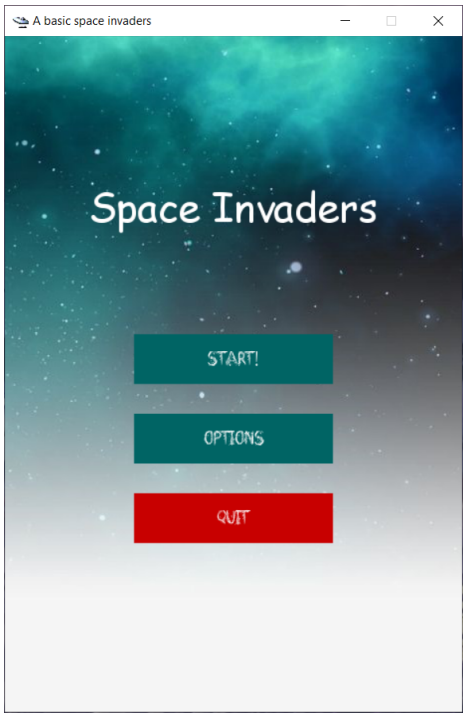
# Space Invaders - IA

Ivan de Jesus Pereira Pinto

2020-02-13

# Sumário

# Resumo


Neste relatório descrevemos as técnicas utilizadas nos agentes do jogo *space invaders*, além do processo de treinamento.

# Capítulo 1

# Introdução

Esse projeto visa o desenvolvimento de técnicas da área de Inteligência Artificial para o jogo de Space Invaders. Especificamente almeja-se a implementação de IAs para as naves inimigas. A maior parte da pesquisa nessa área se dá no desenvolvimento de Agentes que aprendem a jogar contra o ambiente, no estilo de um processo de decisão de Markov conforme ilustramos na Figura 1.1:

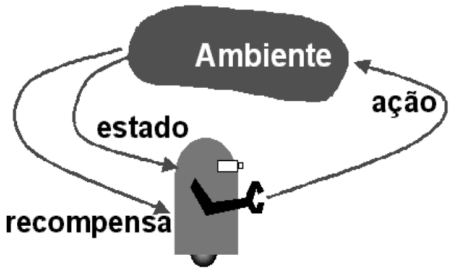



Figura 1.1: Processo de decisão de markov. [@faria1999explorando]

O SpaceInvaders aqui desenvolvido almeja no entanto politicas de controle para o ambiente, que seriam os inimigos. O jogador(humano) se torna o ambiente nesse caso. Para que fosse possível a implementação das técnicas de IA, foi necessário o desenvolvimento do jogo, de um simulador, e de um surrogate para o jogador. Os objetivos deste trabalho são listados a seguir:

1. Construção do jogo de Space Invaders eficiente em C.
2. Construção de um Forward Simulator
3. Desenvolvimento de um surrugate Player que substitua o humano em tempo de planning ou treino.
4. Implementação de Técnicas de IA (Planning e Aprendizado)

# Capítulo 2

# Desafio

O jogo de Space Invaders tem 2 principais desafios: o primeiro é, como vários jogos similares, o número enorme de estados possível, tornando inviável a utilização de métodos exatos ou programação dinâmica para computar a política ótima. O segundo problema, mais particular ao jogo, é o fato de que o agente controla as naves oponentes, aumentando assim o espaço de ações enormemente. Dado que nessa versão do jogo os oponentes tem 5 ações possíveis, 4 direções e tiro, e temos 3 oponentes, o número de ações possíveis vai ser $5 5 5 = 125$ ações. O numero pequeno de oponentes é proposital, pois o espaço de ações cresce muito, tornando inviável as técnicas aqui descritas. Versões mais eficientes estão em desenvolvimento.

No capítulo 3 descrevemos as técnicas utilizadas neste trabalho, e no capítulo 4 descrevemos a implementação como um todo.

