Trabalho 2 : Inteligência Artificial

Juliana Camilo Repossi

14 de agosto de 2023

Resumo

O artigo em questão aborda o desenvolvimento e a análise de resultados de uma IA que aprende a jogar o jogo do Dino da Google (http://chrome://dino e https://nira.com/chrome-dino), através do conceito de aprendizagem por reforço (reinforcement learning). Nesse modelo, a máquina vai aprendendo com as decisões que ela experimenta, as que resultam em bons scores no jogo e as que pioram a sua performance. Assim ela é capaz de reconhecer padrões e jogar cada vez melhor o jogo do dino, maximizando a pontuação final, o qual é o objetivo principal.

1 Introdução

O trabalho aqui detalhado foi proposto pelo Professor Flávio Miguel Varejão, para a disciplina de Inteligência Artificial, ministrada na UFES, e tem como proposta criar um modelo de IA que aprende por reforço diante de um ambiente totalmente imprevisível, como o de um jogo.

Para a construção desse agente inteligente foi utilizada uma técnica já bem conhecida na área: combinar uma heurística a um método de classificação para treiná-la a jogar. A heurística escolhida para essa implementação foi o algoritmo genético e o modelo de classificação foi de rede neural.

Neste caso a heurística será responsável por produzir uma gama de pesos para serem aplicados na rede neural e, a rede, que é um classificador, usará dos pesos combinados com as entradas do jogo para tomar uma decisão. Essa decisão será se o dino "pula"ou "abaixa".

Para compor os dados de entrada da rede neural foram escolhidos os seguintes dados do jogo:

- distance - Distância do dino até o próximo obstáculo;
- obHeight - Altura do próximo obstáculo;
- speed - Velocidade atual do jogo;
- obTyped - Tipo do obstáculo que virá, podendo ser "SmallCactus", "LargeCactus"ou "Bird"tendo este último três variações de altura: baixo, médio e alto.

Esses atributos pareciam bons dados de antemão, visto que principalmente a distância e a velocidade impactam muito diretamente a decisão da IA. Uma vez que a medida que a velocidade do dino cresce, ele deve pular com maior antecedência/distância do obstáculo que virá. Também é muito importante saber a altura do objeto, pois se ele tiver a uma altura que dê para abaixar essa decisão deverá ter mais preferência pela nossa IA com o passar do tempo, já que o tempo de resposta para abaixar e disponibilidade para agir a um posterior obstáculo será mais rápida se comparado a tentar pulá-lo. Dessa forma é esperado que ela entenda essa nuance durante o treinamento. Por fim, o quarto atributo escolhido para essa implementação foi o tipo do objeto e pode não ter sido uma escolha tão boa, mas iremos deixar essa constatação para a análise de resultados.

Para o melhor entendimento, o artigo será dividido em subseções. Essas trarão detalhes sobre o classificador e a heurística escolhida. Os resultados que essa combinação trouxe para a aplicação do jogo do dinossauro. E, por fim, trará uma conclusão geral embasada na comparação dos resultados obtidos por essa implementação comparado-a com a disponibilizada pelo professor. O código de implementação da IA bem como a geração dos resultados estatísticos da conclusão estarão disponíveis no repositório do gitHub disponível nesse link.

2 Descrição do Classificador Rede Neural

Como já mencionado o classificador escolhido para essa aplicação foi uma rede neural. Essa rede conta com 4 neurônios de entrada, 4 neurônios intermediários e 1 neurônio de saída, como mostrado na figura abaixo pelos círculos pretos:

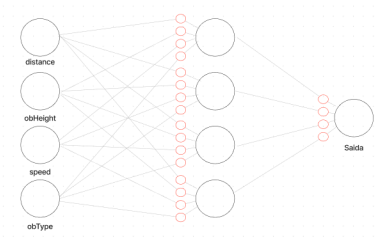

Figura 1: Esboço da rede neural com 9 neurônios utilizada

Na camada primeira camada, ou camada de entrada, esses neurônios têm como valores os dados de distância do dinossauro até o próximo obstáculo, altura deste, velocidade do dinossauro, e do tipo de objeto que irá surgir. Este último atributo precisou ser tratado, visto que a rede neural trabalha com valores numéricos, e não nominais. Assim, a escolha foi por transformar os dois tipos de cactos e o tipo pássaro com altura baixa em 0, e os tipos pássaro com altura média e alta em -700. Esses valores foram escolhidos a fim de inserir um viés para que a máquina aprendesse mais rápido a responder com um valor >=0 aos primeiros, já que para passá-los só é possível com um pulo. E, com um valor <0 aos segundos, já que diante deles abaixar pode ser uma boa decisão.

