APLICAÇÃO DO ALGORITMO Q-LEARNING NO JOGO FLAPPY BIRD

Rafael Scotti Zanella

rafaels.zanella@gmail.com

Resumo. Este relatório tem a finalidade de descrever a aplicação da técnica de Inteligência Artificial, aprendizado por reforço, Q-Learning, no jogo Flappy Bird. Assim será apresentado a modelagem do projeto, os resultados e problemas encontrados.

1 INTRODUÇÃO E APRESENTAÇÃO DO PROBLEMA

Após as aulas de Inteligência Artificial (IA), um dos algoritmos que mais chamou a atenção, foi a técnica de aprendizado por reforço, Q-Learning; a qual funciona basicamente de forma a aplicar uma recompensa ou penalidade a um agente de acordo com o resultado de suas ações. Tendo isso em mente, foi resolvido aplicar essa técnica a algum jogo para ver de fato esse algoritmo funcionando na prática.

Assim, foi procurado algum jogo que não fosse muito complexo, mas que apresentasse um certo grau de dificuldade para ser jogado. Dessa forma, foi escolhido o jogo *Flappy Bird*.

O jogo *Flappy Bird*, acontece em um cenário em que o agente é um pássaro que deve desviar de obstáculos (canos, ou "*pipes*"), utilizando de apenas duas ações, pular ou não pular. Assim visando chegar na maior distância possível. A grande dificuldade desse jogo, é conseguir aplicar as ações corretamente de forma a atingir o alvo sem encostar nos obstáculos.

Para a aplicação da inteligência artificial, não foi utilizado o jogo original, foi criado uma réplica utilizando Javascript, sem muitos elementos visuais, contendo apenas o necessário para o funcionamento do jogo. Também foi utilizado o *framework Electron* que permite criar aplicações *desktop* utilizando Javascript.

Na tela do jogo, é possível visualizar diversas informações a respeito dos estados, ações e recompensa, como também a geração em que o agente se encontra, seu *score* (pontuação) atual e *record* (pontuação máxima atingida).

2 Q-LEARNING

A técnica de Inteligência Artificial Q-Learning, se baseia no aprendizado por reforço, o qual aplica uma recompensa (positiva ou negativa), de acordo com a ação de um agente em um determinado estado. A técnica utiliza também duas variáveis que determinam o fator de aprendizado e de desconto, as quais podem variar os valores entre 0 e 1.

A fórmula "*Q"*, do algoritmo Q-Learning é descrita da seguinte forma:

Q = S[ultimaAcao] + a*((r) + y*(max(S') - (S[ultimaAcao])).

Sendo:

S[ultimaAcao]: a ação escolhida no último estado;

"*a*": é o fator de aprendizado que pode ser definido como: *0 < a <= 1*;

"*r*": é o valor da recompensa;

"*y*": é o fator de desconto que pode ser definido como: *0 <= y < 1*;

max(S'): é a maior ação do próximo estado (derivado de *S*);

O resultado fica armazenado na variável "*Q*" que é aplicada a ação escolhida referente ao estado *S*.

2.1 Variáveis de definição de estado

Inicialmente, foram definidas cinco variáveis, que são a distância entre o agente o limite superior e inferior, o tamanho dos obstaculos (de cima e de baixo) e a distância até o alvo.

Após algumas correções com o objetivo de melhorar o algoritmo, foram introduzidas mais duas variáveis de estado. A primeira indica o estado de vida do agente e a segunda informa sua velocidade do pulo ou queda em determinado estado. Assim totalizando 7 variáveis de estado.

2.2 Possíveis ações

Como descrito anteriormente, o agente possui duas ações de movimento, que são, Pular ou Não Pular.

