o finite-horizon undiscounted return, que é a soma das recompensas obtidas (fixas) para cada passo:

$$R(\tau) = \sum_{t=0}^{T} r_t.$$

Um outro tipo de retorno é o infinite-horizon discounted return, que é a soma das recompensas obtidas para cada passo, multiplicado por um fator de desconto $\gamma \in (0, 1)$:

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t.$$

Note que $\gamma \in (0, 1)$ é muito conveniente pois, além de garantir a convergência da soma, demonstra que a recompensa maior será atribuida para ação tomada no menor tempo, o que faz sentido para o treinamento do agente.

2.4 *Reinforcement Learning* (Aprendizado por Reforço)

Como vimos anteriormente nas definições gerais do RL, estamos abordando uma área de Machine Learning que tem a finalidade de determinar quais ações um agente deve tomar em ambiente para tornar sua recompensa ou retorno o maior possível. Nós podemos formalizar essa interação do agente com o ambiente (NARVEKAR et al., 2020; NARVEKAR, 2017), (podemos chamar de tarefa ou task) utilizando um Processo de Decisão de Markov (MDP, Markov Decision Process), vamos considerar o caso sem fator de desconto (γ):

- **Definição 1**: Um MDP M é uma 6-tupla $(S, A, p, r, \Delta s_0, S_f)$, onde S é o conjunto de estados, A é o conjunto de ações, $p(s^{'}|s, a)$ é um função de transição que retorna a probabilidade de o agente passar para o estado $s^{'}$, dado que ele está no estado s e tomou a ação a, e $r(s, a, s^{'})$ é a função de recompensa para transição de estado. Além disso, Δs_0 é a distribuição inicial de estados e S_f é o conjunto de estados terminais.

Além da definição 1, lembremos que o agente toma ações mediante uma policy $\pi(a|s)$. O problema central do RL é o agente encontrar uma policy optima π^* que maximiza o retorno esperado G_t (soma acumulada de recompensas R) até o episódio final de T passos.

Existem três principais classes e métodos para aprender π^*, aproximação de função de valor, aproximação de busca de policy e actor-critic método. Na **aproximação da função de valor**, o valor $v_\pi(s)$ é representado pelo retorno esperado quando o agente parte do estado s e segue a policy π até o fim do trajeto.

Na **busca da policy**, nós derivamos a função de valor para encontrar uma policy melhor π, até convergir para a policy optima. Usando a função de valor, é preciso utilizar funções de recompensa para a transição de estados de um ambiente. Se o modelo não é conhecido, uma opção é utilizar uma função de ação-valor, $q_\pi(s, a)$, que dá o retorno esperado ao agente tomar uma ação a no estado s e seguir a policy π:

$$q_\pi(s, a) = \sum_{s'} p(s'|s, a)[r(s, a, s') + q_\pi(s', a')] \tag{2.1}$$

Onde a' $\pi(\cdot|s')$. A função de ação-valor pode ser iterativamente melhorada seguindo para uma função de ação-valor optima q_* com métodos on-policy, tal como o Policy Gradient Descent. Logo, uma policy optima pode ser obtida tomando a ação $\underset{a}{\text{argmax}}[q_*(s, a)]$.

2.5 O Problema do Reinforcement Learning

Para qualquer que seja o retorno medido, a meta do RL é selecionar a policy que maximiza o retorno esperado quando o agente atua numa trajetória τ. Para falar sobre retorno esperado, vamos falar sobre a distribuição de probabilidade ao longo da trajetória. Vamos supor que os ambientes de transição e as policies são estocásticos. Nesse caso, queremos calcular $P(\tau|\pi)$:

$$P(\tau|\pi) = P(s_0, a_0, s_1, a_1, ...|\pi)$$

Porém, pela relação das probabilidades condicionais, temos que:

$$P(A, B) = P(A|B) \cdot P(B)$$

Portando, aplicando a relação, temos:

$$P(\tau|\pi) = P(s_0, a_0, s_1, a_1, ...|\pi)$$

$$= P(a_0, s_1, a_1, ...|\pi, s_0) \cdot P(s_0|\pi)$$

$$= P(s_1, a_1, ...|\pi, s_0, a_0) \cdot \underbrace{P(s_0|\pi)}_{\rho_0(s_0)} \cdot \overbrace{P(a_0|\pi, s_0)}^{\pi(a_0|s_0)}$$

$$= \rho_0(s_0) \cdot P(s_1, a_1, ...|\pi, s_0, a_0) \cdot \pi(a_0|s_0)$$

$$= \rho_0(s_0) \cdot P(a_1, s_2, a_2, ...|\pi, s_0, a_0, s_1) \cdot \underbrace{P(s_1|\pi, s_0, a_0)}_{P(s_1|a_0, s_0)} \cdot \pi(a_0|s_0)$$

$$= \rho_0(s_0) \cdot P(s_2, a_2, ...|\pi, s_0, a_0, s_1, a_1) \cdot \overbrace{P(a_1|\pi, s_0, a_0, s_1)}^{\pi(a_1|s_1)} \cdot P(s_1|a_0, s_0) \cdot \pi(a_0|s_0)$$

$$= \rho_0(s_0) \cdot P(s_2, a_2, ...|\pi, s_0, a_0, s_1, a_1) \cdot P(s_1|a_0, s_0) \cdot \pi(a_0|s_0) \cdot \pi(a_1|s_1).$$

$$= \dots$$

Realizando esta operação repetidas vezes, teremos que para o T-ésimo passo a probabilidade da trajetória dado uma policy será:

$$P(\tau|\pi) = \rho_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t), \tag{2.2}$$

Dessa forma, podemos calcular o retorno esperado de um agente ao longo de um conjunto de trajetórias. O retorno esperado denotado por $J(\pi)$ é então:

$$J(\pi) = \int_\tau P(\tau|\pi)R(\tau) = E_{\tau\sim\pi}[R(\tau)], \tag{2.3}$$

Com base nas equações que encontramos e definimos anteriormente, podemos, portanto, definir o problema central em RL, que pode então ser expresso por:

$$\pi^* = \arg\max_\pi J(\pi), \tag{2.4}$$

com π^* sendo a policy optima. Ou seja, a política de tomada de ações que maximiza a recompensa acumulada ao longo de uma trajetória que o agente pode tomar. Se tomarmos como exemplo um sistema que o agente é um drone de entregas, a policy optima π^* vai conduzir o agente a tomar as melhores tomadas de ações para que ele chegue ao destino designado no menor tempo possível e desviando dos possíveis obstáculos do ambiente. Isso se deve, sobretudo, ao treinamento por aprendizado por reforço, que confere estímulos positivos e negativos ao agente, no intuito de tornar o agente mais apto a tomar as decisões necessárias para a tarefa ser feita com o máximo de recompensa acumulada.

3 METODOLOGIA

Para realizar experimentos que são pertinentes para o nosso estudo, utilizamos a plataforma do Unity como ferramenta inicial para simular ambientes e agentes que precisam realizar uma determinada tarefa. Nesse sentido, utilizaremos o toolkit ml-agents da plataforma e utilizaremos o Tensorboard (Tensorflow) para visualizar o comportamento dos experimentos.

3.1 *Unity*

Unity é um motor para criação de jogos, que possui um toolkit de agentes que podem ser ensinados a realizar alguma tarefa por meio de treinamento. Nesse sentido, a policy do agente será treinada por meio do algoritmo PPO (Proximal Policy Optimization) (SCHULMAN et al., 2015; SCHULMAN et al., 2017), isso é configurado pelo arquivo de extensão *.yaml* que garante que será esse algoritmo a ser utilizado.

