Trabalho 2 de Inteligência Artificial

Carlo Smaniotto Mantovani (21103691-8)

# 1. Rede Neural e Minimax

Primeiramente, se buscou entender o algoritmo base do trabalho. O algoritmo utiliza uma Rede Neural com 9 entradas, três camadas oculta de 9 neurônios e uma camada de saída com 9 neurônios de saída, como se trata de 9 entradas a rede irá possuir um total de 360 pesos, sendo 90 de cada camada oculta e 90 da camada de saída:

10 pesos por neurônio (9 + bias) \* 9 neurônios \* 4 camadas = 360 pesos

Essa rede propaga uma dada configuração de tabuleiro de jogo da velha pelos seus neurônios, gerando um valor de saída em cada neurônio de saída. Para esse trabalho especifíco o cálculo das saídas do neurônios foi realizado com a função de tangente hiperbólica.

O neurônio que gerou o maior valor de saída será o que irá determinar a próxima jogada, ou seja, se por exemplo este neurônio for o neurônio 2, a rede irá jogar na posição 2 do tabuleiro. Segue a configuração de índices do tabuleiro:

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

O algoritmo de Minimax, por sua vez, joga contra a rede e possui o objetivo de maximizar as suas chances de vitória e minimizar as chances de derrota. No contexto do jogo da velha, ele possui uma árvore que representa as jogadas possíveis e, ao observar os diversos ramos da árvore, seleciona a jogada que possui uma maior chance de vitória ou empate.

No entanto, esse algoritmo possui um custo muito alto, devido a necessidade de se verificar todas as jogadas possíveis a partir de cada configuração de tabuleiro em cada jogada.

# 2. Algoritmo Genético

## 2.1. População

No algoritmo genético implementado a população consiste em cromossomos que representam os pesos da rede que variam de -1 a 1. Essa população é representada através de uma matriz 30x361, sendo que cada linha da matriz é um cromossomo e cada coluna um gene, logo, a junção de uma linha e coluna representa o gene de um cromossomo. A última coluna de cada cromossomo representa a aptidão daquele cromossomo que será calculada durante o teste de cada um. Segue um exemplo de população inicial cujos pesos são gerados aleatoriamente:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | P0 | P1 | ... | P359 | P360 | P361 (Apt) |
| C0 | 0,245... | -0,123... | X | 0,753... | -0,456... | 0 |
| C1 | -0,823... | 0,575... | X | 0,321... | -0,783... | 0 |
| ... | X | X | X | X | X | X |
| C29 | -0,895... | -0,364... | X | 0,568... | 0,634... | 0 |

A aptidão inicia em 0, visto que ela será calculada posteriormente.

## 2.2. Aptidão

A aptidão é calculada durante a execução e é realizado uma “pontuação” com base na jogada da rede e na configuração do tabuleiro em cada momento do jogo. Segue a base de pontuação utilizada:

1. Jogada em Posição Livre: aptidão +(2 \* rodada atual) + bonus(inicia alto e diminui com o decorrer do algoritmo).
2. Derrota da Rede: aptidão -20.
3. Vitória da Rede: aptidão + (120/rodada atual) (maior pontuacao para vitoria com poucos movimentos) + bonus(aumenta conforme o decorrer do algoritmo).
4. Empate: aptidão +30 + bonus(aumenta conforme o decorrer do algoritmo).
5. Jogada da Rede que Bloqueia uma Potencial Vitória do Oponente: aptidao + 50.
6. Jogada da Rede que Pode Ganhar na Proxima Rodada: aptidao + 20.

A jogada em posição livre tem um valor mais significante com o decorrer do jogo visto que é mais difícil da rede jogar na posição correta quando há menos posições livres. O oposto se aplica em relação a vitória, visto que uma vitória em poucos movimentos é melhor.

## 2.2. Seleção

A seleção é realizada utilizando elitismo e torneio. Após realizado o cálculo de aptidão se seleciona o cromossomo que obteve a maior aptidão e se copia ele para a nova população na linha 0 dela (elitismo). Esse cromossomo não será resultado de cruzamento e não sofrerá mutação.

