

Desenvolvimento de uma Inteligência Artificial para aprender a jogar jogos em Allegro

Arthur de Senna Rocha

Universidade Federal de Minas Gerais
Escola de Engenharia

Trabalho de Conclusão de Curso II

26 de outubro de 2020

Sumário

- 1 Introdução
- 2 Modelagem do Sistema
- 3 Implementação
- 4 Análise dos Resultados
- 5 Considerações Finais
- 6 Referências

Introdução

- Cenário mundial com grande avanço de técnicas de ML
- Reconhecimento de padrões e previsões cada vez mais precisas
- Sistemas de DL aplicados a conjuntos de aplicações cada vez mais amplos



Introdução

- A complexidade das tarefas que podem ser resolvidas por DL vêm crescendo significativamente
- Valor para pesquisa em múltiplas áreas da ciência
- Aplicações de aprendizado de máquina e *deep learning* são altamente lucrativas
- Potencial de investimento em pesquisa, modelagem de novos problemas e estudo de técnicas de aprendizado de máquina

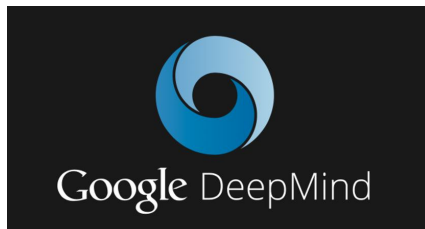


Introdução

Proposta

- Inspirado pelo trabalho realizado pelo *Deep Mind* (MNIH et al., 2013)
- Desenvolver uma IA capaz de aprender a jogar diferentes jogos em *Allegro*
- Algoritmo de Deep Reinforcement Learning
- Nenhuma regra sobre o jogo é dada e, inicialmente, a IA não tem informações sobre o que precisa fazer

Motivação

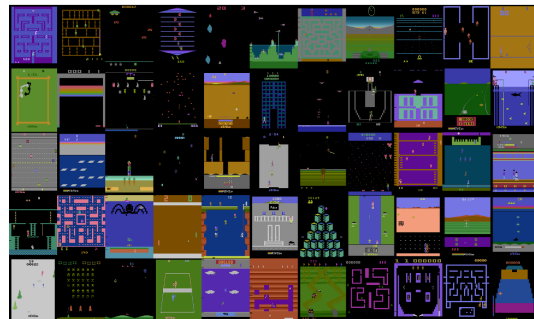


Jogando Atari com DRL

- *Deep Mind* desenvolveu uma rede neural convolucional
- Recebe como entrada uma matriz multi-dimensional correspondente aos pixels da tela em cada instante do jogo
- Aplicando técnicas de *Reinforcement Learning*, a rede é treinada para jogar diferentes jogos

Motivação

- Modelo usado com sucesso para treinar múltiplos jogos Atari 2600
- Proposta de desenvolver um sistema semelhante, mas voltado para jogos em *Allegro*



O Ambiente



Allegro



Allegro

- *Allegro* (HARGREAVES, 1990) é uma biblioteca de C e C++ voltada principalmente para videogames e programação de multimídia
- Permite lidar com tarefas comuns de baixo nível:
 - criar janelas
 - aceitar entrada do usuário
 - carregar dados
 - desenhar imagens
 - reproduzir sons

Descrição do Problema

O Agente

- O sistema receberá, inicialmente, somente as limitações físicas do jogo
- O agente deve ser capaz de elaborar uma estratégia para maximizar sua pontuação
- O sistema deverá ser capaz de lidar com cenários aleatórios e não-aleatórios
- O sistema deve ser generalizado para que possa ser aplicado à diferentes cenários e treinado para jogar diferentes jogos

Descrição do Problema

O Agente

- O sistema receberá, inicialmente, somente as limitações físicas do jogo
- O agente deve ser capaz de elaborar uma estratégia para maximizar sua pontuação
- O sistema deverá ser capaz de lidar com cenários aleatórios e não-aleatórios
- O sistema deve ser generalizado para que possa ser aplicado à diferentes cenários e treinado para jogar diferentes jogos