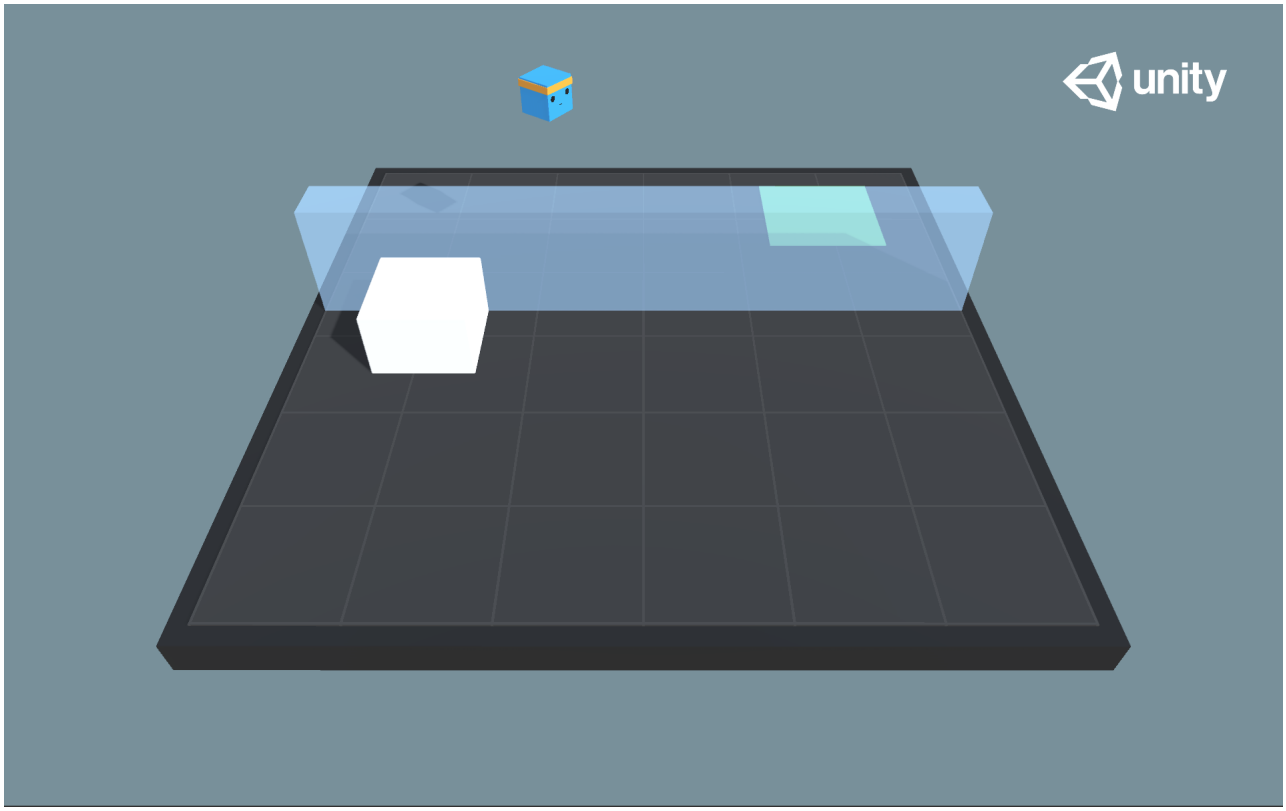

Figura 2 – Wall Jump Environment - Unity

Um dos ambientes que utilizamos foi o Wall Jump. Que consiste em ensinar um agente a pular uma parede para chegar no seu objetivo. O conjunto de ações que podem ser tomadas pelo agente é:

1. Movimento frontal: Frente, trás, no action (fica parado)
2. Movimento lateral: Esquerda, direita, no action (fica parado)
3. Rotação: Esquerda, direita, no action (fica parado)
4. Pular: Pula, no action (fica parado),

Além disso, o agente segue o padrão de recompensa da seguinte forma:

1. -0.0005, a cada passo (step).
2. +1.0 se o agente chegar ao objetivo.
3. -1.0 se o agente cair da plataforma.

O experimento com esse ambiente pode ser feito de duas formas: *Big Wall Jump* e *Small Wall Jump*. Em específico, a única diferença entre as duas situações é o tamanho da parede que o agente precisa pular. Dessa forma, o *Small Wall Jump* pula de uma barreira menor do que a capacidade de pulo dele, já no *Big Wall Jump* o agente precisa pular uma barreira que não será viável somente com seu pulo, necessitando empurrar o bloco branco (demonstrado na imagem 2) para utilizar ele como apoio e dai pular a barreira.

3.2 Planejamento dos Experimentos

Para entender melhor o comportamento da policy (cérebro do agente), vamos conduzir alguns experimentos utilizando o algoritmo citado anteriormente: Proximal Policy Optimization (PPO). O intuito é modificar de maneira controlada as variáveis que regem o comportamento do agente e daí extrair informações do comportamento do agente ao longo de um treinamento.

Nesse sentido, vamos modificar os hiper-parâmetros α (relacionado à taxa de aprendizagem), β (relacionado aos fatores entrópicos) e ϵ (relacionado ao limite de divergência entre a nova policy e a antiga policy) para analisar o comportamento do agente ao passar por um treinamento. Cada hiper-parâmetro da policy diz respeito a funções importantes do processo de treinamento, portanto, é imprescindível que exista uma avaliação da influência de cada um desses parâmetros no processo de aprenizagem.

3.2.1 Variáveis de Interesse dos Experimentos

Para cada experimento, vamos utilizar o Tensorboard para analisar três variáveis importantes no processo de treinamento: **Entropia** (grau de desordem da policy), **Recompensa acumulada** e **Tempo de um episódio**. Além disso, faremos um estudo detalhado a respeito da influência da utilização de um *Curriculum Learning*.

3.3 Introdução

Para utilizar o toolkit ml-agents no treinamento do nosso agente, é necessário que utilizemos um arquivo de configurações, que para o presente projeto, utilizaremos os arquivos do tipo PPO que seguem o algoritmo de otimização de policy citado nas seções anteriores.

Nesse sentido, para treinar nosso agente exclusivamente utilizando o terminal, criaremos um ambiente executável para que ele possa ser passado como uma linha de comando no terminal utilizando o "mlagents-learn".

Tendo os arquivos de configuração bem definidos com o que queremos analisar do aprendizado e o ambiente executável criado, podemos executar o programa e aguardar os resultados para observarmos nos gráficos pelo TensorBoard.

3.3.1 Definindo quais parâmetros analisar

Como visto anteriormente, é possível destacar vários parâmetros de treinamento para analisar suas influências. Dentre esses, destacaremos três hiper-parâmetros: beta (β), epsilon (ϵ) e learning rate (α).

1. Beta (β): (default = 5.0e-3) Força de regularização de entropia, que torna a policy "mais aleatória". Isso garante que o agente explore o espaço de ação durante o treinamento. Incrementando o beta, o policy tomará decisões mais desordenadas. Isso deve ser ajustado de acordo com o gráfico de entropia que pode ser analisado pelo TensorBoard. É de se esperar um decrescimento lento à medida com que houver aumento da recompensa. Se a entropia cair rapidamente, aumente o beta. Se a se a entropia cair lentamente, diminua o beta.

2. Epsilon (ϵ): (default = 0.2) Influencia em quão rapidamente a policy pode evoluir durante o treinamento. Corresponde ao limite aceitável de divergência entre a antiga e a nova policy durante a atualização do gradiente descendente. Colocando valores menores para epsilon vão haver atualizações mais estáveis, mas isso vai tornar o treinamento mais lento.

3. Learning Rate (α): (default = 3e-4) Taxa de aprendizado inicial para o gradiente descendente. Corresponde a força de cada etapa de atualização do gradiente descendente. Isso deve ser tipicamente diminuido se o treinamento estiver instável, e a recompensa não estiver sendo aumentada de maneira consistente.

Hiper-Parâmetros	Range Típico
β	1e-4 - 1e-2
ϵ	0.1 - 0.3
α	1e-5 - 1e-3

3.4 Experimentos com *Curriculum Learning* e *Transfer Learning*

Além dos experimentos citados até aqui, faremos uma analise detalhada a respeito da influência da utilização do *Curriculum Learning* sobre o processo de aprendizagem. Isso se deve, sobretudo, aos benefícios que a utilização pode trazer do ponto de vista do tempo de simulação. O tempo que o agente leva para convergir, ou seja, concluir o processo de aprendizagem, pode ser diminuido utilizando métodos como o *Curriculum Learning*.

