Análise de Desempenho Comparativa: Uma Confrontação entre os Algoritmos Minimax e Q-Learning no Jogo da Velha

Maurício Sarpa
Engenharia da Computação
Fundação Hermínio Ometto (FHO)
Araras, São Paulo, Brasil
sarpa@alunos.fho.edu.br

Pablo Novais Ramos
Engenharia da Computação
Fundação Hermínio Ometto (FHO)
Araras, São Paulo, Brasil
pablo@alunos.fho.edu.br

***Abstract*—Este artigo detalha uma análise de desempenho comparativa entre dois paradigmas de Inteligência Artificial (IA), confrontando um algoritmo determinístico clássico, o Minimax, com um método de aprendizado por reforço, o Q-learning. Ambos os algoritmos foram implementados como agentes autônomos para o Jogo da Velha, um ambiente canônico para a experimentação em IA devido ao seu espaço de estados bem definido. Desenvolvidos em Python, os agentes foram submetidos a uma série de confrontos diretos para quantificar e comparar suas métricas de desempenho. Os resultados validam a expectativa teórica: o Minimax atua como um benchmark imbatível de jogo perfeito, enquanto o agente Q-learning, após um treinamento adequado, converge com sucesso para uma política ótima, capaz de empatar consistentemente contra seu adversário perfeito. A discussão do trabalho contrasta a lógica codificada e a busca exaustiva do Minimax com a estratégia emergente que o Q-learning adquire através da experiência, oferecendo insights sobre a aplicabilidade de cada abordagem em diferentes domínios de problemas.**

***Index Terms*—Inteligência Artificial, Jogo da Velha, Minimax, Q-learning, Aprendizado por Reforço, Teoria dos Jogos.**

I. Introdução

O campo da Inteligência Artificial em jogos oferece um rico cenário para o estudo de algoritmos de tomada de decisão. Dentro desse domínio, jogos de baixa complexidade como o Jogo da Velha (Tic-Tac-Toe) funcionam como problemas canônicos, permitindo isolar e analisar o comportamento de diferentes agentes em um ambiente controlado e com regras bem estabelecidas [1], [2].

Este trabalho contribui para essa área através da implementação e análise empírica de dois agentes distintos para o Jogo da Velha. O primeiro é fundamentado no algoritmo Minimax, um pilar da teoria dos jogos que opera com base em busca e informação perfeita. O segundo utiliza Q-learning, uma técnica fundamental de aprendizado por reforço que permite ao agente aprender uma estratégia ótima sem conhecimento prévio do modelo do ambiente. O objetivo central é contrastar essas duas abordagens, avaliando como seus processos de decisão, métricas de performance e características computacionais se manifestam em um ambiente de confronto direto.

II. Metodologia

A abordagem metodológica do projeto foi dividida em três fases: a seleção e preparação do ambiente de jogo, a implementação detalhada de cada um dos agentes e, por fim, a execução de um procedimento experimental para a coleta e análise de dados.

A. Ambiente Experimental: O Jogo da Velha

Para este estudo comparativo, o Jogo da Velha foi selecionado como arena experimental por três motivos estratégicos:

1) **Espaço de Estados Gerenciável:** O jogo possui um número finito e tratável de configurações de tabuleiro (765 estados únicos, excluindo-se simetrias), o que torna viável a exploração completa da árvore de jogo pelo Minimax [3].
2) **Viabilidade para Q-learning Tabular:** Sua natureza discreta e contida é ideal para uma implementação tabular de Q-learning, na qual a tabela de valores (Tabela Q) pode ser armazenada integralmente na memória. Isso permite uma análise pura do algoritmo, sem a necessidade de técnicas de aproximação de função [1].
3) **Alinhamento com as Premissas Teóricas:** Por ser um jogo determinístico, de informação perfeita e soma zero, ele satisfaz todas as condições teóricas para a aplicação ótima do Minimax, estabelecendo-o como um benchmark de performance perfeito [2].

