https://eventos.utfpr.edu.br//sei/sei2020

Modelagem de agente por aprendizado de reforço para jogos educativos

Agent modelling by reinforcement learning for educational games

Thiago da Silva Teixeira
tteixeira@alunos.utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Ponta Grossa, Paraná, Brasil

Helyane Bronoski Borges
helyane@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Ponta Grossa, Paraná, Brasil

Tamara Liz Schwab Ribeiro
tamararibeiro@alunos.utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Ponta Grossa, Paraná, Brasil

RESUMO

Este trabalho explora a utilização de técnicas de aprendizado de máquina por reforço para modelagem de agentes, voltado para jogos educacionais para um público com deficiência intelectual. Para este objetivo foi criado uma função de recompensa para o agente que atenda a política de ação esperada, salvando cada parte da fase de treinamento para a criação de um jogo com uma dificuldade dinâmica, adaptativa ao usuário. O treino do agente apresentou bons resultados, aprendendo a se locomover no espaço do ambiente e selecionando ações ótimas para perseguir o jogador, revelando o potencial do uso do aprendizado de reforço para aplicações educacionais, pelo seu fator adaptativo ao usuário.

PALAVRAS-CHAVE: Aprendizado por reforço. Jogos educacionais. Aprendizado de máquina.

ABSTRACT

This work explore the use of machine learning techniques for agent modeling, aimed at educational games for an audience with intellectual disabilities. For this purpose a reward function was created for the agent that meets the expected action police, saving each part of the training phase for the creation of a game with a dynamic difficulty, adaptive to the user. The agent's training presented good results, learning to move in the space of the environment and selecting optimal actions to pursue the player, revealing the potential of the use of reinforcement learning for educational applications, by its adaptive factor to the user.

KEYWORDS: Reinforcement learning. Educational games. Machine learning.

Recebido: 19 ago. 2020.
Aprovado: 01 out. 2020.

INTRODUÇÃO

O uso de programação de agentes em jogos consiste no uso de técnicas, bem conhecidas e consolidadas, como máquina de estados finito (DIANTY *et al.*, 2015) e otimizações usando algoritmos heurísticos como A* (HART; NILSSON; RAPHAEL, 1968) em estruturas de árvore de comportamento (ZHU, 2019). Isso ocorre por serem algoritmos que apresentam bons resultados sem alocar muito recurso computacional, algo que é crítico no desenvolvimento de jogos.

Este trabalho explora a utilização de métodos de aprendizado de máquina na modelagem de agentes para jogos, de modo que seu nível de inteligência em relação ao ambiente é incrementado a cada nível, permitindo a criação de um jogo com uma dificuldade dinâmica, pois quanto maior o nível de inteligência do agente que está contra o jogador, maior a dificuldade do jogo. Para isso foi aplicado o aprendizado de reforço com o algoritmo Q-learning (WATKINS, 1989), com algumas modificações para modelar o agente com a política de ação desejada.

Jogos educacionais voltados para um público com deficiência intelectual possui alguns desafios, um deles é a alta variação de inteligência e tipo de deficiência, necessitando de adaptação de atividades que se adeque a este tipo de usuários (PIECZKOWSKI, 1999). A utilização de jogos com dificuldade dinâmica pode ajudar nesta adaptação, de modo que o jogador fique no nível de dificuldade ideal para seu aprendizado, sem grandes acréscimos entre níveis, o que pode tender a uma desistência da atividade por seu muito difícil.

O ambiente de teste criado para experimentação do agente consiste em uma interface em que o jogador precisa coletar moedas e fugir de um inimigo. Caso o jogador colete todas as moedas ele vence, caso o inimigo encoste no jogador o jogo termina. A cada vez que o jogador vence é incrementado um nível de dificuldade. Posteriormente, este ambiente será utilizado para o desenvolvimento de software educacional para pessoas com deficiência intelectual.

