Maximização de pontuação no jogo Freeway utilizando os algoritmos de aprendizado por reforço Proximal Policy Optimization (PPO) e Deep Q-Learning (DQN)

Vinicius Alves Matias[1]
[1]Escola de Artes, Ciências e Humanidades da Universidade de São Paulo
Rua Arlindo Bettio, 1000
Vila Guaraciaba, São Paulo/SP
viniciusmatias@usp.br

Abstract

The discussion and evaluation of algorithms for planning and reinforcement learning can be applied to different problems in the literature. One of the ways of verifying the performance of these algorithms is by applying them to games, and in the case of this project, by the Atari 2600 Freeway game. This proposal starts by introducing the problem, followed by an analysis of the possible discretization of the environment to facilitate understanding and evaluation of algorithms for planning. The focus of the work will be discussed next, proposing the use of two reinforcement learning algorithms present in the literature for a later comparison of performance between them.

Introdução

Freeway é um jogo desenvolvido pela Activision e disponibilizado no Atari 2600 (Activision 1981). O objetivo do jogo é fazer uma galinha atravessar uma via expressa enquanto desvia de veículos, acumulando um ponto ao atravessar todas as faixas sem ser atingida. A pontuação máxima possível é igual à 34 pontos. Quando a galinha encontra um carro, ela retorna à posição de início, seguindo a implementação original do jogo. A interface do game original pode ser vista na figura 1.

Pela análise da interface nota-se que o jogo permite dois usuários, contudo, para o desenvolvimento e análise dos algoritmos bastará usar apenas um. Importante notar que os carros se movimentam horizontalmente e as galinhas verticalmente. Os algoritmos que serão utilizados buscarão aumentar a pontuação que é exibida na barra superior.

Para o desenvolvimento dos algoritmos, esse projeto irá se basear no ambiente desenvolvido pela OpenAI Gym (Bellemare et al. 2012). As duas diferenças relevantes entre o jogo original e esta versão são a redução do eixo horizontal e a utilização de apenas um carro por faixa. O projeto terá foco em aprendizado por reforço e utilizará os pixeis da tela como entradas dos algoritmos (210px na vertical e 160px na horizontal, compondo uma matriz 33600px para a interface do jogo). Importante notar para a formulação do processo estocásticos temos:

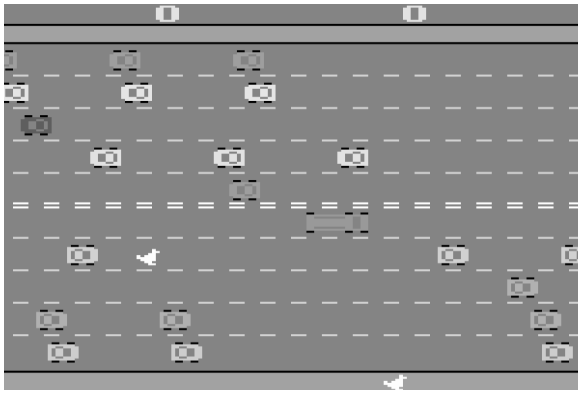

Figure 1: Interface do jogo Freeway original. A imagem foi capturada utilizando o emulador Stella.

- 1 Agente: Representado pela galinha
- 3 Ações: Mover para frente, mover para trás e não se mover
- 3^{33600} estados possíveis: o agente pode realizar uma de 3 ações dentro do ambiente de 33600px. Contudo, dado um estado s_t o ambiente transita para um estado s_{t+1} baseado apenas na configuração corrente do ambiente (isto é, a configuração dos 33600px da vez, assim, é necessário definir uma política markoviana).

Ainda que o enfoque seja em aprendizado por reforço, será dispendido certo tempo em planejamento para discretizar o ambiente e avaliar políticas ótimas na formulação dos processos.

Planejamento Probabilístico

Visando reduzir a quantidade de estados possíveis para conseguir estudar uma política ótima do problema, será proposta a reconstrução do problema em uma matriz $2X2$. Durante o desenvolvimento, é possível que seja mais viável utilizar apenas uma linha ou apenas uma coluna, mas a proposta inicial tentará manter esta manobra. Para formulação do processo markoviano de decisão (do inglês MDP) do

ambiente discretizado podemos considerar uma tupla $< S, A, D, T, R >$ (Mausam and Kolobov, 2012), onde:

- $s \in S$ são os estados, resultando em $3^4 = 81$ estados possíveis. Note que o estado inicial s_0 é fixo e há um conjunto de estados metas G em que o agente chega ao fim da via expressa;
- $a \in A$ são as ações, tendo um total de 3 ações possíveis. A pode ser representado como um conjunto $A = \{UP, DOWN, STOP\}$;
- $d \in D$ é o tempo (ou época) que ocorre a decisão. Visto que há um limite de tempo para finalizar o jogo igual à 2 minutos e 16 segundos (Weiss 2007), D é um conjunto finito (isto é, não poderão ser mapeados estados inifinatamente);
- T é uma função de transição que mapeia os estado corrente, época e ação em um estado s_{t+1}. A função de transição segue uma política, tal passo será comentado na menção aos algoritmos;
- R é uma função de recompensa. À priori, o mapeamento da função pode seguir o resultado: -1 (agente bate com o veículo), 1 (Agente chega ao fim da via expressa) e 0 (agente transitando entre estados).

Visto que queremos maximar a pontuação e a formulação do jogo, devemos buscar uma política π markoviana (como mencionado na introdução), estacionária (não aparenta haver relação entre tempo e transição/recompensa) e determinista (sempre será feita a mesma ação para um mesmo estado).