3.4.1 Métricas para avaliar *Transfer Learning*

Existem muitas métricas para avaliar os benefícios de se utilizar os métodos de Transfer Learning, desde as source tasks à target task. Tipicamente é comparado a trajetória de aprendizagem do agente indo direto para a target task (sem transfer learning) e a trajetória em que ele passa por sources tasks (com transfer learning). Dessa forma, vamos analisar algumas métricas:

1. **Tempo para alcançar um limiar (time to threshold)**: Essa métrica é usada para avaliar o quão rápido o agente pode aprender para encontrar um retorno esperado $G_0 \geq \delta$ na tarefa alvo (target task), onde δ é o limiar da performance de desejo. O tempo pode ser medido em numeros de episodios tomados, número de ações tomadas, tempo de relógio.

2. **Perfomance assintótica (asymptotic performance)**: Compara o final da trajetória, depois de convergir utilizando um gráfico de Com transfer vs. Sem transfer.

3. **Salto no início (Jumpstart)**: É uma métrica que mede o desempenho já no início. Um confronto entre Com transfer vs. Sem transfer pode mostrar diferenças importantes nos gráficos de aprendizagem.

4. **Total de recompensa (Total reward)**: Mede o total da recompensa acumulada comparando o total da recompensa acumulada do agente em um gráfico de Com transfer vs. Sem transfer.

Além disso, podemos distinguir dois tipos de transferência de aprendizagem: A forte e a fraca. A **transferência fraca (Transfer weak)** é representada pela figura (3a), onde é possível ver que o treinamento feito nas tarefas de origem (source task) são

irrecuperáveis, ou seja, as figuras começam juntas. A **transferência forte (Transfer Strong)** é representada pela figura (3b), onde para compensar as tarefas de origem (sorce task) o gráfico muitas vezes é deslocado para a direita como podemos claramente ver em 3

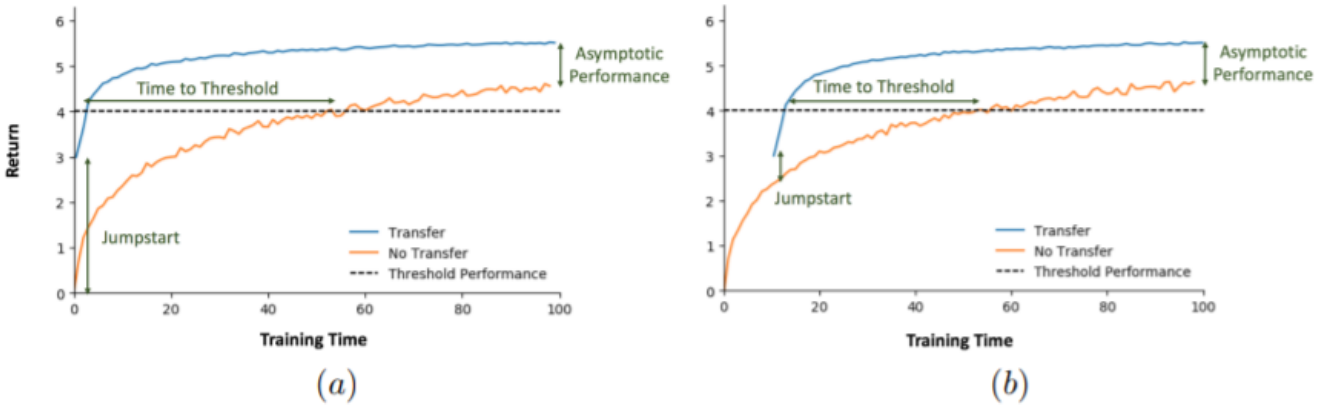

Figura 3 – Performance metrics for transfer learning using (a) weak transfer and (b) strong transfer with offset curves.

Analisando o gráfico 3, podemos ver também as métricas que foram citadas anteriormente. É possível utilizá-las para avaliar o quão eficaz pode ser a utilização do aprendizado por transferência.

4 RESULTADOS E DISCUSSÕES

4.1 Beta, Epsilon e Learning Rate

Com os hiper-parâmetros definidos, podemos variar alguns deles e constatar as consequências no aprendizado do agente ligadas às modificações. Dessa forma, vamos dividir algumas experiências em tópicos.

4.2 A influência do schedule Linear/Constante para os hiper-parâmetros

Por default, os hiper-parâmetros citados começam o aprendizado com o schedule linear, ou seja, com os hiper-parâmetros começando em seu valor máximo (inserido no arquivo de configuração) e decai linearmente até tocar seu valor mínimo no último step do processo de aprendizado.

Nesse sentido, por questões comparativas, vamos realizar dois processos de aprendizado, no qual investigaremos as diferenças entre o schedule linear (decaindo até o mínimo no último step) e constante (constante durante todo o processo).

4.2.1 Análise da recompensa acumulativa

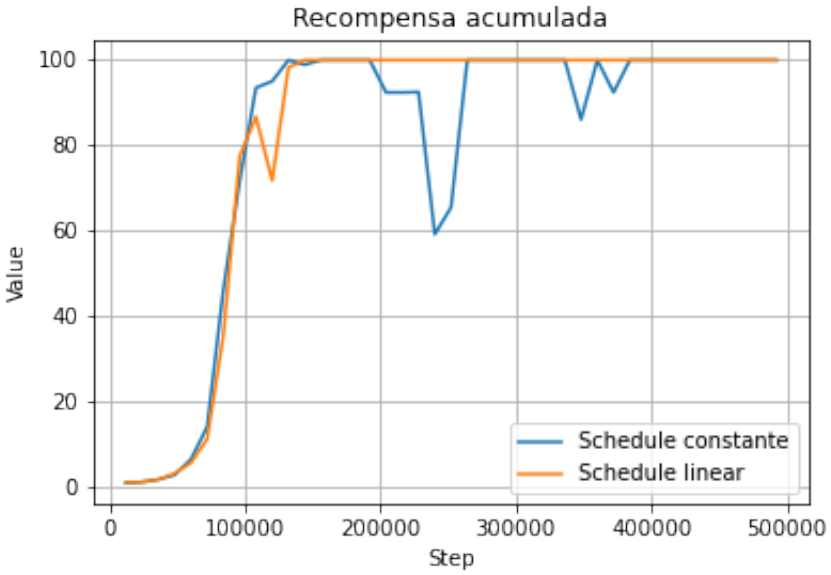

Figura 4 – Recompensa acumulativa x Steps

Vamos analisar o gráfico da recompensa com os dados resgatados do TensorBoard representado na figura 4. Note que a utilização do schedule linear trouxe maior estabilidade

ao aprendizado, pois nota-se que o gráfico laranja se estabiliza de maneira mais suave do que o gráfico azul, que ao chegar no topo de recompensa demonstra instabilidade em passos posteriores.

Isso pode ser explicado pela contribuição de três fatores que os hiper-parâmetros fornecem.

1. O β, responsável pelo controle da força de entropia do sistema, é imprescindível que seja decrementado a medida que o aprendizado evolui. Isso se deve, sobretudo, ao fato de que o agente tem maior liberdade para tomar decisões "mais aleatórias"somente quando seu aprendizado ainda não está significativamente alto. Em outras palavras, se a policy já está treinada o suficiente, não há motivos para o agente tomar decisões muito "aleatórias"do que ele já aprendeu durante o processo.

2. O ϵ, corresponde ao limite aceitável de divergência entre a antiga policy e a nova policy. Se esse limite continua relativamente grande quando o treinamento está praticamente completo, o efeito que pode ocorrer é uma sucessão de decisões erradas que podem contribuir negativamente para o aprendizado do agente, representado pela instabilidade no gráfico laranja.

3. O α, corresponde à taxa de aprendizado da policy, mantendo seu valor fixo, quando a policy chegar a um estado de relativa estabilidade é possível que haja um descontrole devido à maior liberdade de atualização da policy no treinamento.

