PROJETO 4

Eduarda Graziele de Paiva
cc22125@g.unicamp.br

Julia Enriquetto de Brito
cc22139@g.unicamp.br

1 Introdução

Nim é um jogo de estratégia onde dois jogadores alternam-se para remover qualquer quantidade de objetos de uma única pilha, evitando remover o último objeto para não perder. A complexidade do jogo aumenta com o número de pilhas.

Neste projeto, uma IA foi desenvolvida para aprender a estratégia ideal de Nim usando aprendizado por reforço, especificamente com os algoritmos SARSA e Q-Learning, que associam recompensas a pares de estado e ação. A IA joga contra si mesma repetidamente, aprendendo quais ações levam à vitória ou à derrota. As recompensas são atribuídas conforme o resultado: -1 para ações que levam à derrota, 1 para ações que forçam a derrota do adversário e 0 para ações neutras.

2 Aprendizado por Reforço

O aprendizado por reforço (Reinforcement Learning - RL) é uma área do aprendizado de máquina que ensina agentes a tomarem decisões em ambien-

tes dinâmicos, ajustando seu comportamento para maximizar recompensas acumuladas. O agente interage com o ambiente por meio de ações, recebendo feedback em forma de recompensas ou penalidades.

Cada estado (S) representa uma configuração do ambiente, e as ações (A) são escolhas que o agente pode fazer. Após uma ação, o ambiente retorna uma nova situação e uma recompensa (R), indicando o sucesso da decisão.

O objetivo principal é desenvolver uma política que determine as melhores ações para cada estado, maximizando a soma das recompensas futuras. Essa soma é ajustada por um fator de desconto (γ), que prioriza recompensas imediatas sem ignorar o impacto de longo prazo. [3]

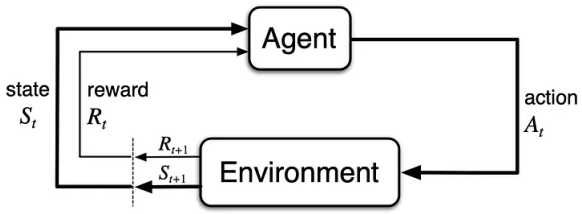

Figura 1: Diagrama de funcionamento do aprendizado por reforço
Fonte: Sutton, R; Barto, A. (2020) [3]

2.1 SARSA (State Action Reward State Action)

SARSA (State Action Reward State Action) é um algoritmo de aprendizado por reforço que ensina agentes a tomar decisões em ambientes dinâmicos. Diferente do Q-learning, SARSA atualiza os valores de acordo com a política atual do agente, seguindo o princípio de aprendizado on-policy, onde o agente aprende com suas próprias experiências diretas.

O nome SARSA vem da sequência de elementos que o algoritmo utiliza para atualizar seus aprendizados:

- State (S): Estado atual;
- Action (A): Uma ação escolhida no estado atual (usando a política do agente);
- Reward (R): Recompensa recebida após tomar a ação A;
- Next State (S'): Próximo estado alcançado após a ação A;
- Next Action (A'): Próxima ação escolhida no estado S', também seguindo a política do agente.

Esses elementos (S, A, R, S', A') são usados para atualizar os valores Q(s,a), que representam as estimativas do valor esperado de realizar uma ação em um estado específico. A atualização segue a fórmula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R + \gamma Q(s',a') - Q(s,a)\right]$$

Onde, α é a taxa de aprendizado que controla a velocidade de atualização, γ é o fator de desconto, que pondera a importância de recompensas futuras. [1]

2.2 Q-Learning

O Q-Learning é um algoritmo de aprendizado por reforço sem modelo, usado para encontrar a melhor política de ações em um ambiente. Ele se baseia nos **valores Q** ($Q(s,a)$), que representam a recompensa esperada ao realizar uma ação a em um estado s.

2.2.1 Funcionamento

- O agente interage com o ambiente, realizando ações e recebendo recompensas.

- Atualiza os valores $Q(s, a)$ usando a fórmula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(R + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \qquad (1)$$

  Onde:

  - α: Taxa de aprendizado.
  - γ: Fator de desconto para recompensas futuras.
  - $\max_{a'} Q(s', a')$: Melhor estimativa para o próximo estado.

- A política ϵ-gananciosa equilibra **exploração** (ações aleatórias) e **exploração** (ações ótimas).

2.2.2 Vantagens

- Funciona sem conhecimento prévio do ambiente.
- Converge para a política ótima em muitos cenários.

2.2.3 Limitação

Espaços muito grandes podem exigir extensões, como Deep Q-Learning. [2]

3 Experimentos computacionais

Todos os experimentos computacionais foram realizados em um PC com processador Apple M1, 16 GB de memória RAM. Os modelos matemáticos foram implementados em Python 3.12.1.

4 Resultados

Os experimentos realizados com os algoritmos SARSA e Q-Learning para o jogo Nim foram comparados com diferentes números de episódios de treinamento. A Tabela 1 apresenta os resultados obtidos.

N° Episódios	**SARSA**	**Q-Learning**
100	3	2
150	2	3
500	0	5
1000	0	5

Tabela 1: Desempenho dos algoritmos de aprendizado

5 Conclusão

Com base nos resultados apresentados, o algoritmo "Q-Learning"demonstrou maior eficiência no jogo Nim em comparação ao algoritmo "SARSA"no contexto de aprendizado por reforço. Isso ocorre devido à forma como o Q-Learning escolhe ações futuras, priorizando sempre a ação ótima, ao contrário do SARSA. Dessa forma, conclui-se que, para um jogo como o Nim, o Q-Learning é a abordagem mais eficiente.

Referências

[1] FOR GEEKS, G. Reinforcement learning in python: Implementing sarsa agent in taxi environment, 2024.

[2] GEEKS FOR GEEKS. Q-learning in python, 2024.

[3] SUTTON, R; BARTO, A. Reinforcement learning, an introduction - second edition, 2020.