B. Agente 1: Abordagem Determinística com Minimax

O primeiro agente foi construído sobre o algoritmo Minimax, um pilar da IA para jogos de dois oponentes. Sua lógica é recursiva e baseia-se em antecipar as jogadas do adversário para minimizar a perda máxima possível. O algoritmo opera construindo uma árvore de jogo a partir do estado atual. Aos nós terminais (folhas da árvore), que representam fins de partida, são atribuídos valores de utilidade: +1 para uma vitória do nosso agente (MAX), -1 para uma derrota e +1 para empates.

Esses valores são então retropropagados pela árvore, permitindo que o agente MAX escolha sempre o caminho que leva ao resultado de maior valor, sob a premissa de que o oponente MIN sempre escolherá o caminho de menor valor.

C. Agente 2: Abordagem Adaptativa com Q-learning

Em contraste direto com a abordagem do Minimax, o segundo agente utiliza Q-learning, um algoritmo de aprendizado por reforço *model-free*. Ele aprende a jogar através da interação direta com o ambiente, guiado por um sistema de recompensas e punições. O objetivo do agente é aprender uma função de valor estado-ação, $Q(s, a)$, que estima a recompensa futura esperada ao se tomar a ação a no estado s.

Esses valores são armazenados em uma Tabela Q e atualizados iterativamente. A regra de atualização é uma aplicação direta da Equação de Bellman, o pilar da programação dinâmica [4], adaptada para o contexto do aprendizado por reforço [2]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (1)$$

Para equilibrar a necessidade de explorar novas jogadas e a de utilizar o conhecimento já adquirido, foi implementada a estratégia *epsilon-greedy* (ϵ-greedy).

D. Procedimento Experimental

A avaliação comparativa foi conduzida por meio de um torneio automatizado, no qual o agente Minimax foi colocado para jogar 5.000 partidas contra o agente Q-learning (após este ter concluído sua fase de treinamento). As métricas de vitórias para cada agente e o número de empates foram registradas para análise estatística.

E. Implementação Detalhada

Para ilustrar os principais componentes da implementação dos agentes, são apresentados a seguir trechos de código relevantes, extraídos dos arquivos do projeto [5].

1) Lógica do Jogo: `check_winner()`: A verificação de vitória é feita de forma sistemática, checando todas as 8 combinações possíveis.

```python
# Em game/tic_tac_toe.py
def check_winner(self):
    """Verifica o estado do jogo."""
    wins = [
        (0, 1, 2), (3, 4, 5), (6, 7, 8),  # linhas
        (0, 3, 6), (1, 4, 7), (2, 5, 8),  # colunas
        (0, 4, 8), (2, 4, 6)              # diagonais
    ]
    for i, j, k in wins:
        if self.board[i] and self.board[i] == self.board[j] == self.board[k]:
            return self.board[i]

    if self.is_full():
        return 'Draw'

    return None
```

Fig. 1. Trecho de código da função `check_winner()`.

2) Agente Minimax: A Recursão: O núcleo do Minimax é uma função recursiva que alterna entre maximizar a pontuação da IA e minimizar a pontuação do oponente.

```python
# Em agents/minimax_agent.py
def minimax(self, board: TicTacToe, depth: int, is_maximizing: bool) -> int:
    """Função recursiva Minimax."""
    score = self.evaluate(board)
    if score is not None:
        return score

    if is_maximizing:
        best_val = -math.inf
        for move in board.get_available_moves():
            board.make_move(move, self.ai_player)
            val = self.minimax(board, depth + 1, False)
            board.board[move] = None  # desfaz movimento
            best_val = max(best_val, val)
        return best_val
    else:
        best_val = math.inf
        for move in board.get_available_moves():
            board.make_move(move, self.human_player)
            val = self.minimax(board, depth + 1, True)
            board.board[move] = None  # desfaz movimento
            best_val = min(best_val, val)
        return best_val
```

Fig. 2. Trecho de código da função `minimax()`.