Dessa forma, é possível notar que o schedule linear é o mais plausível a se utilizar, tendo em vista a maior estabilidade despendida pelo sistema ao decorrer do treinamento.

4.2.2 Análise da entropia

Nesse tópico, vamos analisar o comportamento da entropia em relação as mudanças de shedule. Antes de analisarmos, é de suma importância destacar que a entropia da policy pode ser vista como o grau de desordem do aprendizado e tomada de decisão do nosso agente. Nesse sentido, em diferentes situações precisaremos de diferentes valores de entropia para contribuir para o aprendizado do agente, porém, é necessário se atentar para o fato de que para uma policy já treinada é de se esperar que a entropia se torne minimamente estável. Isso se deve, sobretudo, ao fato de que um agente treinado que toma atitudes mais aleátorias devido a desordem pode contribuir negativamente para o que já foi aprendido.

Sob essa perspectiva, é possível observar, pelo gráfico, que o schedule constante demonstra de fato uma instabilidade nesse sentido, apesar da entropia no gráfico azul continuar diminuindo, ela não se apresentou com tendência de estabilização como o gráfico

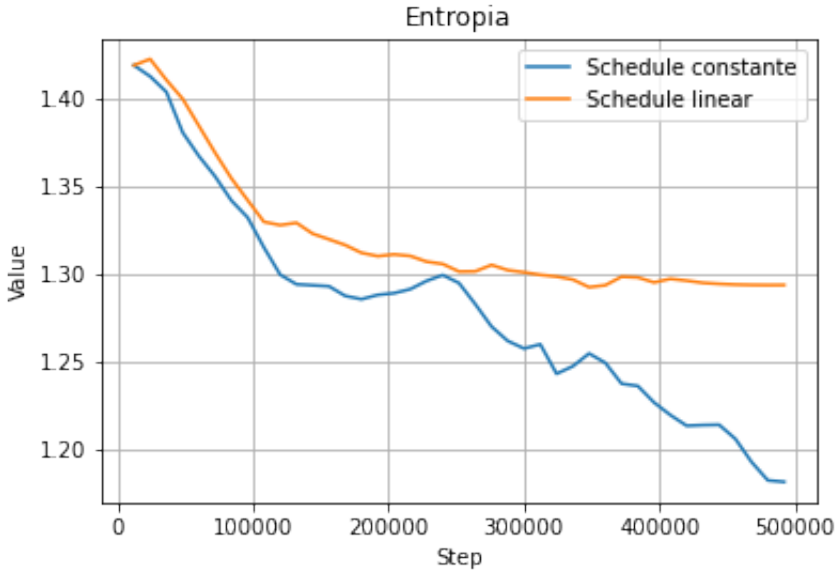

Figura 5 – Entropia x steps

laranja. Portanto, isso pode prejudicar (com instabilidades) o estágio da policy em que ela se encontra treinada, como vimos no gráfico de recompensa da figura 4.

4.3 Variação do beta

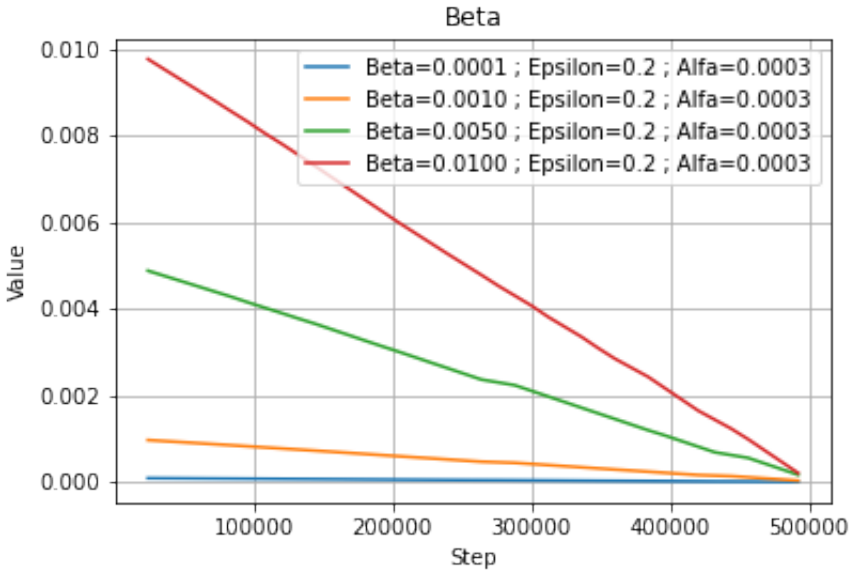

Figura 6 – Beta x steps

Para entender melhor a influência de cada hiper-parâmetro, vamos fixar o Epsilon e o Learning Rate, variando somente o beta para alguns valores dentro do seu range típico (com o schedule linear).

Nesse sentido, vamos variar o beta de acordo com os gráficos da figura 6 para analisar o comportamento dos gráficos de entropia da policy e de recompensa acumulada.

4.3.1 Recompensa acumulada

Como visto anteriormente, vamos analisar quais os aspectos de influência que a variação dos valores de beta tem sobre os graficos de recompensa do agente e da entropia da policy.

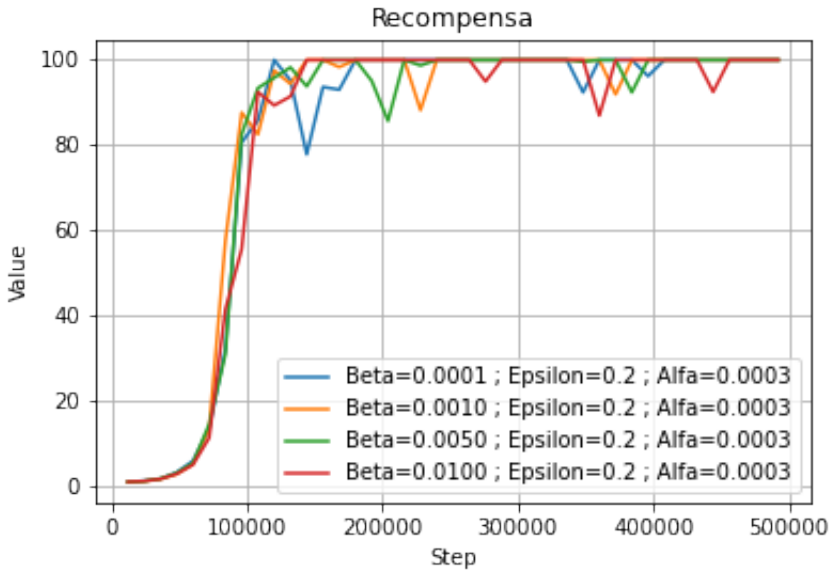

Figura 7 – Recompensa acumulada x steps

Nesse sentido, acompanhe os gráficos da figura 7. É possível identificar que algumas variações na faixa típica dos valores de beta não tem relevância considerável nos gráficos de recompensa representados pela figura 6 (lembre-se que estamos variando os valores iniciais de beta para o schedule linear).

4.3.2 Entropia

Para o gráficos de entropia, temos alguns efeitos mais visíveis. Por definição, é o hiper-parâmetro Beta que controla a força da entropia. Nesse aspecto, é por ele que podemos induzir a policy a aprender por meio do ganho por fatores mais "aleatórios". Quando a entropia decresce muito, aumente o beta, quando a entropia estabiliza, diminua beta.

Nesse sentido, acompanhe os gráficos da figura 8. É possível identificar que a ordem de tendência de estabilização dos gráficos de entropia da policy fornecem para nós a mesma ordem de valores de beta. Para valores mais altos de beta, teremos um valor absoluto de entropia mais alto.