MATERIAIS E MÉTODOS

O aprendizado por reforço (RUSSEL; NORVIG, 2013) consiste em um método de aprendizado de máquina em que o agente interage com o ambiente, por meio de estados s e suas ações a sob o ambiente. Com base nas respostas do ambiente em relação as ações do agente, e de uma função de recompensa, tem-se uma política de ações que visam maximizar a qualidade da ação. Neste trabalho foi usado o algoritmo de reforço Q-learning, que converge valores ótimos da qualidade da ação Q, função de que avalia a qualidade de uma ação, independente da política de ação usada pelo agente, pois o agente se utiliza de uma ação gulosa, e reforça sua ação por meio da função de recompensa e os parâmetros de ação e estado, conforme descreve a Eq. (1).

$$s, a: Q(s,a) \leftarrow (1-\alpha)\, Q(s,a) + \alpha(R(s) + \gamma Q(s',a')) \quad (1)$$

Em que α corresponde a taxa de aprendizagem $(0 < \alpha < 1)$, γ o fator de desconto $(0 < \gamma < 1)$ e R a função de recompensa baseado no estado s. A função $Q(s,a)$ avalia a qualidade da ação tomada pelo agente.

A aproximação iterativa da função Q ótima pode ser feita usando uma exploração e-gulosa, em que as ações correspondentes a exploração ou explotação do agente pelo ambiente é determinado pela probabilidade $1 - e + \frac{e}{A(s)}$, em que $A(s)$ corresponde ao número de ações possíveis no estado s, e e o parâmetro de controle entre a gula e aleatoriedade.

Como demonstrado por Watkins e Dayan (WATKINS; DAYAN, 1992) se um par de estado-ação for visitado de forma a tender ao infinito, a função $Q(s, a)$ irá convergir com probabilidade 1 para a^* ótimo, considerando um valor α suficientemente pequeno. Está avaliação de melhores ações em um estado s pode ser feita por meio da Eq. (2).

$$a(s) = argmaQ(s, a) \tag{2}$$

Resultando a política gulosa como mostra a Eq. (3).

$$\pi(s) = \begin{cases} a^*, \text{com probabilidade } 1 - e \\ a_{aleatório}, \text{com probabilidade } e \end{cases} \tag{3}$$

O que permite uma solução para o problema exploração contra explotação, por meio dessa política gulosa é feito uma transição gradual entre os dois. Exploração é o estado em que o agente escolhe uma ação de forma aleatória, com o objetivo de explorar novos estados, já explotação é quando o agente se utiliza do reforço das ações descobertas na fase de exploração para conseguir estados melhores, de modo que a escolha da ação corresponda a Eq. (2). Entenda-se como bom estado, aquele que quando atingido retorna um valor maior de recompensa em relação aos outros.

DESENVOLVIMENTO

O ambiente de treinamento para a inteligência artificial foi criado em uma matriz com valores binários, indicando a localização dos pisos e das paredes. Tendo como estado s a localização do jogador e do agente, que age no ambiente de modo a mudar sua direção de movimento, resultando em quatro direções (cima, baixo, direta, esquerda), sendo estas suas possíveis ações a.

Todos os pesos de ações são iniciados aleatoriamente entre 0 e 1, em uma tabela chamada de tabela-Q, que mapeia um vetor de pesos de possíveis ações a do agente para cada estado s, sua implementação é feita através do uso de uma matriz de números de ponto flutuante.

Como o estado do agente é somente sua localização e a do jogador, inicialmente o agente não conhece o ambiente, ou seja, não sabe onde estão as paredes, caso o agente tente ultrapassar uma parede, ele permanece no mesmo lugar, porém o estado do agente ainda pode mudar devido à localização do jogador.

A função de recompensa $R(s)$ foi modelada conforme mostra a Eq (4).

$$R(s) = \begin{cases} -\frac{distância(s)}{distância_{máxima}}, \text{se } distância(s) > 0 \\ n_{piso}, \text{ se } distância(s) = 0 \end{cases} \tag{4}$$

Em que a função $distância(s)$ retorna a distância entre o agente e jogador, e $distância_{máxima}$ consiste em uma constante que contém a maior distância

possível entre o agente e o jogador no mapa. Se a $\text{distância}(s) > 0$ a função permite que o agente receba um desconto no valor da tabela-Q, inversamente proporcional à distância de seu objetivo, o que permite o agente possa convergir para o objetivo antes mesmo de sua chegada, facilitando o processo de explotação.

Já no caso de $\text{distância}(s) = 0$, indica que o jogador chegou em seu objetivo, ou seja, o agente e o jogador estão no mesmo lugar. Neste ambiente o valor de recompensa consiste no número de localizações possíveis para o agente, resultando no número de pisos que possui o mapa do ambiente, representado pela constante n_{piso}.

