RL Aplicado ao Nethack Learning Environment

Lucas Antunes Sampaio, Kauan Torrezan Palazzi, Romulo Carneiro de Oliveira Canavesso, Leonardo Anjolleto Ferreira
Ciências da Computação Centro Universitário FEI
msrzkatakan@gmail.com, laferreira@fei.edu.br

Resumo: Este projeto usa Aprendizado por Reforço (RL) para treinar um agente a resolver o problema de Scout, que envolve navegar por níveis de um jogo e encontrar escadas enquanto minimiza mortes. O método Q-Learning foi aplicado para melhorar a tomada de decisão do agente. Cenários controlados no framework Minihack foram utilizados para testar e ajustar o desempenho do agente.

1. Introdução

Este projeto propôs a resolução do problema de *Scout*, apresentado no artigo [1], proposto em cima do jogo Nethack, um famoso jogo de RPG (*Role-Playing Game*) de aventura e exploração onde o jogador deve explorar a masmorra chamada Labirintos de Menace (*Mazes of Menace*), encontrar o Amuleto de Yendor (*Amulet of Yendor*) e então sair da masmorra, passando por várias adversidades como inimigos, armadilhas, fome e cansaço como descrito no artigo [1]. Por ser um jogo de RPG, o jogador pode escolher entre 13 classes diferente que impactam diretamente na jogabilidade, também descrito no artigo [1], podemos ter uma ideia de como o Nethack na imagem (1).

Exemplo de cenário criado no Minihack (1)

Este problema consiste em que o agente precisa aprender a se locomover por cada nível do jogo, de forma que não morra tão cedo no jogo, encontre a escada para o próximo nível ao mesmo tempo que explora o nível atual da melhor forma possível, como apresentado no artigo [2]. Realizamos um treinamento de um agente utilizando o método de aprendizagem por reforço (RL), além de utilizar a ferramenta Minihack, cuja qual é destinada ao RL e nos permite criar e testar diversas tarefas por meio da criação de cenários/fases controlados utilizando *assets* (elementos que compõe o jogo) do Nethack como descrito no artigo [6], desta forma, foram usados cenários baseados em elementos do próprio Nethack com a premissa de treinamento dos agentes, de forma a tornar tal feito mais fácil para o próprio agente no começo, com cenários fáceis e simples. Com a melhora dos resultados será possível dificultar ainda mais os cenários futuramente, com objetos que dificultam a exploração do agente e inimigos, como é possível visualizar na imagem (1) um exemplo de cenário inicial utilizado para os testes dos agentes.

2. Metodologia

Para trabalhar neste no problema de *Scout* e obter resultados interessante, utilizamos a técnica Aprendizado por Reforço (*Reinforcement Learning* ou RL) em especifico Q-Learning, que consiste em um aprendizado por iterações entre o ambiente/cenário e o agente, de forma que não seja necessário uma experiencia previa por parte do agente sobre o cenário, mas sim um sistema de recompensas (pontos) onde o agente pode ser recompensado ou punido em função das decisões tomadas por ele, desta forma o agente possui o aspecto de avaliar suas decisões a partir de decisões previamente tomadas, tornando o agente apto como já testado e apresentado no livro [3]. Para acelerar o aprendizado do agente em fases complexas foi implementado um método de Aprendizagem Aprofundada (*Deep Learning*) conhecido por Destilação Aleatória em Rede (*Random Network Distillation*), que consiste em uma aprendizagem que estimula o agente a explorar coisas novas em cada tentativa, utilizando de duas redes neurais: A rede alvo (que fixa configurações imutáveis por todo processo de treino do agente) e a rede preditora (que e treinada para adivinhar as saídas da rede alvo e tornar as suposições desta rede mais próximas das saídas de rede alvo). Desta forma a rede preditora estará aprendendo a partir das decisões do Q-Learning e dos dados coletados pelo agente e aproximando suas suposições a partir de um gradiente descendente, ao mesmo tempo que estimula o agente a explorar coisas novas no cenário e tomar decisões novas para que seja possível coletar mais dados e fazer mais aproximações precisas, como descrito no artigo [4], a RND é um recurso muito poderoso em um treinamento do agente em jogos de exploração (como o Nethack), desta forma, o RND complementa ainda mais o Q-Learning no aspecto de curva de aprendizagem e tomada de decisão do agente, mas ainda é necessário otimizar os resultados. Então utilizamos também o algoritmo RSMProp (*Root Mean Square Propagation*), um algoritmo que consegue ajustar a taxa de aprendizagem para cada parâmetro, ele também consegue normalizar os gradientes gerados, como visto no artigo [5]. Desta forma otimizamos os resultados de aprendizagem do agente possibilitando uma maior rapidez no aprendizado do agente por meio da otimização dos gradientes descendentes e da otimização das decisões tomadas pelo agente.