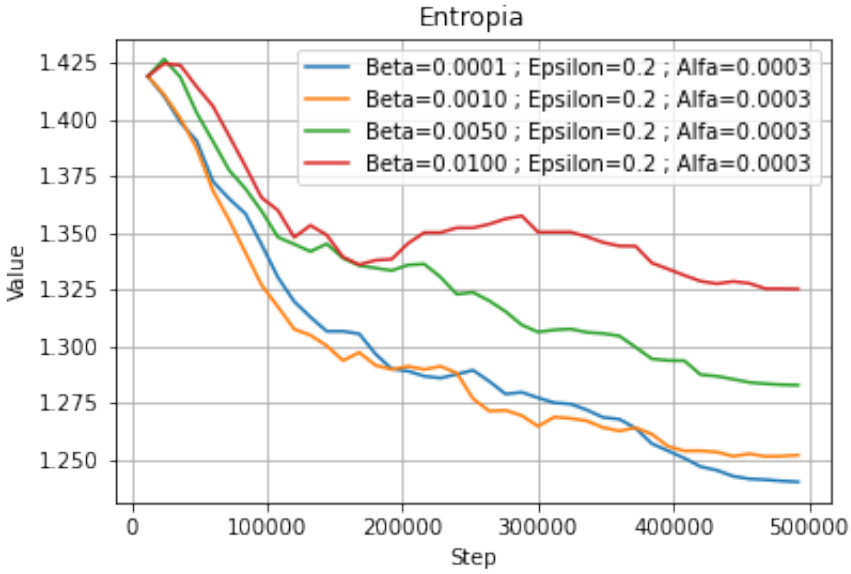

Figura 8 – Entropia x steps

4.4 Variação do Epsilon

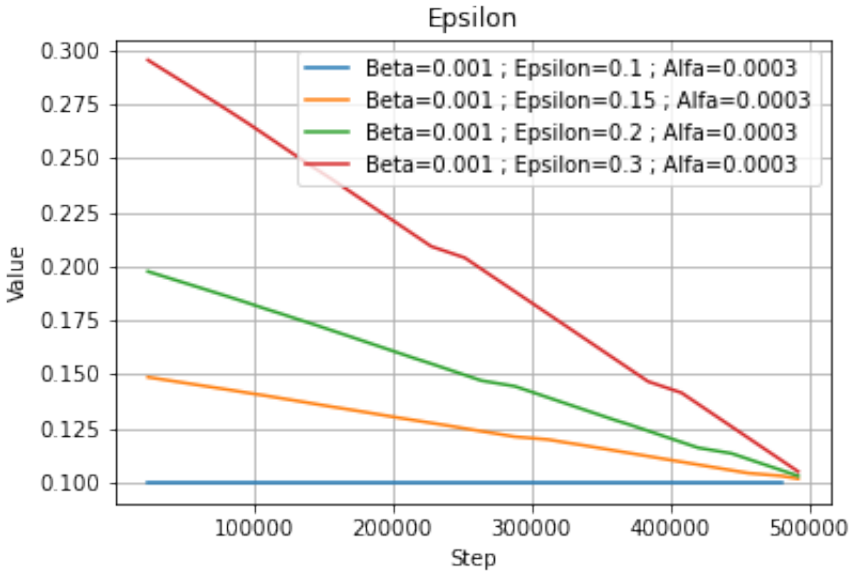

Figura 9 – Epsilon x steps

Dando continuidade ao estudo, analogamente ao caso anterior, vamos fixar os valores de beta e alfa, variando Epsilon com valores dentro do seu range de valores típicos. Nesse sentido, vamos variar o Epsilon de acordo com os gráficos da figura 9 para analisar o comportamento dos gráficos de entropia da policy e de recompensa acumulada.

4.4.1 Recompensa acumulada

Como visto anteriormente, vamos analisar quais os aspectos de influência que a variação dos valores de Epsilon tem sobre os graficos de recompensa do agente e da entropia da policy.

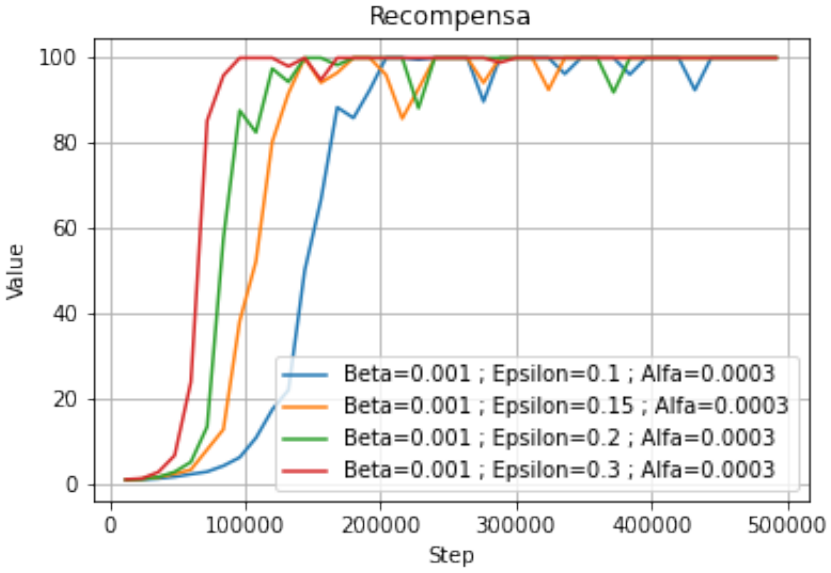

Figura 10 – Recompensa acumulada x steps

Nesse sentido, acompanhe os gráficos da figura 10. É possível identificar que, com algumas variações na faixa típica dos valores de Epsilon, existe uma influência desse hiper-parâmetro no tempo que a policy leva para ser treinada. Isso pode ser explicado pela definição do epsilon: A variação do limite aceitável de divergência entre a antiga e a nova policy pode tornar o processo de treinamento mais demorado e mais estável para epsilon menor (maior resistência à mudanças abruptas) ou menos demorado e menos estável para epsilon maior (menor resistência à mudanças abruptas).

4.4.2 Entropia

Por definição, é o hiper-parâmetro Epsilon controla o limite aceitável de divergência entra a antiga policy e a nova policy. Nesse aspecto, ele impõe que as mudanças da policy sejam mais gradativas ao "ignorar"valores de mudanças muito abruptas.

Nesse sentido, acompanhe os gráficos da figura 11. É possível identificar que a ordem de tendência de estabilização dos gráficos de entropia da policy fornecem para nós a ordem inversa do crescimento de Epsilon. Em outras palavras, para valores mais altos de Epsilon, teremos um valor absoluto de entropia mais baixo (levando-se em conta onde os valores tendem a estacionar.