Já na camada intermediária esses valores de entrada são submetidos a ponderações feitas pelos círculos vermelhos. Cada um desses círculos representa um peso a ser definido pela heurística que será detalhada mais a frente. Por agora só basta saber que cada círculo vermelho terá um valor numérico ao qual cada entrada será multiplicada e somada para produzir um valor numérico para cada um dos 4 neurônios intermediários.

E, por fim, um neurônio de saída, que se utilizará dos quatro resultados produzidos em cada um dos quatro neurônios anteriores para fazer uma nova ponderação com os seus valores de peso para depois de uma soma final, responder ao jogo com um "UP", se o valor numérico de saída for maior ou igual a zero e "DOWN"se o valor numérico de saída for menor que zero.

Essa é a descrição do funcionamento da rede neural aplicada a este contexto. E para que ela funcione e traga bons resultados precisamos que ao combinarmos essas entradas a essas operações sobre os pesos e valores numéricos intermediários tenhamos uma boa decisão, ou seja, que nosso dinossauro pule no momento certo e abaixe nos demais momentos. Para chegar a esse comportamento esperado, é preciso regular esse comportamento por meio dos hiperparâmetros da rede, ou seja, dos pesos que ela usa para fazer os cálculos de decisão.

Para buscar esses pesos ideais para deixar nosso dinossauro performático vamos nos utilizar de uma heurística, que será detalhada na próxima seção.

3 Descrição da Meta Heurística algoritmo genético

Como dito a heurística exerce o papel de procurar pelos melhores pesos para compor a rede neural, de modo que ela possa responder da forma mais inteligente possível aos obstáculos do jogo.

É comum utilizar de heurísticas nesses casos já que estamos diante de um universo infinito de possibilidades de combinações de pesos a serem escolhidos. E, testar cada uma delas a fim de encontrar a melhor solução torna-se inviável. Nesses casos as heurísticas agregam no sentido de prover técnicas para alcançar respostas adequadas, embora várias vezes imperfeitas, para uma dada situação. Assim, iremos contar com o auxílio de uma heurística muito interessante para nos auxiliar a encontrar bons pesos para o classificador, o algoritmo genético.

Para fazer essa implementação foi preciso transformar o esquema visual da figura acima em uma lista de pesos de tamanho $4(4 + 1)$. Após definir o tamanho, começamos criando várias listas iniciais de tamanho 20

com números inteiros gerados aleatoriamente de -1000 até 1000. Foram testados outros intervalos maiores, mas eles não apresentaram ganhos de scores significativos, logo, manteve-se a geração inicial aleatória de menor amplitude.

Após essa geração inicial, submete-se cada uma dessas listas de pesos à rede neural em 3 partidas do jogo, guardando a média menos o desvio das pontuações geradas para cada combinação. Partindo dessa base, a ideia é aplicar os conceitos de genética aos pesos considerando seus respectivos resultados, com a intenção de produzir pontuações ainda melhores da seguinte forma:

- Primeiro Passo: Fazer a seleção natural dos indivíduos mais fortes! Dada a dupla de scores e pesos o algoritmo separa N amostras com os melhores resultados e coloca-as na próxima iteração, ou seja, perpétua as amostras melhores adaptadas ao jogo;
- Segundo Passo: Faz uma seleção ponderada de itens para participar dos fenômenos genéticos. Essa seleção é aleatória, porém dá maior peso para serem escolhidas as listas de pesos com melhores pontuações;
- Terceiro Passo: Faz um crossover nessas amostras! Esse evento consiste na combinação de duas listas de pesos de forma que uma seja o pai e a outra seja a mãe, e delas saiam duas combinações diferentes com parte dos pesos do pai e parte dos da mãe. A mistura "genética"é uma tentativa de produzir uma sequência mais adaptável ao ambiente do jogo.
- Quarto Passo: Aplicar mutações a essas listas. Mutações seriam mudanças específicas em um item das listas, somando um valor, trocando por outro, mudando seu sinal, etc. Também na tentativa de criar uma combinação ainda melhor para o contexto. Foram testados vários valores de mutações nesse trabalho, porém o que melhor respondeu foi de de soma/subtração de uma unidade.

Depois disso juntam-se as melhores amostras às amostras modificadas pelo algoritmo genético a fim de reavaliar o desempenho delas. Esse processo é repetido até uma quantidade de tempo, ou um número máximo de iterações, ou até que as amostras convirjam. A ideia é que com o passar das iterações os scores vão ficando cada vez melhores. Ou seja, vão evoluindo "geneticamente".