Para o torneio, se seleciona duas vezes dois cromossomos aleatoriamente e se armazena o que tem a melhor aptidão dentre esses, obtendo dessa forma o pai e a mãe para, então, realizar o cruzamento. Segue um exemplo do torneio:

|  |  |
| --- | --- |
| CX\_1: Aptidão = 55.0 | CX\_2: Aptidão = 10.0 |
| CY\_1: Aptidão = 35.0 | CY\_2: Aptidão = 70.0 |
| Pai: CX\_1 | Mãe: CY\_2 |

## 2.3. Cruzamento

O cruzamento do pai e da mãe é feito for média aritimética dos valores na mesma posição de ambos os pais, gerando, dessa forma, um único filho. Segue um exemplo com uma quantidade pequena de genes:

|  |  |  |  |
| --- | --- | --- | --- |
|  | P1 | P2 | P3 |
| Pai | 5 | 10 | 6 |
| Mãe | 3 | 2 | 8 |
| Filho | (5+3)/2 = 4 | (10+2)/2 = 6 | (6+8)/2 = 7 |

A frequência do processo de cruzamento depende da taxa de cruzamento, para uma taxa de 0.9, por exemplo, a cada seleção de pais existe uma chance de 90% de se realizar o cruzamento, caso isso não aconteça, é copiado para a nova população o pai ou mãe com a melhor aptidão. Esse processo de torneio e cruzamento se repete até se completar a geração de uma nova população que poderá sofrer mutação.

Para o trabalho foi utilizada uma taxa fixa de crossover de “0.86” (86%), pois tal valor forneceu os melhores resultados.

## 2.4. Mutação

Devido a alta quantidade de pesos, a mutação possui uma chance aleatória de ser realizada por gene, ao invés de um potencial gene para a população inteira, tal abordagem permite evitar uma convergência prematura. Após se criar a população nova, caso ocorra mutação sobre um gene, o valor desse gene é alterado para um valor aleatório de -1 a 1.

No caso, a frequência da mutação está associada a taxa de mutação, sendo que uma taxa de mutação alta implica em mutações frequentes e caso contrário, mutações infrequentes. Por exemplo, para uma taxa de mutação de 0.05, a chance de mutação é de 5% por gene, caso contrário, não nenhuma alteração na população.

Para o trabalho, foi utilizada uma taxa de mutação inicial de “0.01” (1% por gene) que decresce com o decorrer do algoritmo, isso ajuda a manter os pesos adequados conforme ocorre uma convergência da aptidão do algoritmo.

# 3. Treinamento por Aprendizado por Reforço

Primeiramente se inicializou a rede com valores aleatórios para os cromossomos e inicializando sua aptidão em 0. Após isto se inicia o loop de treinamento do jogo da velha por um dado número de iterações, sendo que a cada iteração se utiliza uma nova população que foi alterada pelo algoritmo genético.

Cada cromossomo joga uma partida e sua aptidão será pontuada com o decorrer dela. Caso o jogo acabe, seja por vitória, derrota, empate ou jogada inválida (posição ocupada), a aptidão calculada até esse ponto é salva para aquele cromossomo.

A cada jogada que não é da rede, dois tipos de jogadas podem ser realizadas: jogada aleatória em posição válida ou jogada pelo algoritmo de minimax com base em uma taxa de frequência que aumenta com o decorrer do treinamento. Logo, a “dificuldade” do jogo aumenta conforme a rede vai sendo aprimorada.

O processo se repete para um dado número de iterações determinada pelo usuário até elas acabarem. Ao término do treinamento, é selecionado o cromossomo que possui a melhor aptidão da última população e salvo em um arquivo texto, para ser utilizado posteriormente, caso necessário.

# 4. Front End

Como Front End foi criado uma representação simples do tabuleiro a cada turno de cada jogador, se iniciando pelo jogador “O” que é o usuário. Após iniciado o jogo, cada jogador (usuário e rede) deve posicionar, seja “X” (rede) ou “O” (usuário), em uma posição válida não ocupada, o jogo para se for uma posição inválida.