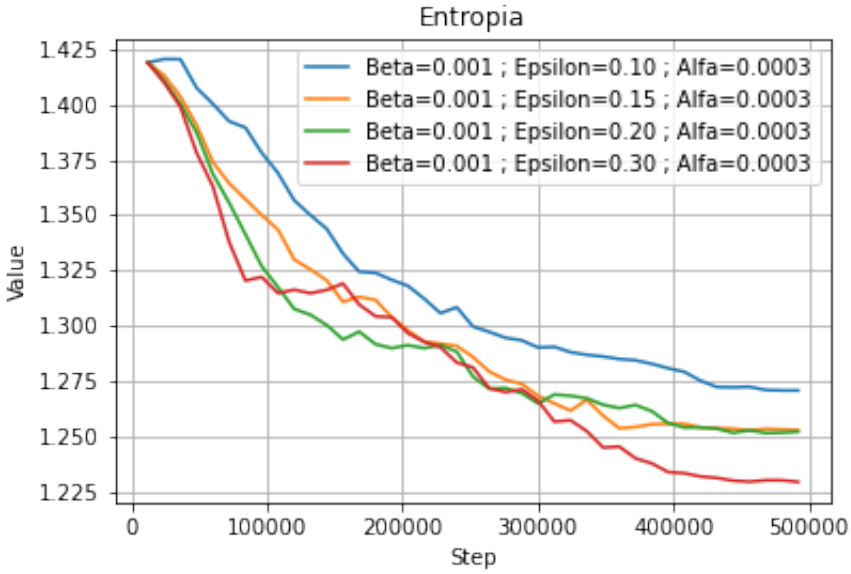

Figura 11 – Entropia x steps

4.5 Variação do Alfa

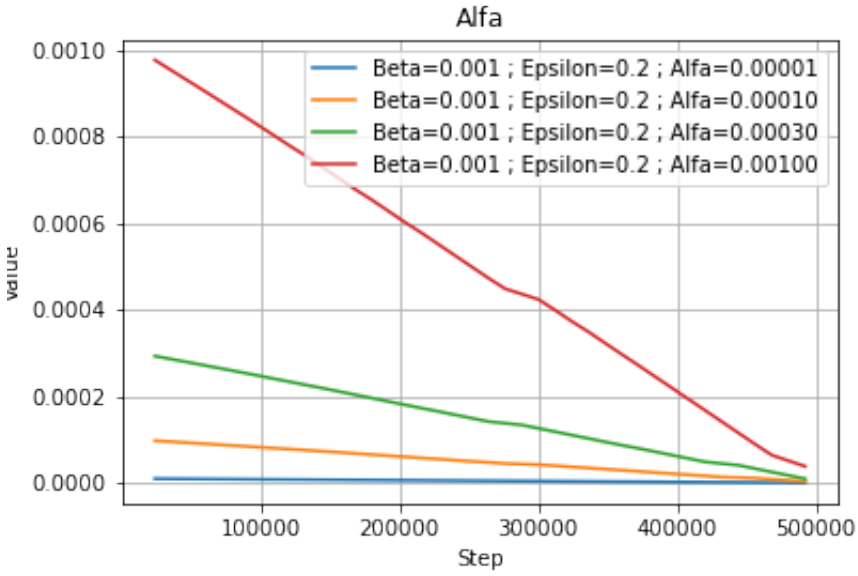

Figura 12 – Alfa x steps

Para finalizar a análise desses hiper-parâmetros, vamos fixar beta e epsilon, variando alfa dentro dos valores de seu range típico. Nesse sentido, vamos variar o Alfa de acordo com os gráficos da figura 12 para analisar o comportamento dos gráficos de entropia da policy e de recompensa acumulada.

4.5.1 Recompensa acumulada

Como visto anteriormente, vamos analisar quais os aspectos de influência que a variação dos valores de Alfa tem sobre os gráficos de recompensa do agente e da entropia da policy.

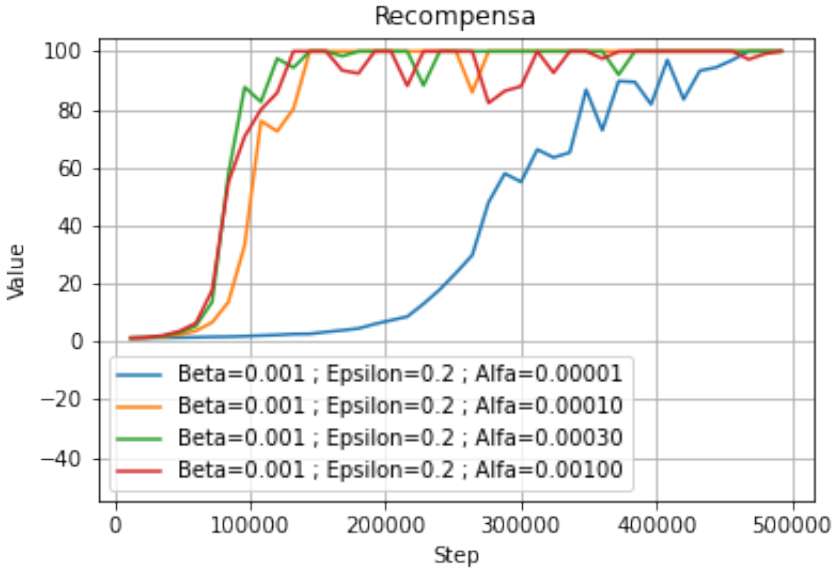

Figura 13 – Recompensa acumulada x steps

Nesse sentido, acompanhe os gráficos da figura 13. É possível identificar que algumas variações na faixa típica dos valores de Alfa tem influência considerável no gráfico de recompensa acumulada. Para valores mais baixos de Alfa, temos alterações consideráveis no tempo de convergência do treinamento da policy. Isso se deve, sobretudo, ao fato do parâmetro alfa representar a taxa de aprendizado que está diretamente ligada à taxa de atualização da policy. Portanto, para menores valores de alfa, maior será o tempo do processo de treinamento do agente.

4.5.2 Entropia

Para o gráficos de entropia, temos alguns efeitos muito curiosos. Por definição, é o hiper-parâmetro Alfa que controla a velocidade de aprendizado (taxa). Nesse aspecto, é por ele que podemos induzir a policy a aprender de forma mais lenta ou mais rápida. A influência disso na entropia é evidenciada pela tendência de estabilização, em função do valor de Alfa.

Nesse sentido, pela figura 14, é possível identificar que o aumento de Alfa corrobora na maior inclinação da reta tangente ao gráfico de entropia, fazendo com que o decrescimento da desordem seja mais acentuado. Além disso, a estabilização do sistema se dá em valores de entropia absoluta mais baixa (interpretado como uma maior estabilidade

da policy). Portanto, para maiores valores de alfa, maior será a taxa de decrescimento da entropia e menor será o seu nível absoluto de estabilização.

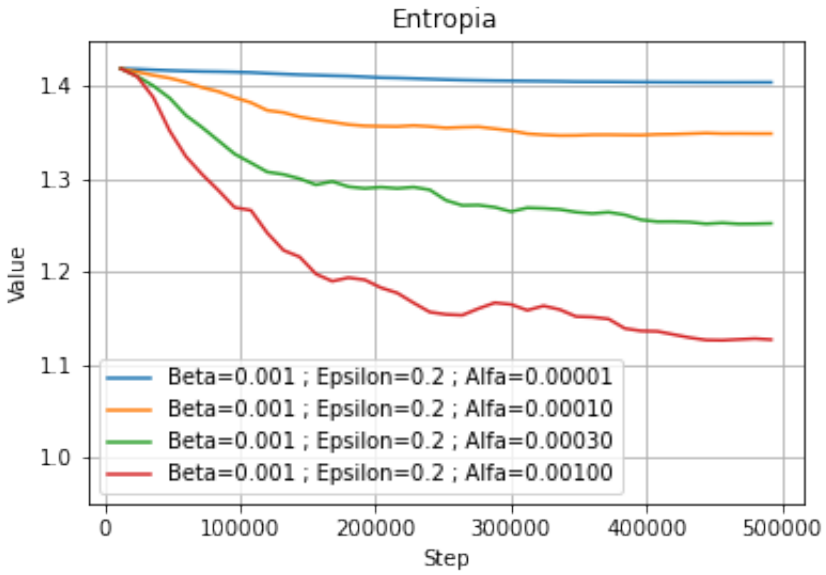

Figura 14 – Entropia x steps

4.6 *Small Wall Jump* e *Big Wall Jump*

Tendo tudo configurado da maneira correta, o treinamento pode ser iniciado. Isso deve demorar cerca de um dia, pelo menos, de treinamento. Após finalizar, é possível analisar os desempenhos por meio do Tensorboard. Nesse aspecto, o ambiente foi executado utilizando dois modelos, um para cada caso:

1. SmallWallJump: Este é utilizado para os casos em que o muro (quando tem altura) possui altura suficientemente baixa de tal forma que o pulo do agente é suficiente para ultrapassar a barreira. Note que esse caso é mais direto, e será possível comprovar isso por meio dos gráficos.

2. BigWallJump: Este é utilizado para os casos em que o muro possui altura maior do que a altura de pulo do agente, dessa forma, será necessário que a policy seja treinada a conduzir o agente a empurrar o bloco disponível por ele, para que ele pule no bloco (que possui altura mais baixa) e em seguida pule o muro. Note que essa tarefa é mais complexa do que a anterior, e para isso o curriculum learning será crucial, no sentido de que inicialmente serão inseridas alturas mais baixas, aumentando a dificuldade das tarefas a medida com que elas vão sendo conquistadas.

4.6.1 *Small Wall Jump*

Em primeiro lugar, vamos analisar os resultados associados ao primeiro modelo (mais simples). É importante destacar que os hiper-parâmetros que analisamos nas seções anteriores estão setados como default e schedule linear.