# Capítulo 3

# Metodologia

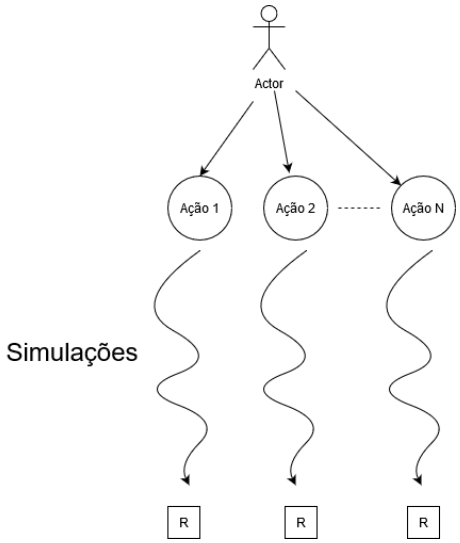
Descrevemos aqui os métodos utilizados neste projeto. Importante: a recompensa é +1 pra cada dano recebido pela nave do jogador( do ponto de vista dos oponente, os quais estamos tentanto otimizar) e -1 pra cada dano recebido pela nave oponente. Além disso, +100 é dado de o tiro inimigo matar o oponente, e -100 é recebido se uma nave inimiga morrer.

## 3.1 Neural Network

Uma rede neural de 1 camada escondida foi treinada com busca aleatória no espaço de pesos. O processo é muito simples, com a rede sendo avaliada em um número de partidas com os pesos gerados, e caso ela seja a melhor rede até agora( no acúmulo de recompensas), salvamos seus pesos e os modificamos em direções aleatória usando ruído gaussiano. O resultado foi aquém do esperado, talvez devido a complexidade do problema e a esparcidade das recompensas, que ocorrem com pouca frequência.

## 3.2 Monte Carlo 1-Step Planning

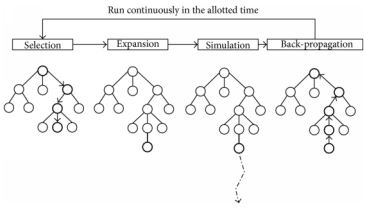
Utiliza-se simulação monte carlo para se estimar o retorno esperado de cada ação possível no estado atual. Rollouts/playouts/partidas aleatórias são simuladas até certo passo estipulado. A recompensa acumulada até aquele passo é utilizada como estimativa do retorno esperado. Quanto mais simulações melhor, logo é importante a velocidade do simulador. O funcionamento é ilustrado na figura abaixo:




Por causa do enorme número de ações, esse método se torna menos eficiente. Melhorias possíveis são a utilização de métodos que selecionam subconjuntos de ações disponíveis.

## 3.3 Monte Carlo Tree Search

O método de melhor resultado, é uma melhoria do MC onde se constroi uma árvore de ações na memória. Seu funcionamento se dá pelos 4 passos ilustrados na figura abaixo:




## 3.4 Reinforcement Learning

# Capítulo 4

# Aplicação

## 4.1 Instalaçao

Para instalar no Windows é necessário um compilador de c(gcc foi utilizado no desenvolvimento), e de python 3.6 de 32 bits. Utilizando o anaconda basta criar um ambiente como se segue: Colocar variável de ambiente: set CONDA_FORCE_32BIT=1
Criar ambiente: conda create -n py36_32 python=3.6 Ativar ambiente: conda activate py36_32 Instalar numpy e pygame: pip install numpy,pygame

As depedências do python devem ter sido satisfeitas. O programa com UI vai precisar chamar o jogo SpaceInvaders em C, para que possa funcionar. Temos que compilar o código em C com o comando a seguir, da pasta SpaceInvaders

gcc -Wall -fPIC -shared src/Space.c src/list.c src/genann.c -o src/Space.so

Se tudo tiver dado certo, o jogo deve funcionar. Basta utilizar python menu.py

## 4.2 Jogo

O Jogo Space invader foi desenvolvido em C de modo a ser o mais eficiente possível. Além disso, foram utilizadas estruturas de dados com menor complexidade, como por exemplo: 1. o uso de Matriz para representar o espaço do jogo, tendo O(1) para checar se a nave do jogador ou dos oponentes receberam dados dos tiros. 2. O(n) para mover os tiros, onde n é a quantidade de tiros ativos na tela, armazenados em uma lista. Tiros fora da tela são descartados Um foward simulator foi implementado para permitir que os agentes possam fazer simulações, para planning e treinamento. Os comandos para jogar são os seguintes: direcionais movem a nave, e espaço atira.

O Vídeo abaixo demonstra uma partida rapida de um humano contra a IA MCTS

# Capítulo 5

# Considerações finais

We have finished a nice book.