3 CRONOGRAMA

DATA	ATIVIDADES
01/11	1 Definição do problema, suas variáveis e limites; 2 Definição da técnica; 3 Início do desenvolvimento;
08/11	1 Desenvolvimento do jogo;
15/11	1 Desenvolvimento do jogo; 2 Início do desenvolvimento do algoritmo Q-Learning;
22/11	1 Desenvolvimento do algoritmo Q-Learning; 2 Solucionando problemas no projeto (*bugs*);
29/11	1 Entrega do relatório; 2 Apresentação final;

4 MODELAGEM DA SOLUÇÃO

Para o desenvolvimento do trabalho, foi utilizado a linguagem de programação Javascript, e os *frameworks p5.js* e *Electron*, respectivamente um para o desenvolvimento da parte gráfica do jogo e o outro para criação de uma aplicação *desktop*.

Foram criadas duas classes, uma para o pássaro (*Bird*) e outra para os obstáculos (*Pipes*). E em um outro arquivo, é feito o gerenciamento do jogo, como a criação de *pipes*, *bird*, controles, movimentos e, também, a técnica de Inteligência Artificial Q-Learning.

Para implementação do algoritmo de IA, foi necessário um *HashMap*, onde ficam armazenados todos os estados do jogo e suas ações, para assim então, ser possível enviar para a função "*Q*" o último estado jogado (*S*), o estado derivado de *S* (*S'*) e a última ação escolhida.

Após estas ações, o algoritmo escolhe uma recompensa "*r"* que será aplicada ao último estado jogado, podendo ser positiva ou negativa.

Também é possível salvar os estados já descobertos em um arquivo *JSON*, porém é necessário digitar no *console*, "*mapToJson(mapEstados);*", pois não foi implementado uma forma gráfica para realizar tal tarefa. Além disso também falta implementar um método para carregar esse arquivo para dentro da aplicação.

5 RESULTADOS

Após o desenvolvimento e algumas horas rodando, o programa não funcionou. A primeira mudança feita, foi nos fatores de aprendizado e desconto, porém, também não funcionou. Então foi decidido alterar o sistema de recompensas, que antes, apenas detectava se o pássaro estava ou não batendo em algum obstáculo e já aplicava alguma recompensa. Então, foi criada uma outra variável de estado, "*vivo*", o qual mantém um valor booleano para saber se o pássaro está vivo ou não; essa variável é constantemente alterada, devido a detecção de colisão. Assim, foi deixado mais algumas horas rodando. Quando estava em torno de 400 repetições, o pássaro começou apresentar sinais de aprendizado mostrando um certo esforço para desviar dos obstáculos, porém, após algumas milhares de repetições, a pontuação máxima foi 4.

Logo foi decidido que por existirem milhões de estados, eles não são todos totalmente necessários, por exemplo, o pássaro não precisa de um momento exato para pular ou cair, apenas um momento que seja suficiente para desviar dos obstáculos. Então as variáveis de estados foram divididas por 20, com exceção da que verifica a distância no eixo x entre o pássaro e o alvo, que foi dividida por 16.

Feito essas alterações, o pássaro não apresentou melhorias no aprendizado, chegando ao mesmo *score* de 4 pontos, porém, além de ter sido em menos tempo, diminuiu consideravelmente a quantidade de estados.

Após foi alterado mais uma vez os fatores de aprendizado e desconto, o que resultou em um progresso de 4 para 11 pontos.

A princípio parecia funcionar, então foi deixado aproximadamente 12 horas rodando para ver se surgiam melhoras. O que não aconteceu.

Então o código foi revisado mais uma vez, e após não saber onde estava o problema, foi decidido reescrever o método que gerenciava os estados e também o algoritmo Q-Learning. Foi então que alguns *bugs* da versão anterior começaram a aparecer, o algoritmo não estava reconhecendo o exato estado anterior ao S'. Então foi necessário um certo tempo repensando e refazendo algumas partes do código, até que o problema pareceu solucionado. Mais algum tempo rodando e os pontos chegaram a apenas 8; Após foi alterado a quantidade de recompensas e penalidades, o que resultou em uma melhora para 16 pontos. A qual aconteceu apenas uma vez, pois após deixar algumas horas rodando novamente, o máximo foram 11 pontos.