3) Agente Q-Learning: A Equação de Aprendizado: A atualização da Tabela Q é feita usando a equação de Bellman, que combina a recompensa imediata com a melhor recompensa futura esperada.

```python
# Em agents/qlearning_agent.py
def update_q(self, state: tuple, action: int, reward: float, next_state: tuple, next_moves: list):
    key = self._state_key(state)
    # ...
    old_q = self.q_table[key].get(str(action), 0.0)
    future_q = 0.0
    if next_moves:
        future_q = max(self.get_q(next_state, a) for a in next_moves)

    # Equação de Bellman
    new_q = old_q + self.alpha * (reward + self.gamma * future_q - old_q)
    self.q_table[key][str(action)] = new_q
```

Fig. 3. Trecho de código da função `update_q()` do agente Q-Learning.

III. Resultados e Discussão

Os resultados empíricos obtidos no torneio confirmaram as predições teóricas sobre o comportamento de cada agente. O agente Minimax estabeleceu-se como um jogador perfeito, permanecendo invicto em todas as 5.000 partidas, seja vencendo ou forçando o empate.

O desempenho do agente Q-learning, por sua vez, demonstrou uma clara curva de aprendizado. Após o treinamento, o agente foi capaz de convergir para a política ótima do Jogo da Velha, conseguindo forçar o empate contra o adversário perfeito em todas as ocasiões. Isso valida a capacidade do aprendizado por reforço de descobrir estratégias ótimas a partir da experiência.

Para ilustrar o processo de treinamento do agente Q-learning, a Figura 4 apresenta as métricas de desempenho (vitórias, empates e derrotas) e o valor de ϵ ao longo dos episódios de treinamento.

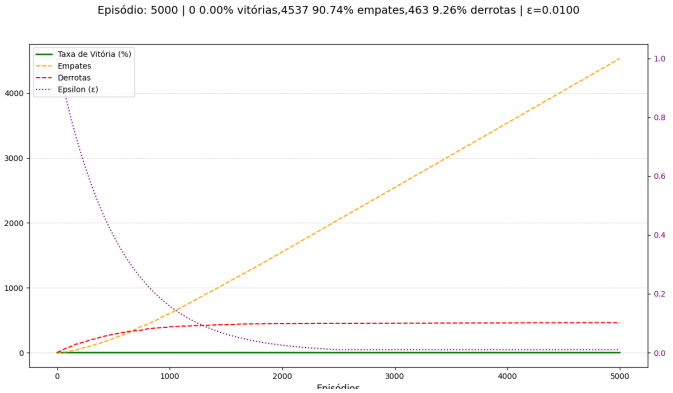

Fig. 4. Desempenho do Agente Q-learning durante o treinamento. A linha verde representa a taxa de vitória, a laranja os empates, a vermelha as derrotas e a roxa tracejada o decaimento de ϵ.

Como pode ser observado na Figura 4, a taxa de vitórias (linha verde) permanece em 0% quando o Q-learning enfrenta um agente ideal (ou joga contra si mesmo no início), o que é esperado para o Jogo da Velha onde um jogador perfeito pode sempre forçar um empate ou vitória. No entanto, o mais relevante é o aumento progressivo do número de empates (linha laranja tracejada) e a diminuição das derrotas (linha vermelha tracejada) ao longo dos 5000 episódios de treinamento. Isso indica que o agente Q-learning está aprendendo a evitar perdas e a alcançar resultados ótimos (empates contra um adversário perfeito). A linha pontilhada roxa representa o decaimento do parâmetro ϵ, que gradualmenter eduz a exploração aleatória em favor da exploração das ações já conhecidas como as melhores, permitindo que o agente converja para uma política ótima.