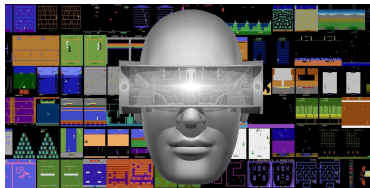
Restrições

- Acesso ao código fonte dos jogos
- Jogos devem ser implementados em *Allegro*
- Jogos devem ser 2D

Descrição do Problema

Objetivos

- Criar e treinar uma rede neural convolucional capaz de aprender políticas através de pixels brutos em ambientes complexos
- O agente deve alcançar resultados superiores aos de uma abordagem aleatória e próximos aos de um agente humano
- Implementar um agente que seja capaz de aprender a jogar o maior número de jogos possíveis sem conhecimento prévio do ambiente



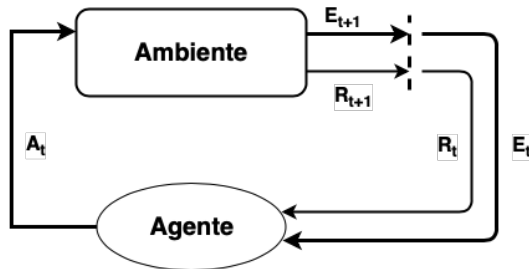
Modelagem do Sistema

Reinforcement Learning

- Abordagem computacional para entender e automatizar o aprendizado direcionado a objetivos e a tomada de decisões
- Ênfase na aprendizagem de um agente a partir da interação direta com seu ambiente, sem exigir supervisão exemplar ou modelos completos do ambiente
- Abordagem caracterizada por tentativa e erro e recompensa atrasada

Reinforcement Learning

- A **política** define a maneira que o agente deve se comportar em um determinado momento
- Um **signal de recompensa** define o objetivo de um problema de aprendizado por reforço
- A **função de valor** especifica o que é bom a longo prazo
- Objetivo final é maximizar a função de valor



Reinforcement Learning

A Função de Retorno

- Deseja-se encontrar a política que maximize as recompensas obtidas a longo prazo
- A função de retorno pode ser calculada como a soma das recompensas futuras
- Um fator de desconto γ é adicionado à equação, de forma a ajustar o peso de recompensas futuras
- γ deve assumir um valor $0 < \gamma < 1$, e geralmente é atribuído um valor de $\gamma \approx 0.9$, pois quanto maior o valor de γ , maior será o peso das recompensas futuras

$$R(t) = \sum_{\tau=1}^{T-t} \gamma^{\tau-1} R(t+\tau) = R(t+1) + \gamma R(t+2) + \dots + \gamma^{T-t-1} R(T) \quad (1)$$

Reinforcement Learning

A Política

- A política π ótima para um agente, pode ser definida como escolher a ação a em um estado s que irá maximizar a função de retorno esperado \mathbb{E}

$$F^*(s, a) = \max_{\pi} (\mathbb{E}[R_t | s_t = s, a_t = a, \pi]) \quad (2)$$

Reinforcement Learning

A Política

- A política π ótima para um agente, pode ser definida como escolher a ação a em um estado s que irá maximizar a função de retorno esperado \mathbb{E}

$$F^*(s, a) = \max_{\pi} (\mathbb{E}[R_t | s_t = s, a_t = a, \pi]) \quad (2)$$

Limitações

- Para calcular os retornos é necessário esperar que o episódio termine
- Para cenários com episódios muito longos ou infinitos esse modelo pode ser ineficiente

Reinforcement Learning

Q-learning

- O método de aprendizagem conhecido como *Q-learning* aproxima o retorno esperado de forma recursiva
- Não é necessário esperar até que o episódio termine
- Podemos atualizar o valor da função de valor $Q(s, a)$ sempre que uma ação a for tomada

$$Q(s_t, a_t) = r_t + \gamma \cdot \max_{a_{t+1}} (Q(s_{t+1}, a_{t+1})) \quad (3)$$