4.6.1.1 Recompensa acumulativa, tamanho do episódio e entropia

Para o caso do modelo SmallWallJump, é de se esperar que exista uma maior regularidade no processo de treinamento. Sem conter momentos de mudança brusca no regime que o gráfico já vem desempenhando. Isso se deve, sobretudo, ao fato de que não vão haver mudanças de lições ao decorrer das atividades menores programadas no curriculum que possam afetar drasticamente o processo de aprendizado.

Figura 15 – Recompensa x steps - SmallWallJump

Nesse sentido, podemos confirmar isso através dos resultados de três gráficos importantes. O primeiro está localizado na figura 15, onde consta um regime relativamente regular durante todo processo de treinamento.

Figura 16 – Tamanho do episódio x steps - SmallWallJump

O segundo, é evidenciado pelo tempo de cada episódio na figura 16. É de se esperar que, a medida com que a policy vai sendo treinada, o agente vai tomando decisões cada vez mais assertivas, isso encurta o tempo do episódio e torna a inclinação média da reta tangente do gráfico cada vez mais próxima de zero (cada vez menos variação).

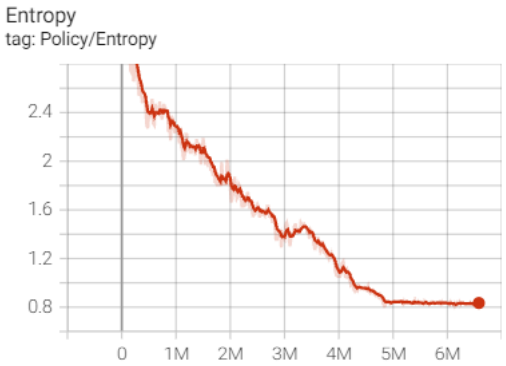

Figura 17 – Entropia x steps - SmallWallJump

Por fim, é possível analisar a entropia da policy. Seguindo a mesma linha de raciocínio. No início do treinamento, é de se esperar a entropia esteja um pouco mais alta e vá decaindo com o tempo, haja vista o início ser o momento em que mais é necessário que o agente tome atitudes "mais aleatórias"para que haja um ganho de recompensa por desordem. Com o tempo, atitudes mais desordenadas serão menos frequentes e a policy terá uma entropia relativamente estável.

Isso pode ser evidenciado pela figura 17, onde a policy condiz com o previsto anteriormente, destacando o momento de estabilização que está claramente evidenciado por volta do step 5M.

4.6.2 *Big Wall Jump*

Por fim, iremos explorar o caso mais complexo como foi citado anteriormente. É importante destacar que os hiper-parâmetros que analisamos nas seções anteriores estão setados como default e schedule linear.

4.6.2.1 Recompensa acumulativa, tamanho do episódio e entropia

Para o caso do modelo BigWallJump, é de se esperar que exista uma menor regularidade no processo de treinamento. Nesse aspecto, é possível que existam pontos de mudanças abruptas. Isso se deve, sobretudo, ao fato de que vão haver mudanças de lições ao decorrer das atividades menores programadas no curriculum que possam afetar drasticamente o processo de aprendizado.

Nesse sentido, podemos confirmar isso através dos resultados de três gráficos importantes. O primeiro está localizado na figura 18, onde consta dois momentos de quebra da regularidade do processo de aprendizado.

1. Por volta do step 2M, existe uma leve queda na recompensa, isso se deve ao incremento da altura do muro durante o processo de aprendizado.

Figura 18 – Recompensa x steps - BigWallJump

2. Por volta do step 5M, existe uma queda brusca na recompensa, isso se deve ao fato da altura do muro ter atingido um valor maior do que o pulo do agente. Portanto, é nesse momento em que a policy precisa se adaptar para conduzir o agente a utilizar o bloco disponível para ultrapassar a barreira.

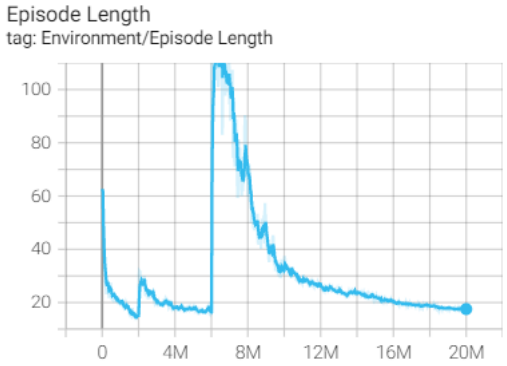

Figura 19 – Tamanho do episódio x steps - BigWallJump

Além disso, é possível identificar as consequências citadas acima sobre o tamanho de cada episódio. É possível notar, através do gráfico na figura 19, que nos dois momentos citados houve um aumento drástico no tamanho do episódio. Isso se deve às novas dificuldades que a policy precisa se adaptar para conseguir atingir o objetivo.

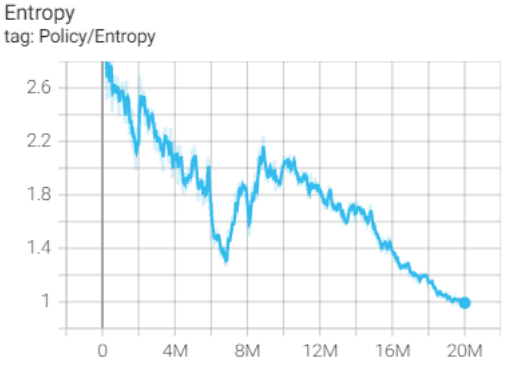

Figura 20 – Entropia x steps - BigWallJump

Por fim, a análise da entropia revela os aspectos de desordem enfretados pela policy na figura 20. Para os dois momentos de "virada"citados anteriormente, podemos destacar que:

1. Por volta do step 2M, existe um aumento leve de entropia, devido à mudança de lição presente no curriculum.

2. Por volta do step 5M, existe um aumento considerável no valor absoluto da entropia. Isso pode ser explicado pelo fato de que nesse momento a policy precisa conduzir o agente a tomar decisões mais "aleatórias"para que haja um ganho de aprendizado pelo fator de desordem. O que se mostra benéfico para o agente, pois claramente a entropia mostra uma tendência de estabilização em niveis mais baixos (maior estabilidade) após a conclusão das tarefas impostas pelo curriculum

4.6.3 Comparativo do uso do Curriculum Learning

Para termos uma noção da influência de se utilizar o curriculum learning, é importante que vejamos um comparativo do processo de aprendizado (Com Curriculum e Sem Curriculum). Para tanto, vamos analisar os gráficos de recompensa acumulada e de entropia, para que possamos interpretar as principais diferenças entre o aprendizado com e sem curriculum.

4.6.4 Recompensa Acumulada

Como vimos em seções anteriores, o curriculum é, em suma, o fracionamento de um objetivo em tarefas menores. A primeira implicação que isso pode ter é que o aprendizado sem o curriculum pode apresentar maior tempo para que o agente consiga se adequar as atividades impostas. Isso se deve à falta de linearidade na sequência de atividades. Com curriculum, o agente se adequa mais fácil, primeiramente as atividades menores, para progredir de forma gradativa (na maioria das vezes).

Dessa forma, acompanhe o gráfico da figura 21. É possível identificar que, como previsto anteriormente, o gráfico que representa o aprendizado sem curriculum demanda uma maior resistência ao crescimento e estabilização. Em contrapartida, o gráfico que representa o aprendizado com curriculum demonstra um crescimento considerável logo no inicio do aprendizado, seguido de uma queda considerável por volta do passo 6M (que representa a mudança de atividade presente no curriculum). Portanto, a utilização do curriculum tornou o aprendizado mais gradativo do que o aprendizado sem curriculum.