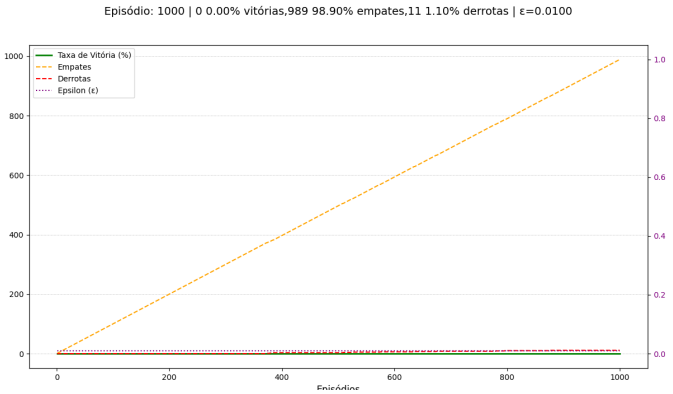

Fig. 5. Desempenho do Agente Q-learning ao final do treinamento (Episódio 1000). A linha verde representa a taxa de vitória, a laranja os empates, a vermelha as derrotas e a roxa tracejada o valor de ϵ.

A Figura 5 detalha o desempenho do agente Q-learning em 1000 episódios, ao final do treinamento. Observa-se que, das 1000 partidas representadas, o agente obteve 0 vitórias (0.00%), 989 empates (98.90%) e apenas 11 derrotas (1.10%). Este resultado é notável e demonstra a eficácia do treinamento, dado que o Jogo da Velha permite um empate ótimo para jogadores perfeitos. As 1.10% de derrotas podem ser diretamente atribuídas ao valor residual de $\epsilon = 0.0100$ (1%), que permite uma pequena probabilidade de ações aleatórias mesmo ao final do treinamento. Este pequeno percentual de exploração é crucial para evitar a convergência prematura a um subótimo, mas ocasionalmente leva a resultados não ideais. Contudo, o alto percentual de empates confirma que o agente Q-learning convergiu para uma política ótima.

A discussão central emerge do contraste entre os dois paradigmas. O Minimax representa a IA simbólica: sua "inteligência" é explícita, codificada em uma lógica de busca que, embora eficaz, é inflexível e computacionalmente cara em jogos de maior complexidade [1]. O Q-learning, por outro lado, representa uma inteligência emergente e adaptativa. Sua estratégia não é pré-programada, mas sim aprendida implicitamente nos valores de sua Tabela Q.

IV. Conclusão

Este estudo demonstrou com sucesso a implementação de dois paradigmas de IA, validando seus respectivos pontos fortes no ambiente controlado do Jogo da Velha. A análise comparativa confirmou que tanto uma abordagem de busca determinística quanto uma de aprendizado por reforço podem alcançar um desempenho ótimo, mas por caminhos fundamentalmente distintos.

O Minimax serviu como um benchmark de racionalidade perfeita, enquanto o Q-learning provou a robustez dos algoritmos de aprendizado em adquirir estratégias sofisticadas sem um modelo do mundo. Conclui-se que a escolha entre essas abordagens é estratégica e depende da natureza do problema: para sistemas fechados e resolvidos, o Minimax é uma solução garantida. Para ambientes dinâmicos, complexos ou com informações incompletas, a aprendizagem por reforço oferece um caminho mais escalável.

Como trabalhos futuros, sugere-se a transição de um Q-learning tabular para abordagens baseadas em aproximação de função, como as Redes Q-profundas (DQN), permitindo a aplicação desses conceitos a jogos com espaços de estados muito maiores [1], [2].

References

[1] K. C. Lermen and M. P. Pereira, "Jogo Da Velha: Implementação de um Agente Inteligente Utilizando Minimax e Q-learning," in *Anais do Salão de Iniciação Científica*, 2021, vol. 11, no. 1.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[3] M. G. O. Silva and M. A. V. P. e. Silva, "O Jogo Da Velha E O Algoritmo Minimax," *Brazilian Journal of Development*, vol. 9, no. 4, pp. 26867–26875, 2023.

[4] R. E. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1957.

[5] M. Sarpa and P. N. Ramos, "Tic-Tac-Toe AI: Agentes Inteligentes para o Jogo da Velha," GitHub, 2024. [Online]. Available: https://github.com/Sarpa-m/tictactoeai. [Accessed: Jun. 17, 2025].