Segue um exemplo do menu inicial e a seleção da opção de treinar a rede:

A screenshot of a computer

Description automatically generated

Opção 1 do Menu

Ao selecionar essa opção o usuário determina um número de iterações para o treinamento, ao final do treinamento, é salvo os pesos resultantes em um arquivo e apresentados no terminal para o usuário:

A picture containing text, font, screenshot

Description automatically generated

Salvo os Pesos da Rede em um arquivo

A segunda opção do menu permite que o usuário jogue com a rede treinada de forma livre, iniciando pelo usuário. O jogo continua até algum jogador vencer ou algum jogador selecionar uma posição inválida, segue uma vitória do usuário e uma situação que a rede bloqueia a vitória do usuário (jogador “O”):

A black screen with white text

Description automatically generated with low confidence

Vitória do Usuário

A screenshot of a computer

Description automatically generated with medium confidence

Bloqueio pela Rede

Para evitar o treinamento a cada execução o menu também possui a opção de “Carregar Pesos Pré-Calculados” que utiliza pesos, salvos em um arquivo, que já tiveram um bom resultado, em execuções prévias do algoritmo.

# 5. Resultados e Testes

Foi observado durante o treinamento da rede uma grande oscilação da aptidão dos cromossomos com o decorrer das gerações dos algoritmos genéticos, para minimizar esse problema, foram definidas outras formas de pontuar a aptidão como bloqueio de uma jogada de vitória e uma potencial vitória da rede. No entanto, apesar de terem sido realizadas diversas alterações nos parâmetros do algoritmo genético, no cálculo da aptidão e na topologia da rede, o resultado final, apesar de ter melhorado consideravelmente, não obteve um resultado satisfatório.

Uma possível razão para isso pode ser a limitação na forma de pontuação do algoritmo, devido à natureza do jogo da velha. Por se tratar de um jogo com poucas configurações possíveis do tabuleiro, é provável que o algoritmo tenha tido dificuldade em discernir entre uma configuração de rede (pesos) boa e uma ruim. No trabalho, foram implementadas inicialmente todas as formas possíveis de pontuação, porém, mesmo assim, o algoritmo não conseguiu alcançar o resultado esperado.

# 6. Considerações Finais

Com a realização deste trabalho conclui-se que é necessário considerar diversos aspectos para implementar a solução de IA ncessária. Se deve considerar não apenas como o algoritmo em si funciona, mas os seus parâmetros que podem influenciar drasticamente o resultado final da solução, como taxa de crossover e mutação. Além disso, é importante utilizar uma estrutura de dados que seja compatível e apropriada para o algoritmo genético, assim como um cálculo de aptidão que seja razoável com o resultado que se deseja obter.

A implementação de diversos conceitos de IA, como algoritmo de busca, rede neural e aprendizado por reforço, permitiu um maior entendimento dos diferentes aspectos envolvidos na construção de uma solução de IA. Apesar das dificuldades encontradas ao longo do desenvolvimento do algoritmo, esse trabalho possibilitou uma aprofundada desses conceitos em um sistema de IA..

No decorrer do trabalho, a principal dificuldade foi estabelecer um cálculo de aptidão preciso que permitisse que a rede aprendesse corretamente, minimizando a ocorrência de erros. Apesar dos esforços e testes para determinar parâmetros adequados para o algoritmo genético e estabelecer um valor de aptidão razoável para as várias possibilidades de jogadas da rede e configurações de tabuleiro, o resultado final do treinamento ficou abaixo do esperado.

# 7. Referências

* Algoritmos Utilizados em Aula.
* SUTTON, Richard S., BARTO, Andrew G. Reinforcement Learning: An Introduction. 2nd ed, Cambridge, MIT Press, 2018. 548p.
* Wikipedia. Mutation (Genetic Algorithm). https://en.wikipedia.org/wiki/Mutation\_(genetic\_algorithm). Acesso em 25/06/2023.