No método de treinamento do agente, a cada episódio de treino, o agente começa em um lugar aleatório do mapa, juntamente com outro agente que simula o jogador que permanece parado. Como o objetivo da modelagem do agente é perseguir o jogador, esta estratégia de treinamento permite que o agente aprenda a seguir o jogador, minimizando o número de passos até chegar no jogador em cada caso, onde é obtido o maior valor de recompensa. Assim o agente não escolhe outros tipos de estratégias, valores ótimos de Q, que por mais que sejam mais eficientes no ponto de vista do modelo de aprendizado, destroem a jogabilidade, como esperar o jogador chegar perto para ele começar a seguir, ou até mesmo ficar parado em cima da moeda, sendo impossível para o jogador ganhar.

RESULTADOS

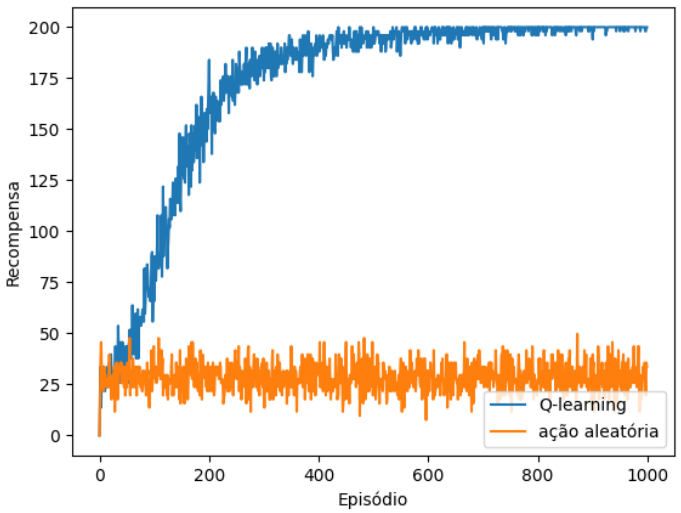
Para a utilização do algoritmo de aprendizado Q-learning alguns parâmetros de treinamento do agente precisam ser definidos, tais como:

- Episódios = 100.000
- Taxa de aprendizado (α) = 0.1
- Fator de desconto (γ) = 0.95
- Fator de controle de gula (e) = 1
- Desconto de controle de gula = $\frac{e}{\frac{episódios}{2} - 1}$

A parte de exploração é necessária para o agente localizar o jogador e assim por meio da explotação o agente reforça a rota ótima descoberta. Usando um fator de controle de gula e com o valor inicial igual a 1, permite que no início do treinamento o agente explore o ambiente e a cada episódio o e é subtraído pelo fator de desconto de controle de gula, de forma que o agente não irá mais escolher aleatoriamente os estados depois da metade do treino, deixando para a explotação convergir nas ações ótimas.

O treinamento do agente é analisado pelo gráfico da Figura 1. O número de episódio é dividido em 1000 partes que formam os pontos no gráfico. Cada ponto representa a média das recompensas naquele intervalo. Observa-se que antes dos 40.000 episódios há um crescimento da média das recompensas, mostrando que o agente consegue chegar até o jogador mesmo na fase de exploração, como demostra a curva da recompensa média.

Figura 1 – Gráfico de recompensa média da fase de treino



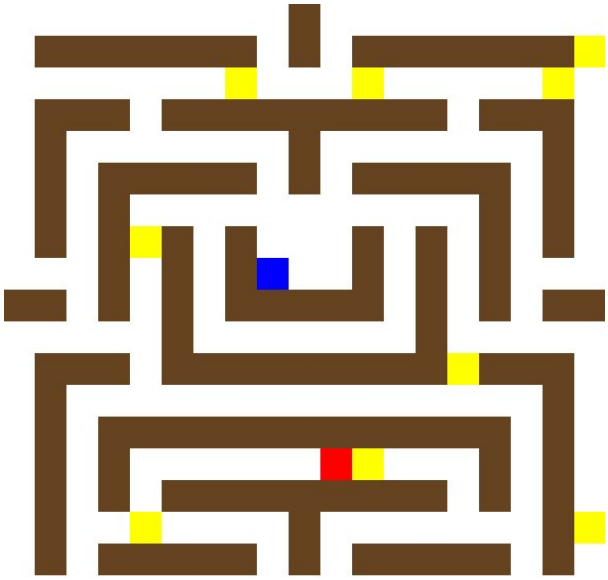
Fonte: Autoria própria (2020).