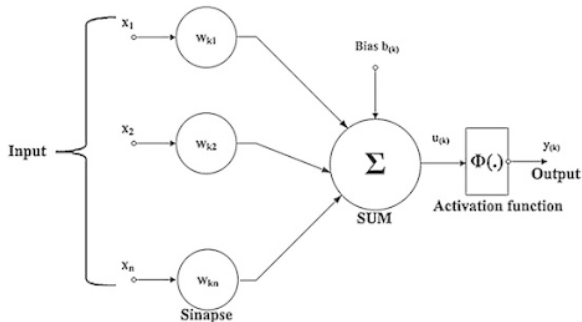
Modelagem do Sistema

Deep Learning

- Área do aprendizado de máquina que propõe que os computadores aprendam com a experiência
- Ajustem à novas entradas de dados
- Compreendam o mundo em termos de hierarquia de conceitos, sendo cada conceito definido por sua relação com conceitos mais simples

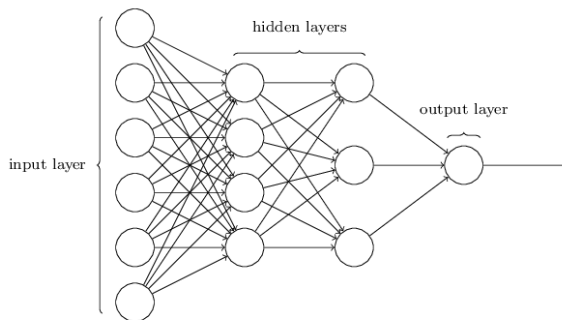
Rede Neural

- Rede neural composta de múltiplas camadas
- Cada neurônio é caracterizado pelo um **peso**, **bias** e uma **função de ativação**
- A informação se move da camada de entrada para as camadas ocultas

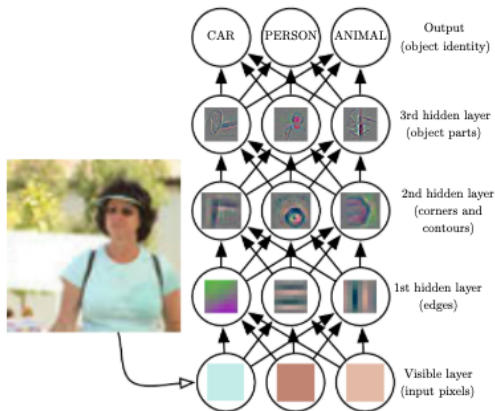


Rede Neural

- As camadas ocultas fazem o processamento e enviam a saída final para a camada de saída
- Os pesos e bias dos neurônios são atualizados com base no erro
- Uma vez que todos os dados passaram por este processo, os pesos e bias finais são usados para previsões



Deep Learning



Modelagem do Sistema

Deep Q Learning

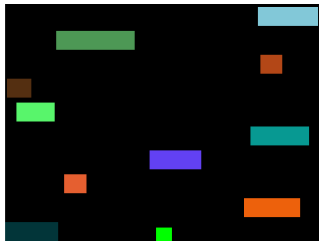
- O *deep reinforcement learning* (DRL) é uma abordagem do *deep learning* que utiliza as técnicas de aprendizagem por reforço para treinar o agente
- Essa abordagem consiste em fornecer ao sistema parâmetros relacionados ao seu estado e uma recompensa positiva ou negativa com base em suas ações
- O *Deep Q Learning* (DQN) é um tipo de DRL onde uma rede neural é treinada utilizando a abordagem de *Q-learning*
- No DQN os pesos da rede são atualizados conforme a **Equação 4**

$$\theta_t = \theta_t + \alpha(r + \gamma \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \nabla_{\theta} Q(s_t, a_t) \quad (4)$$

O Jogo

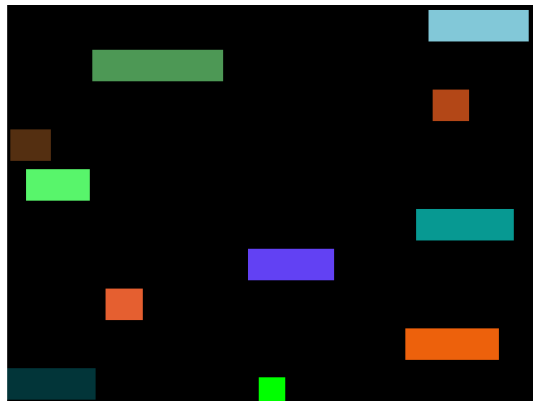
Frogger

- O ambiente utilizado para treinamento da rede foi o jogo *Frogger*
- Jogo simples, cujo objetivo é levar o agente da posição inicial até o topo da tela sem colidir com nenhum obstáculo
- As ações possíveis em cada estado são: 'cima', 'baixo', 'esquerda', 'direita' e 'não se mexer'