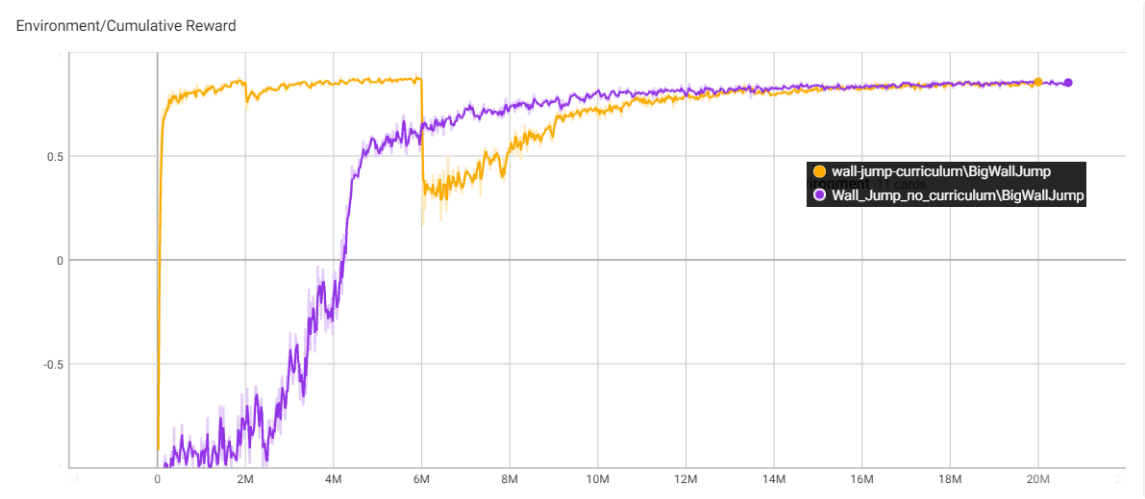

Figura 21 – Recompensa acumulada x steps - Roxo: Sem curriculum | Amarelo: Com curriculum

4.6.5 Entropia

Do ponto de vista da entropia, existe um aspecto bastante interessante a se discutir. A sequência de aprendizado mais gradativa apresentada no uso do curriculum pode ser um forte fator que contribui com o maior controle e diminuição da entropia da policy. Isso se deve, sobretudo, a maior desordem do aprendizado quando ele não tem um roteiro com etapas menores para se seguir.

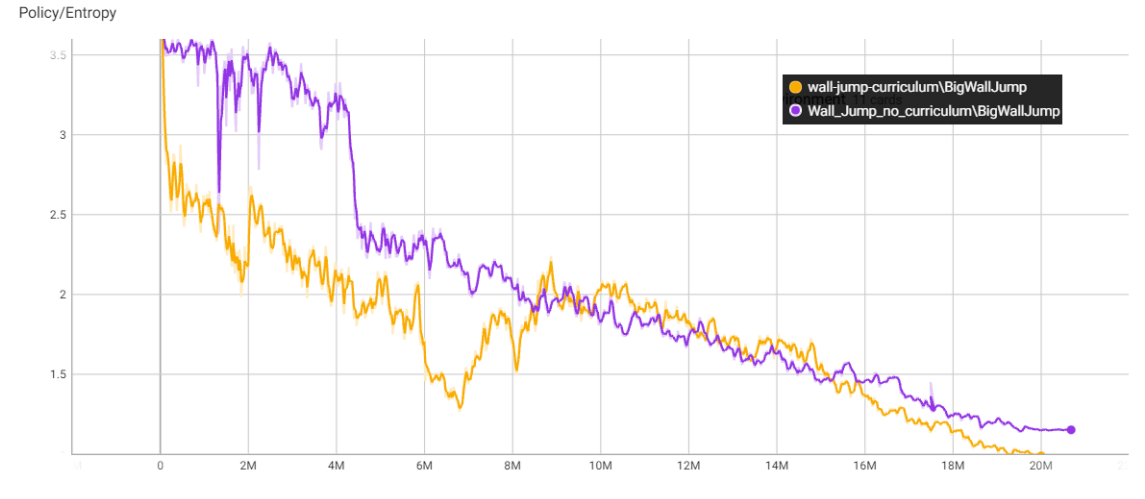

Figura 22 – Entropia x steps - Roxo: Sem curriculum | Amarelo: Com curriculum

Dessa forma, analisando o gráfico da figura 22, é possível notar exatamente o que foi mencionado anteriormente. O gráfico que representa o aprendizado sem curriculum se mantém acima do gráfico com curriculum até por volta do passo 9M, quando daí eles seguem muito próximos do ponto de vista da entropia.

5 CONCLUSÕES E TRABALHO FUTURO

5.1 Influência dos hiper-parâmetros

A análise da influência dos hiper-parâmetros para os modelos utilizados está sendo de fundamental importância para entender o comportamento do agente ao ser treinado para realizar uma determinada tarefa. Isso tem impacto direto na melhor avaliação de como podemos desenvolver sistemas inteligentes, no qual o agente será apto para desempenhar seu papel. Isso se deve, sobretudo, ao monitoramento detalhado de variáveis como: **Recompensa acumulativa**, **Entropia da policy** e **tempo de um episódio**.

5.2 Benefícios do *Curriculum Learning*

O critério comparativo do uso do *Curriculum Learning* é crucial para a demonstração de que o método pode ser extremamente benéfico para algumas situações e indispensáveis para outras. Em alguns casos, o processo de treinamento poderia levar muito mais tempo para ser concluido, o que nos traz um enorme ganho de tempo e recurso de processamento. A divisão de uma tarefa maior em tarefas menores traz um caráter intuitivo e funcional no processo de aprendizagem do agente.

5.3 Trabalhos futuros

Nesta etapa, visamos contemplar um maior número de algoritmos de optimização de policy, incluindo Gradiente Estocástico Descendente e Algoritmos Genéticos. A análise performática de cada um deles é crucial para estabalecer um critério comparativo para, sobretudo, propor soluções mais convenientes para os problemas que temos na atualidade. Nesse sentido, visamos ter uma maior clareza a respeito do funcionamento de alguns dos principais algoritmos utilizados hoje em problemas de Aprendizado por Reforço. Dessa forma, poderemos reconhecer e até propor soluções que podem ser mais viáveis que outras do ponto de vista de tempo de aprendizado e custo de processo. (MNIH et al., 2016)

REFERÊNCIAS

GéRON, A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. [S.l.]: Oreilly, 2019. v. 2nd. Citado na página 5.

MNIH, V.; BADIA, A. P.; MIRZA, A. G. M.; LILLICRAP, T. P.; HARLEY, D. S. T.; KAVUKCUOGLU, K. Asynchronous methods for deep reinforcement learning. *arXiv*, v. 1, n. 1602.01783, p. 0–19, 2016. Disponível em: <https://arxiv.org/pdf/1602.01783.pdf>. Citado na página 30.

MNIH, V.; SILVER, K. K. nad D.; GRAVES, A.; ANTONOGLOU, I.; WIERSTRA, D.; RIEDMILLER, M. Playing atari with deep reinforcement learning. *arXiv*, v. 1, n. 1602.01783, p. 0–19, 2013. Disponível em: <https://arxiv.org/pdf/1312.5602v1.pdf>. Citado na página 5.

NARVEKAR, S. Curriculum learning in reinforcement learning. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*. [S.l.: s.n.], 2017. p. 1–2. Citado na página 8.

NARVEKAR, S.; PENG, B.; LEONETTI, M.; SINAPOV, J.; TAYLOR, M. E.; STONE, P. Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv*, v. 1, n. 2003.04960, p. 0–50, 2020. Disponível em: <https://www.jmlr.org/papers/volume21/20-212/20-212.pdf>. Citado na página 8.

SCHULMAN, J.; MORITZ, S. L. P.; JORDAN, M.; ABBEEL., P. High-dimensional continuous control using generalized advantage estimation. *arXiv*, v. 1, n. 1506.02438, p. 0–14, 2015. Disponível em: <https://arxiv.org/pdf/1506.02438.pdf>. Citado na página 11.

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. Proximal policy optimization algorithms. *arXiv*, v. 1, n. 1707.06347, p. 0–12, 2017. Disponível em: <https://arxiv.org/pdf/1707.06347.pdf>. Citado na página 11.