O primeiro algoritmo que se planeja utilizar para o problema discreto é o de iteração de política (Policy Iteration). Sendo uma mescla dos algoritmos de avaliação e melhoria de política, podemos por meio deste identificar a convergência para a política ótima em um ambiente discreto - com número finito de políticas (Sutton and Barto 2020). Em cada iteração será realizada a avalição com base nas 3 ações, tendo, dada a configuração do MDP e da política, 3 estados possíveis por vez para manipular. O algoritmo deverá buscar as recompensas positivas manipulando a política até chegar um momento que não haja diferença no valor da política entre épocas. Se necessário, para o projeto final o ambiente discreto será reduzido mais ainda para conseguir executar este algoritmo, como lembrado anteriormente.

Outro algoritmo que pode ser usado para trazer comparações interessantes de iteração de valor (Value Iteration). Com ele podemos chegar próximo à uma política ótima, dado que estabelecemos um limiar que definirá a acurácia do método. Tal algoritmo se baseia na equação de Bellman e consegue determinar uma política com um menor nível de acurácia, mas tendendo a gastar menos iterações que o algoritmo de iteração de políticas.

Aprendizado por Reforço

Muitos pontos importantes para entendimento do problema foram mencionados no tópico de Planejamento Probabílistico para que possamos estender a discussão ao enfoque do projeto: Aprendizado por Reforço. Dado a configuração do problema mencionada na introdução, temos a possibilidade de usar dois algoritmos para maximizar a pontuação no jogo Freeway: a primeira utilizando o algortimo Proximal Policy Optimization (PPO) e a segunda utilizando Deep Q-Learning (DQN). O algoritmo Asynchrounous Actor-Critic Agent (A3C) também mostrou-se interessante e talvez possa ser utilizado durante o projeto no caso de mostrar-se realmente relevante, contudo, os outros dois algoritmos demonstram-se durante o estudo tão bem adequados ao problema quanto o A3C.

O algoritmo PPO (Schulman et al. 2017) utiliza alguns conceitos fundamentais para definir a política. O primeiro é o conceito de Advantage $\hat{A}_t$ que consiste, basicamente, em uma estimativa da qualidade de uma ação em um determinado estado comparada à ação média para este estado. Este conceito é utilizado para tomar uma ação suficientemente segura no problema, dada a configuração do estado. O algoritmo toma o cuidado de se pensar em cada estado e na ação a ser tomada como suficientemente segura, mas baseia-se não somente na política corrente, mas também na política anterior. Dado que tem essas duas políticas, o algortimo calcula a razão $r_t(\theta)$ entre a política atual $\pi_\theta(a_t|s_t)$ e a política anterior $\pi_{\theta k}(a_t|s_t)$. Essas conceitos são mesclados em uma função que define uma estimativa $L^{CLIP}(\theta)$ de quão distante está a política nova da política antiga, reduzindo essa diferença na tomada das ações. A função é calculada como $L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$, onde $clip$ é o responsável por fechar o intervalo, e ϵ é um valor baixo, normalmente 0.2. O processo pode ser repetido para um determinado número de épocas para aplicar o novo valor de θ baseado em L sob o θ_{old}. Essa redução para segurança na transição das políticas pode, por exemplo, melhorar as técnicas para desviar de obstáculos tomadas nas políticas.

O algoritmo DQN é outra técnica relativamente nova (Mnih et al. 2013), consistindo em utilizar redes neurais (por vezes, convolucionais) para otimizar as escolhas de ações sob políticas em aprendizado por reforço. A utilização de redes neurais torna-se de grande ajuda em problemas de jogos por já terem diversas técnicas estudadas aplicando imagens (logo, pixeis) para solucionar problemas que seriam mais complicados de se fazer apenas com aprendizado por reforço. Para tal, o algoritmo utiliza a ideia de Q-Learning (uma função que mapeia estados e ações para determinar qual seria a recompensa se escolhe-se cada ação em um espaço). Assim, busca-se uma política com a ação que maximize Q, como a função: $\pi^*(s) = argmax_a Q^*(s, a)$. Para encontrá-la no mundo real, a rede neural teria como entrada os dados do ambiente (pixeis) e como saída os valores de Q. Esses valores de Q da rede neural seriam comparados com os valores de Q anteriores, tiraria-se a soma do quadrado das diferenças como a função de custo e seria repassado para a camada de entrada (backpropagation) para ajustar o aprendizado.

Por fim, busca-se para o projeto final realizar a proposta aqui mencionada com o mínimo de adaptações durante o desenvolvimento do projeto.

Referencias

Activision, Inc. 1981. Atari 2600 Instructions Archive. Santa Clara, CA. Activision AG-009-03 Rev. 2.

Bellemare, MG.; Naddaf, Y; Veness, J; and Bowlingm. 2013. The Arcade Learning Environment: An Evaluation Platform for General Agents. Journal of Artificial Intelligence Research 47: 253-279.

Mausam and Kolobov, A. 2012. Planning with Markov Decision Processes: An AI Perspective. Morgan and Claypool Publishers.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M. 2013. Playing Atari with Deep Reinforcement Learning.

Schulman, J.; Wolski, F.; Dhariwal, P.; and Radford, A. and Klimov, O. 2017. Proximal Policy Optimization Algorithms. CoRR, abs/1707.06347.

Sutton, R. S.; and Barto, A. G. 2020. Reinforcement Learning: An Introduction. The MIT Press.

Weiss, B. Classic Home Video Games, 1972-1984: A Complete Reference Guide. 2007. McFarland and Company.