Recompensas

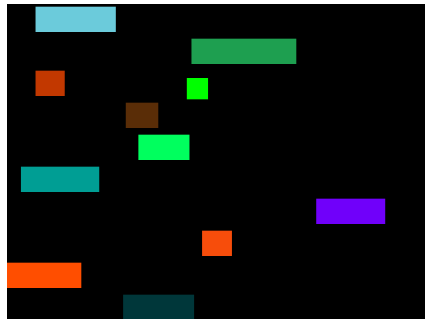
- $r = -0.05$ caso o agente realize uma ação que o mantenha na mesma linha que se encontrava previamente
- $r = 1$ caso o agente se mova para cima
- $r = -1$ caso o agente se mova para baixo
- $r = -1$ caso o agente realize uma ação que o leve a colidir com algum obstáculo
- $r = 10$ caso o agente alcance seu objetivo



Implementação

Estados do Jogo

- A pontuação do jogo pode depender de uma sequência anterior de ações
- A análise do estado do jogo pode ser mal-representada observando apenas a imagem x_t do instante atual



Implementação

Estados do Jogo

- Para solucionar esse problema considera-se como o estado atual s_t , uma sequência de observações $s_t = (x_{t-n}, a_{t-n}, \dots, a_{t-1}, x_t)$
- Fornece ao agente um melhor contexto sobre o estado em que se encontra

Implementação

Entrada do Sistema

- Os dados de cada observação são extraídos dos pixels exibidos na tela em cada instante de tempo
- Tela do jogo representa uma janela de 640×480 pixels para cada instante
- Cada estado constitui um conjunto de $n = 4$ instantes
- Vetor de entrada resultante tem um tamanho de $640 \times 480 \times 3 \times 4 = 3686400$

Implementação

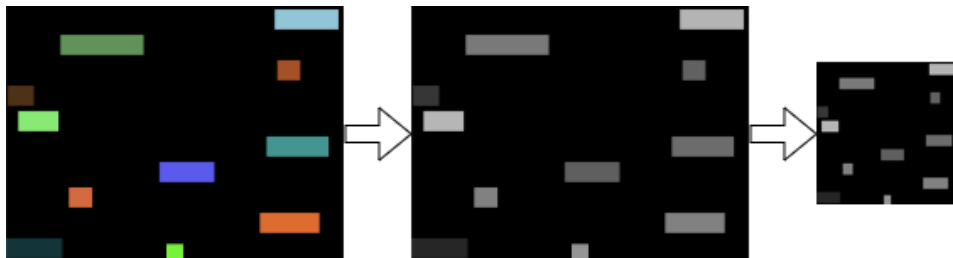
Entrada do Sistema

- Os dados de cada observação são extraídos dos pixels exibidos na tela em cada instante de tempo
- Tela do jogo representa uma janela de 640×480 pixels para cada instante
- Cada estado constitui um conjunto de $n = 4$ instantes
- Vetor de entrada resultante tem um tamanho de $640 \times 480 \times 3 \times 4 = 3686400$

Pré-processamento de Imagens

- Volume de dados extremamente alto
- É necessário reduzir esse tamanho para obtermos uma solução viável