4 Descrição dos Experimentos Realizados e seus Resultados

Depois de definida a estrutura, o próximo passo é deixar que a IA aprenda a partir de tentativa e erro a melhorar a sua forma de jogar. Depois de alguns treinos e tentativas, mudando a taxa de mutação, o coeficiente de mutação, a taxa de crossover e a taxa de seleção, chegou-se na versão escolhida para ser apresentada nesse artigo. Vale ressaltar que algumas versões tiveram desempenhos muito próximos e outras bem inferiores. Por exemplo, com o treinamento descobriu-se que deixar a taxa de seleção dos N melhores baixa era mais benéfico para a aprendizagem, já que o algoritmo teria mais bases distintas para modificar podendo encontrar soluções melhores com o passar das iterações. Assim, escolheu-se manter a taxa de mutação alta, em 0.8, com uma pequena modificação em compensação, somando ou subitraindo de uma unidade o peso. Uma taxa de seleção baixa, 0.2. E, uma taxa de crossover alta, devido ao mesmo motivo da taxa de mutação. Nessas configurações, treinando 500 dinossauros em paralelo, com a primeira geração aleatória no intervalo de -1000 a 1000, a IA apredeu rapidamente a a chegar nos seus 1300-1500 pontos sem nenhuma dificuldade. Pode-se perceber isso pelo gráfico a seguir que demonstra a evolução dos scores em um treinamento de 1.000 iterações, pontuando o melhor score de cada rodada.

É possível observar muitas coisas nessa evolução, mas duas em especial serão colocadas em evidência nessa análise. A primeira é a velocidade que o agente inteligente aprende a ficar bom no jogo partindo de dados aleatórios, isso pode indicar que o jogo tenha muitos máximos locais nessa modelagem, muitas combinações diferentes com boa performance. Porém pode-se perceber que ao longo das demais iterações o jogo apenas oscila, não conseguindo evoluir mais do que 1800 pontos de média menos desvio em 3 rodadas subsequentes. Assim, foram pensadas algumas tentativas para aumentar esse score:

- Tentativa 1 - Realimentar a entrada do jogo: a ideia era colocar bons valores obtidos em execuções passadas nas execuções subsequentes a fim de chegar mais rápido em uma pontuação maior. Mas isso

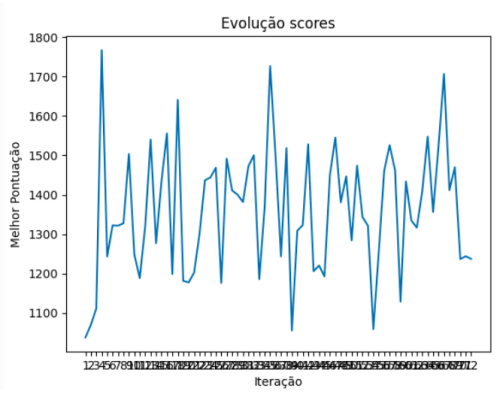

Figura 2: Evolução do melhor score da IA durante 1.000 rodadas com 500 dinossauros em paralelo.

não contribuiu para que o dino aprendesse a passar dos 1800 pontos. Essa solução surtiria efeito se algumas das boas soluções obtidas em execuções passadas estivessem próximas a uma ótima solução, o que não foi o caso.

- Tentativa 2 - Modificar os parâmetros do sistema. Diferentes combinações foram testadas, como: aumentar a amplitude de valores que a geração inicial randômica faria, aumentar/diminuir as porcentagens de crossover e mutação, além da magnitude da mutação. Foram testadas a soma e multiplicação de valores grandes e a troca do sinal dos valores mutados, com uma péssima performance. E, modificar os pessos com magnitudes pequenas, que tiveram melhor desempenho, mas ainda incapazes de evoluir os escores para uma média maior que 1500 pontos. A quantidade de dinossauros e as horas também foram exploradas em valores mais altos, mas sem sucesso.

- Tentativa 3 - Começar o jogo numa velocidade maior, próxima a que os dinossauros começavam a perder, bem como começar na velocidade normal e subi-la mais rapidamente, para tentar encontrar um dino mais adaptável a esse meio. E o único resultado que se obteve aqui foi dar a sorte de encontrar um cenário em que eles se saiam bem, mas em mais rodadas não traziam benefícios.

Diante de boas três tentativas fracassadas a ideia foi assistir a algumas rodadas do jogo acontecendo para entender porque a IA não conseguia evoluir em uma velocidade maior para além dos 1700 pontos. Só assim foi possível visualizar que o gargalo da aprendizagem era que ela não era capaz de capturar duas nuances do jogo: a primeira, que ela não aprendia a abaixar o dinossauro mais rápido com o aumento da velocidade, assim não conseguia fazer o pulo para o próximo obstáculo; E, a segunda, que ela não era capaz de superar obstáculos com um pulo duplo propositalmente, apenas casualmente, ou seja, sem querer. Isso a limitou. Bom, mas porque ela não aprendeu a superar esses casos? Simples, não foi dado dados suficientes a ela para enxergar esses cenários do jogo, com as entradas escolhidas o agente só tomava uma decisão gulosa que englobava apenas o próximo objeto. E isso nas altas velocidades era mortal. Tirando essa limitação ela teve boas performances se comparada ao resultado disponibilizado pelo professor, como segue na comparação dos 30 scores das rodadas finais jogadas por cada agente inteligente.