Nota-se também, na Figura 1, que depois do episódio 60.000, a linha de recompensa tende a uma reta, indicando que o agente conseguiu chegar no seu resultado em praticamente todos os casos posteriormente, apenas com pequenas variações na curva. A linha de ação aleatória representa um agente sem um algoritmo de aprendizado, que explora o ambiente de forma aleatória e serve de referência para avaliar o aprendizado do agente.

O treinamento do agente é dividido em 30 níveis de forma uniforme, salvando as tabelas Q com o aprendizado do agente. No jogo estas tabelas Q são carregadas referentes a cada nível de jogo, de modo que o jogo tenha uma dificuldade progressiva.

O ambiente foi implementado utilizando a linguagem Python (VAN ROSSUM et al., 2009), com o apoio de um modulo voltado para o desenvolvimento de jogos *pygame* (SHINNERS, 2011). O ambiente possui uma interface, conforme ilustra a Figura 2, em que a parte branca representa o piso; o quadrado azul o agente treinado por meio do algoritmo Q-learning; o quadrado vermelho o jogador; o quadrado marrom a parede; e o quadrado amarelo as moedas. Tal ambiente com visualização serve para testar manualmente a interação entre o agente e o jogador, que utiliza as setas direcionais do teclado para mover o personagem e capturar as moedas necessárias para ganhar o jogo.

Figura 2 – Visualização do ambiente



Fonte: Autoria própria (2020).

CONCLUSÃO

A modelagem do agente funcionou para seu propósito, tendo como objetivo principal perseguir o jogador, de forma a ter um incremento de aprendizado sob o ambiente de forma gradual, deixando o jogo com uma dificuldade gradativa. Porém, para chegar neste objetivo foi necessário utilizar técnicas não intuitivas, para evitar que o agente chegue em uma conclusão ótima que deixe o jogo impossível para o jogador.

Uma divisão da fase de treino do agente, salvando o estado da tabela Q, permitiu uma base para a criação de um jogo com dificuldade adaptativa, revelando um potencial uso para criação de jogos educativos voltado a deficientes intelectuais, por possuir propriedades adaptativas.

Apesar de ser um problema simples de locomoção, o agente conseguiu aprender a se locomover no ambiente e chegar a perseguir o jogador, apenas com sua localização e a do jogador. Para trabalhos futuros, considerando em um jogo que o agente comece em um local fixo, otimizações do tamanho da tabela podem ser feitas, já que seus movimentos constituem um fator estocástico, sendo possível utilizar como estado somente a localização do jogador, porém teste precisam ver feitos para avaliar a convergência da política de ação esperada.

Com base no aprendizado deste agente será implementado um jogo educacional voltado a pessoas com deficiência intelectual.

REFERÊNCIAS

DIANTY, R. E. *et al.* First Aid Simulation Game with Finite State Machine Model. In: International Conference on Information, Communication Technology and Systems. 2015. Anais... Surabaya, 2015, p. 157-162.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. **IEEE Transactions on Systems Science and Cybernetics.** v. 4, p. 100-107, 1968.

PIECZKOWSKI, T. M. Z. O espaço das crianças portadoras de necessidade educacionais especiais-Deficiência mental na educação. **Revista Pedagógica**, v. 1, n. 3, p. 33-52, 1999.

RUSSEL, S. J.; NORVIG, P. **Inteligência artificial**. Tradução da 3ª Edição. Elsevier, 2013.

SHINNERS, P. Pygame. 2011. Disponível em: <https://www.pygame.org/>. Acesso em: 4 set. 2020.

VAN ROSSUM, G.; DRAKE, F. L. **Python 3 Reference Manual**. Scotts Valley, CA: CreateSpace, 2009.

WATKINS, C. J. C. H. **Learning from Delayed Rewards**. 1989. 241f. Tese de Doutorado, King's College. Cambridge, Inglaterra, 1989.

WATKINS, C. J. C. H.; DAYAN, P. Q-learning. **Machine learning**. 1992.

ZHU, X. Behavior tree design of inteligente behavior of non-player character (NPC) based on Unity3D**. Jornal of Intelligente & Fuzzy Systems**. v. 37. p. 6071-6079, 2019.