Pré-processamento de Imagens



Redução de Imagens

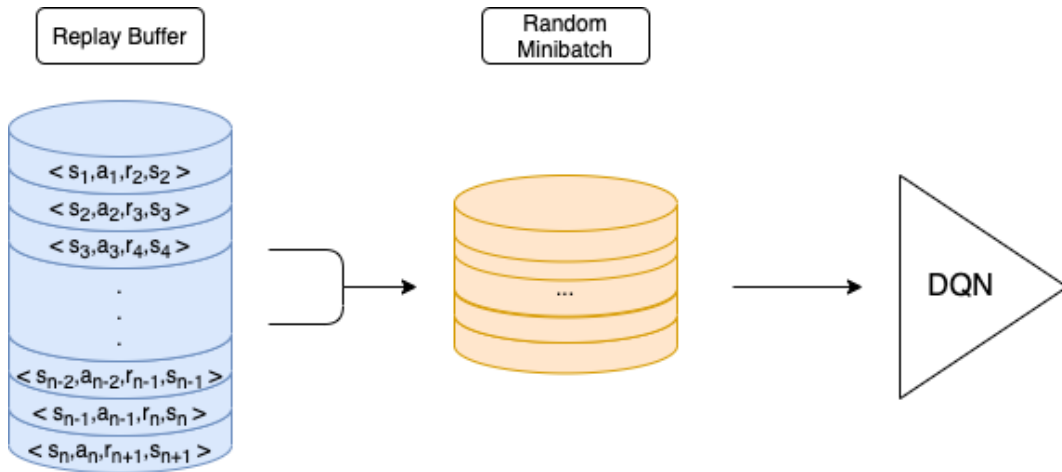
- Redução do vetor de entrada
- Remove dados desnecessários
- Vetor de entrada resultante $84 \times 84 \times 4 = 28224$
- Redução de 99.23%

Dados de Treinamento

Experience Replay

- Todas as observações realizadas pelo agente são armazenados em um *Replay Buffer*
- Para treinar a DQN recupera-se um conjunto de dados amostrados aleatoriamente da experiência do agente até o presente momento

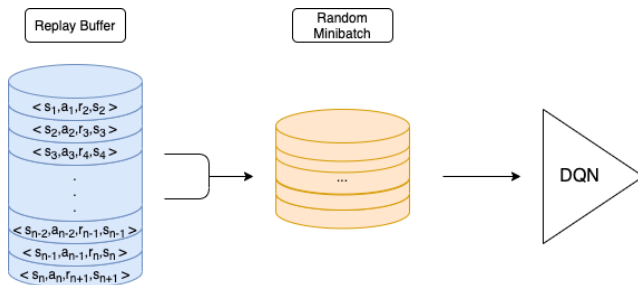
Experience Replay



Dados de Treinamento

Experience Replay

- O *Replay Buffer* é uma lista FIFO
- Permite otimizar o armazenamento de dados
- Fornece uma melhor distribuição de informações para o aprendizado



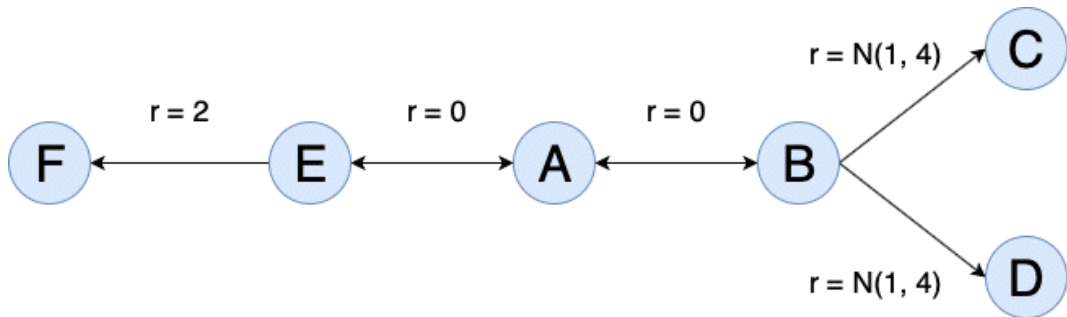
Rede Neural Dupla

$$Q(s_t, a_t) = r + \gamma \cdot \max(Q(s_{t+1}, a_{t+1}) | \forall a_{t+1})$$

Problema do aprendizado Q

- Equação estima o valor das recompensas
- Problema surge do valor máximo
- Em ambientes com ruído a função é tendenciosa

Rede Neural Dupla



Rede Neural Dupla

Solução

- Solução proposta por (HASSELT; GUEZ; SILVER, 2015)
- Consiste na implementação de duas redes neurais, uma rede primária e uma rede objetivo
- Somente a rede primária é treinada em cada iteração
- Rede objetivo recebe os pesos da rede primária periodicamente
- Ajuda a estabilizar os pesos da rede objetivo