3. Desenvolvimento

Utilizando o Minihack, foram criados cenários controlados para os testes de agentes sem treinamento, foram testados 10 agentes com os seguintes parâmetros para todos eles: “learning_rate” (diz respeito sobre as atualizações dos Q valores sendo ajustadas a cada iteração), “gamma” (diz respeito sobre o peso que o agente da a recompensas futuras), “epsilon” (diz respeito a probabilidade do agente de explorar ações novas e desconhecidas a fazer o que já conhece), “epsilon_decay” (diz respeito a taxa de decaimento da probabilidade de explorar novas ações), “epsilon_min” (diz respeito a um limite inferior para os decaimentos epsilon) e “num_episodes” (diz respeito ao número de episódios que serão realizados no treinamento), as recompensas são representadas por meio de um sistema de pontuação, onde, ações como: sobreviver por mais tempo, evitar obstáculos, evitar interações desnecessárias, completar o nível de maneira rápida, são recompensadas com uma pontuação positiva, enquanto ações como: morrer, não evitar obstáculos, tomar muito tempo para sair do nível atual, realizar ações desnecessárias, são punidas com pontuação negativa.

Tabela I – Valor dos Parâmetros nos 3 Treinamentos

Parâmetros	Valores 1	Valores 2	Valores 3
learning_rate	0.1	0.3	0.1
gamma	0.99	0.99	0.95
epsilon	1.0	1.0	1.0
epsilon_decay	0.995	0.995	0.995
epsilon_min	0.01	0.01	0.01
num_episodes	100	100	100

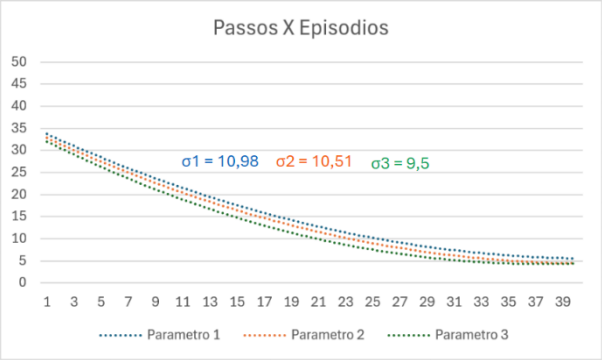

Figura I – Gráfico de Passo por Episódios

7. Conclusões

Foi possível observar a partir dos testes que, os agentes que foram treinados a partir dos parâmetros 1, tiveram um aprendizado regular comparado aos demais, os agentes que foram treinados a partir dos parâmetros 2 tiveram um aprendizado semelhante porem um pouco melhor comparado aos do parâmetro 1, já os que treinaram a partir dos parâmetros 3 foram os que conseguiram aprender mais rápido e estabilizaram antes, desta forma podemos concluir que, ao mudar o learning_rate e o gamma foi possível obter resultados melhores, em ênfase o gamma, o que mostra que quanto mais o agente aprender no inicio da simulação, melhor ele desempenhara

10. Referências

[1] RAYMOND, Eric S.; STEPHENSON, Mike et al. A Guide to the Mazes of Menace: Guidebook for NetHack. NetHack DevTeam, 2020

[2] HEINRICH KÜTTLER et al. The NetHack Learning Environment. arXiv (Cornell University), 24 jun. 2020.

[3] SUTTON, R. S. Reinforcement Learning, Second Edition: An Introduction: An Introduction. Cambridge, Massachusetts: The Mit Press, 2018.

[4] BURDA, Yuri; OPENAI, Harrison; EDWARDS; OPENAI, Amos; STORKEY; OPENAI, Oleg Klimov. EXPLORATION BY RANDOM NETWORK DISTILLATION. 2023.

[5] ELSHAMY, R. et al. Improving the efficiency of RMSProp optimizer by utilizing Nestrove in deep learning. Scientific Reports, v. 13, n. 1, p. 8814, 31 maio 2023.

[6] MIKAYEL SAMVELYAN et al. MiniHack the Planet: A Sandbox for Open-Ended Reinforcement Learning Research. arXiv (Cornell University), 27 set. 2021.