	Média	**Desvio**	Iter1	Iter2	Iter3	Iter4	Iter5
Base(professor)	**1068.183333**	**304.035524**	1214.00	759.00	1164.25	977.25	1201.00
Real(juliana)	**1517.216667**	**238.634595**	1863.00	1767.50	1089.00	1715.00	1799.750

Iter6	Iter7	Iter8	Iter9	Iter10	Iter11	Iter12	Iter13	Iter14	Iter15
930.00	1427.75	799.50	1006.25	783.50	728.50	419.25	1389.50	730.00	1306.25
1509.50	1695.75	1332.50	1621.00	1435.25	1156.50	1767.25	1843.00	1198.75	986.25

Iter16	Iter17	Iter18	Iter19	Iter20	Iter21	Iter22	Iter23	Iter24	Iter25
675.50	1359.50	1000.25	1284.50	1350.00	751.00	1418.75	1276.50	1645.75	860.00
1292.75	1891.00	1228.25	1619.25	1482.25	1419.25	1404.75	1717.75	1307.75	1517.00

Iter26	Iter27	Iter28	Iter29	Iter30
745.50	1426.25	783.50	1149.75	1482.25
1441.25	1626.50	1530.50	1738.00	1520.25

teste-T	teste-Wilcoxon
5.129775e-08	0.000015

Destarte, é fácil inferir que a solução desenvolvida teve performance superior a de base comparada, com diferenças significativas entre as duas soluções, o que pode ser comprovado pelo valor menor que 0,05 nos testes teste t pareado e não paramétrico de wilcoxon, apontando que há diferenças significativas entre eles a um nıvel de significância de 95%. Assim como é evidente visualmente pelo boxplot a seguir.

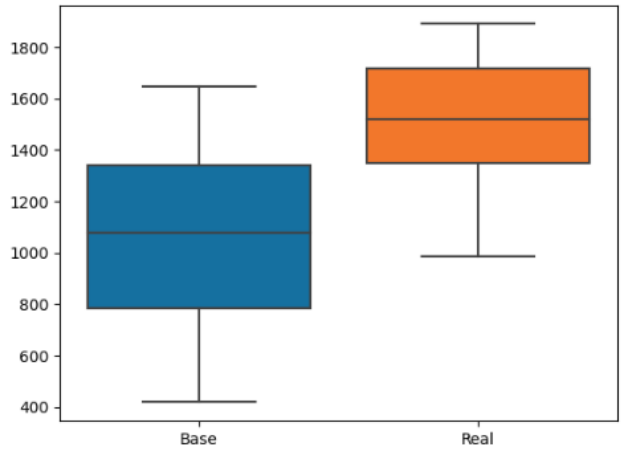

Figura 3: Boxplot de comparação dos resultados em 30 rodadas da base disponibilizada pelo professor e a da implementação do artigo(Real)

5 Conclusão

Diante dos dados apresentados pode-se concluir que a combinação da rede neural com o algoritmo genético foi uma boa escolha para implementar a técnica de treinamento de uma IA por aprendizagem por reforço, já que alcançou bons resultados com uma certa rapidez, como já demonstrados nas seções anteriores. Porém cabe citar que esse seria um modelo de entrada para o estudo dessa IA, cabendo como próxima fase fazer a mudança dos dados de entrada e repetindo todo o processo de treino a fim de inserir nela a capacidade de enxergar o próximo objeto do mapa para que tome uma decisão mais escalável e menos gulosa com o aumento da velocidade do jogo. Nesse mesmo sentido, caberia reavaliar a inserção da entrada obType, que de maneira geral não parece ter sido uma boa escolha, já que de forma macro só foi capaz de passar para o modelo a altura do objeto, sendo assim redundante. Então, numa próxima fase de estudo essa suposição caberia ser estudada. A implementação contribuiu grandemente para entender melhor como é feita a criação de um agente inteligente sem código imperativo e foi de grande valia para a aprendizagem.

6 Referências Bibliográficas

As referências para a escrita desse relatório foram obtidas a partir dos materiais de aula disponibilizados pelo Dr. Flávio Miguel Varejão, bem como suas aulas expositivas e laboratórios ministrados na disciplina de Inteligência Artificial - UFES. E, dos vídeos Inteligência Artificial com Dinossauro da Google e Inteligência Artificial destruindo no dinossauro da Google! (Rede Neural).