Pseudocódigo

Algorithm 1: Corpo Principal

Inicializa o ambiente

Inicializa a rede primária

Inicializa a rede objetivo

for *episodio* = 1:NUM_EPISODIOS **do**

while *jogo não termina* **do**

ação ← modelo.obtemAcao(estado, ϵ)

proximaObservacao, *recompensa*, *fim* ← ambiente.executaAcao(*ação*)

proximoEstado ← obtemEstado(estado, *proximaObservacao*)

experienceReplay.adicionaTupla(estado, *ação*, *recompensa*, *proximoEstado*, *fim*)

treinaModelo()

end

end

Pseudocódigo

Algorithm 2: treinaModelo

$\langle s_t, a_t, r_t, s_{t+1}, fim \rangle \text{batch} \leftarrow \text{experienceReplay.obtemBatch}()$

if *fim* **then**

$y \leftarrow r_t$

else

$y \leftarrow r_t + \gamma \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$

end

Gradiente descendente

$\Delta\theta \leftarrow \frac{\partial}{\partial\theta}(y - Q(s, a))^2$

Atualiza os pesos da rede

Limitações

Integração do sistema com o jogo

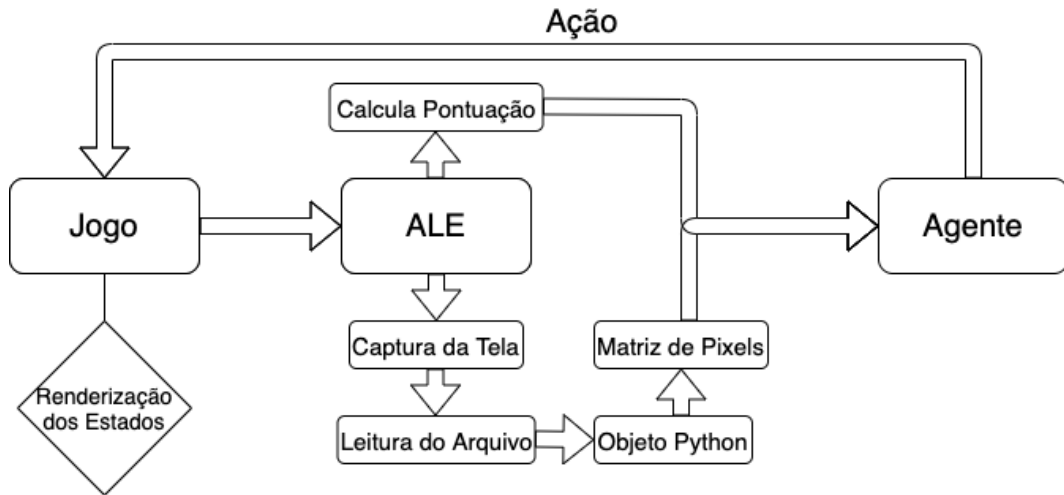
- ALE e rede neural implementados em Python
- Jogo consiste de um programa em C
- Dificulta a comunicação em tempo real
- Jogos Atari 2600 possuem um emulador em Python que fornece os dados do jogo sem a necessidade de renderizar as imagens (BROCKMAN et al., 2016)
- Com jogos em *Allegro* as imagens devem ser renderizadas

Limitações

Tempo de Treinamento

- Conjunto de dados a ser analisado é muito alto
- Imagens devem ser tratadas em cada instante de tempo
- Estado consiste de um vetor de tamanho $84 \times 84 \times 4 = 28224$
- Cada frame do jogo deve ser renderizada para que a ALE possa capturar as imagens em tempo real
- A rede pode levar dias ou semanas para convergir para uma política ótima

