Projeto 4

Pedro Nunes - cc22147@g.unicamp.br

Saulo Nathan - cc22149@g.unicamp.br

1. Introdução:

Sobre o Jogo:

O jogo Nim é um jogo de estratégia em que dois jogadores se alternam para remover objetos de diferentes pilhas. O objetivo é evitar ser o jogador que remove o último objeto. Este jogo tem relevância significativa como problema de aprendizado por reforço, pois envolve tomada de decisão em um ambiente com múltiplos estados possíveis e o uso de táticas para vencer.

Algoritmos Utilizados: SARSA e Q-Learning:

- **SARSA (State-Action-Reward-State-Action)**: Um algoritmo **On-Policy** que atualiza os valores Q (recompensas) com base em ações realmente tomadas por ele.
- **Q-Learning**: Um algoritmo **Off-Policy** que atualiza os valores de Q (recompensas) com base na melhor ação possível para o próximo estado, independentemente da ação tomada.

Objetivos do Projeto:

- Construir uma IAs capazes de aprender estratégias otimizadas para o jogo Nim.
- Comparar o desempenho das implementações dos algoritmos SARSA e Q-Learning.

---

2. Metodologia:

Descrição do Ambiente de Jogo:

- **Regras do Nim**: O jogo começa com pilhas contendo um número específico de objetos. Os jogadores alternam turnos e podem remover qualquer

quantidade de objetos de uma única pilha em cada jogada. O jogador que remove o último objeto perde.

- **Estados e Ações**:
  - **Estado**: Representado pela configuração das pilhas (por exemplo, `[1, 3, 5, 7]`).
  - **Ação**: Representada por uma tupla `(i, j)`, onde `i` é a pilha escolhida e `j` o número de objetos removidos.

Implementação:

As principais funções implementadas incluem:

- `update_model`: Atualiza o valor Q com base na recompensa recebida após uma ação.
- `get_value`: Retorna o valor Q associado a um par (estado, ação). Se não existir, retorna zero.
- `update_value`: Recalcula o valor Q para um par (estado, ação) considerando a recompensa e o valor futuro esperado.
- `best_future_reward` (apenas Q-Learning): Determina a maior recompensa possível para um estado futuro.
- `choose_action`: Seleciona a próxima ação a ser tomada, com base em uma política ε-greedy.

Estratégias de Treinamento:

- **Parâmetros de aprendizado**:
  - Taxa de aprendizado (α) = 0,5

    α\alpha
  - Taxa de exploração (ϵ) = 0,1

    ϵ\epsilon
  - Fator de desconto (γ) = 0,9

    γ\gamma

  Foram utilizados estes valores, pois foram os que apresentaram os resultados mais satisfatórios e estáveis.

- **Número de simulações**: Foram realizados **1000 treinamentos** para cada algoritmo, garantindo aprendizado suficiente para observar convergência de desempenho e sendo realizados ainda de forma rápida.

---

3. Resultados:

Desempenho da IA:

- **Comparação entre SARSA e Q-Learning**:
  - **Taxa de vitórias**: O Q-Learning demonstrou melhor desempenho em cenários complexos, vencendo a maioria das partidas contra o SARSA e estratégias básicas.
  - **Tempo de aprendizado**: O Q-Learning convergiu mais rapidamente, enquanto o SARSA apresentou um aprendizado mais estável.

Análise das Estratégias:

- **SARSA**: Desenvolveu estratégias mais conservadoras, focando em evitar situações de risco imediato.
- **Q-Learning**: Explorou ações mais agressivas, com foco em maximizar a recompensa futura, mesmo que isso envolvesse riscos momentâneos.

---

4. Discussão:

Aprendizado e Comportamento:

- **Impacto dos Parâmetros**: Valores de α e γ influenciaram diretamente a capacidade de exploração e convergência das IAs. Parâmetros mal configurados resultaram em aprendizado inconsistente.

Comparação com Estratégias Humanas:

- **Semelhanças**: Em situações simples, com pouco treinamento, as estratégias das IAs se assemelharam às humanas, especialmente quando a IA precisou forçar o oponente a pegar o último objeto.
- **Diferenças**: O Q-Learning, em particular, explorou jogadas menos óbvias, que podem ser contraintuitivas para jogadores humanos.

---

5. Conclusão:

Este projeto nos demonstrou a aplicação prática dos algoritmos SARSA e Q-Learning em um jogo de estratégia e tomada de decisão, destacando suas diferenças e eficiência. O Q-Learning apresentou melhor desempenho geral, mas o SARSA mostrou-se útil em outros cenários. A exploração eficiente e o ajuste de hiperparâmetros foram cruciais para o sucesso do aprendizado.

---

6. Referências:

- https://www.datacamp.com/pt/tutorial/introduction-q-learning-beginner-tutorial
- https://www.freecodecamp.org/portuguese/news/uma-introducao-ao-q-learning-aprendizagem-com-reforco/
- https://www.datacamp.com/tutorial/sarsa-reinforcement-learning-algorithm-in-python
- https://builtin.com/machine-learning/sarsa