A partir de mais uma análise no código, havia sido concluído que por se ter muitos estados, fazer o aprendizado baseado apenas no estado S e S' não estava sendo suficiente. Então o sistema de recompensas sofreu alterações mais uma vez. Por existirem milhares estados que visualmente não parecem de fato ser diferentes, definir um número X de estados para aplicar uma recompensa acaba sendo inviável. Então passou-se a ser analisado o posicionamento do pássaro em relação ao cenário. Assim todos os estados conseguem receber uma recompensa ou penalidade e não apenas o estado anterior a uma colisão, que era o que estava acontecendo.

Devido às alterações, o pássaro se mostrou mais "inteligente" porém apresentava um mesmo problema que havia nas outras versões, ele não tinha conhecimento das questões como força, gravidade, e velocidade, assim ele não sabia o quanto deveria esperar para pular novamente. Então foi adicionada mais uma variável de estado que mostra a velocidade alcançada em cada estado enquanto pula ou cai.

Após essas alterações em 4 minutos o pássaro já havia superado o *record* anterior. Em 300 gerações já apresentava um score de aproximadamente 160 pontos.

Agora, no momento em que esse relatório está sendo escrito, o jogo se encontra na geração de número 4538 com um record de 420 pontos.

Para tal pontuação, o fator de aprendizado (a) foi definido como 0.1, o fator de desconto (y) 1.0, e a recompensa (r) foi definida da seguinte forma:

Se no estado S' o pássaro morrer: r = -0.6;

Se a distância em relação ao limite superior for menor ou igual a 2 e a última ação for Pular: r = -1;

Se a distância em relação ao limite inferior for menor ou igual a 1 e a última ação for Cair: r = -1;

Se a distância em relação ao limite superior for menor ou igual a 2 e a última ação for Cair: r = +1;

Se a distância em relação ao limite inferior for menor ou igual a 1 e a última ação for Pular: r = +1;

Se a distância em relação ao pipe inferior for menor ou igual 1 e a última ação for Cair: r = -0.1;

Se a distância em relação ao pipe superior for maior ou igual a -2 e a última ação for Pular: r = -0.1;

Se a distância em relação ao pipe inferior for menor ou igual a 1 e a última ação for Pular: r = +0.01;

Se a distância em relação ao pipe superior for maior ou igual a -2 e a última ação for Cair: r = +0.01;

Qualquer posicionamento diferente dos citados recebe recompensa de +0.0005;

6 PROBLEMAS E SOLUÇÕES

O principal problema no desenvolvimento deste trabalho, foi criar um programa com interface gráfica, pois não havia estudado isso ainda e, devido a isso, metade do tempo disponível para implementação da IA, foi aprendendo algum framework que facilitasse isso. Assim foi utilizado o *p5.js*, que é uma biblioteca javascript com funcionalidades de desenho.

Para criação do jogo, foi utilizado como base um tutorial encontrado no youtube, o mesmo que ensina também a utilizar a biblioteca *p5.js*.

Em relação ao desenvolvimento da técnica Q-Learning, o maior problema foi entender como aplicar as recompensas de forma que o agente aprendesse de fato a jogar.

7 CONSIDERAÇÕES FINAIS

A técnica de aprendizado por reforço, Q-Learning, é de fato eficiente, embora, quando se tem milhões de estados, talvez não seja a melhor técnica. O maior problema na sua utilização é o tempo demorado para descobrir os estados e treinar o agente e, também a definição de valores como os fatores de desconto e aprendizado e, as recompensas. Porém após descobertos os estados a técnica proporciona uma grande evolução na "inteligência" do agente. Portanto é possível obter ótimos resultados utilizando essa Inteligência Artificial.

8 REFERÊNCIAS BIBLIOGRÁFICAS

Italo Lelis - LearnSnake: Teaching an AI to play Snake using Reinforcement Learning (Q-Learning). Disponível em:
<https://italolelis.com/snake> Acesso em 15 de novembro de 2018.

The Coding Train - Coding Challenge #31: Flappy Bird. Disponível em:
<https://www.youtube.com/watch?v=cXgA1d_E-jY> Acesso em 02 de novembro de 2018.

P5.Js. Disponível em:
<https://p5js.org/> Acesso em 02 de novembro de 2018.

Electron. Disponível em:
<https://electronjs.org/> Acesso em 08 de novembro de 2018.