Limitações



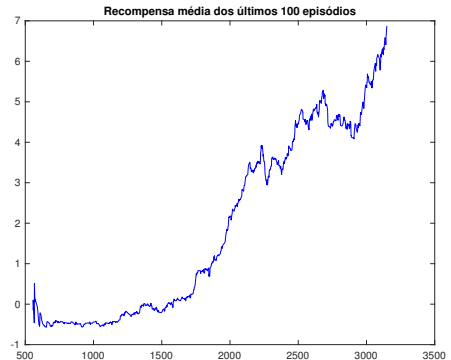
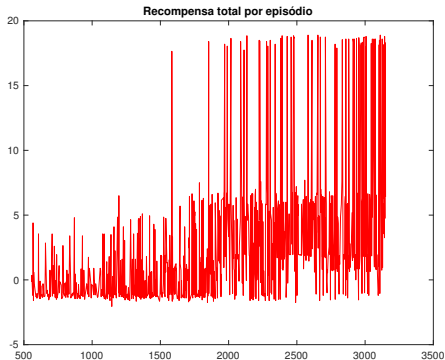
Limitações

Limitações de Hardware

- MacBook Air 2015
- Processador: 1.6 GHz Dual-Core Intel Core i5
- Memória: 4 GB 1600 MHz DDR3
- Sistema Operacional: macOS Catalina | Windows 10

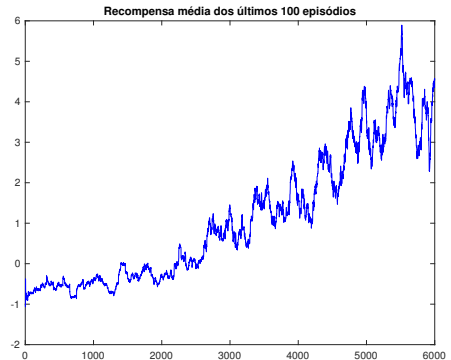
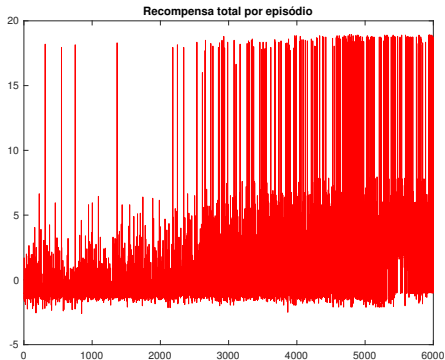
Resultados

Resultados



Resultados

Resultados



Conclusões

- Resultados obtidos são limitados mas promissores
- Agente foi capaz de encontrar uma política claramente superior à uma abordagem aleatória
- Dado um ambiente de treinamento mais propício, e com tempo de treinamento suficiente, espera-se que o sistema encontre resultados ainda melhores

Conclusões

- Resultados obtidos são limitados mas promissores
- Agente foi capaz de encontrar uma política claramente superior à uma abordagem aleatória
- Dado um ambiente de treinamento mais propício, e com tempo de treinamento suficiente, espera-se que o sistema encontre resultados ainda melhores


Limitações

- Comunicação entre o jogo e o ALE é limitado
- Aumenta o tempo de processamento dos dados
- Uma comunicação direta e em tempo real tornaria possível a implementação de soluções que poderiam otimizar o processo


Propostas de Continuidade


- Executar o sistema em um período suficientemente longo para obter um modelo com uma política equiparável à de um agente humano
- Desenvolvimento de uma comunicação direta em tempo real entre o ALE e o jogo
 - Cálculo da pontuação poderia ser feita pelo jogo e não pelo ALE
 - Possível implementação de uma solução que evite a renderização das imagens do jogo
 - Redução da margem para erros e possibilitando um sistema ainda mais genérico
- Aplicar o sistema para outros jogos com diferentes mecânicas de forma a avaliar melhor o potencial da rede implementada


Referências I

 BROCKMAN, G. et al. *OpenAI Gym*. 2016.

 HARGREAVES, S. *Allegro*. 1990. Disponível em: <<https://liballeg.org/index.html>>. Acesso em: 25 nov 2020.

 HASSELT, H. van; GUEZ, A.; SILVER, D. *Deep Reinforcement Learning with Double Q-learning*. 2015.

 MNIH, V. et al. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. Disponível em: <<http://arxiv.org/abs/1312.5602>>.

 SILVA, A. P. Ambiente para desenvolvimento de inteligência artificial em jogos allegro. Departamento de Ciência da Computação (DCC) da Universidade Federal de Minas Gerais (UFMG), Belo horizonte, Brasil, 2019. Disponível em: <https://github.com/artphil/allegro_game_ai>. Acesso em: 8 out 2019.