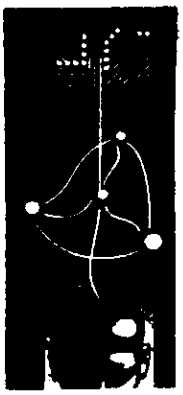

1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

Implementação de um Jogo de Damas Utilizando uma Rede Neural Multi-Camadas Associada a uma Heurística.

Paulo André S. Perez
USP - ICMSC -SCE
13560-150 - São Carlos - SP - Brasil
e-mail: perez@icmsc.usp.br

Roseli Ap. Francelin *
USP - ICMSC - SCE
13560-150 - São Carlos - SP -Brasil
e-mail: rafrance@icmsc.usp.br

Maria Carolina Monard *
USP - ICMSC - SCE
13560-150 - São Carlos - SP - Brasil
e-mail: monard@.icmsc.usp.br

Sumário

Neste trabalho discutimos a união da Programação Simbólica à Conexionista na implementação de um Jogo de Damas. Uma heuristica é proposta de forma a encontrar o melhor movimento a ser efetuado pelo computador, isto em conjunto com uma Rede Neural Multi-Camadas que é treinada e utilizada para estimar quais serão os movimentos do usuário. O modelo é analisado, tendo como foco as estruturas de dados, a heurística adotada, a Rede Neural, o processo de aprendizagem e o algoritmo que gera, com base na saída da Rede, os vários possíveis movimentos do usuário.

Uma discussão sobre a capacidade de aprendizagem do Jogo, é feita a partir dos resultados obtidos. Sugestões para futuros trabalhos são feitas.

* Trabalho desenvolvido com o apoio da FAPESP

I - INTRODUÇÃO

Tipicamente vemos o conflito entre a os adeptos da Inteligência Artificial Simbólica e os da Conexionista. Contudo é fato que ambas buscam a representação do conhecimento e o processo de aprendizagem. cada uma apresentando pontos fortes e fracos em determinadas aplicações. Nada mais natural então. do que propor a união destas de forma a resolver um dado problema. Como exemplo. dada a facilidade de implementação. enfocaremos um Jogo de Damas. onde teremos um heurística que encontrará quais são os possiveis movimentos de um jogador. e dentre eles escolherá o melhor com base no contra-movimento a ser efetuado pelo usuário. o qual será estimado por uma Rede Neural Multi-Camadas.

Este trabalho. é organizado com segue. Na Seção II, propomos a Rede Neural Multi-Camadas e discutimos sua utilização. Na Seção III. mostramos o algoritmo da heuristica adotada. Na Seção IV. é abordada a implementação. Finalmente. conclusões e futuros trabalhos são apresentados na Seção V.

II - A REDE NEURAL

Redes Neurais Artificiais (RNA) possuem uma grande diversidade de aplicações. dentre elas: processamento de sinais [1], reconhecimento de padrões [1-2] e otimização [3-5]. Dentre as RNA. destacamos a chamada Rede Neural Multi-Camadas (Figura 1). isto dada a sua capacidade de aprendizagem e generalização.

Utilizaremos uma RNA Multi-Camadas para representar o conhecimento do usuário no Jogo de Damas. de tal forma que as entrada da Rede correspondem a atual disposição das pedras no tabuleiro. e a saída. o movimento que o usuário faria diante desta situação (Figura 2).

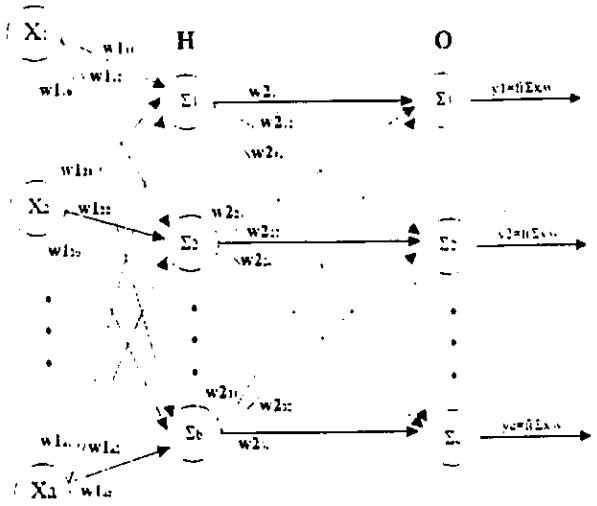

Figura 1 - Rede Neural Multi-Camadas

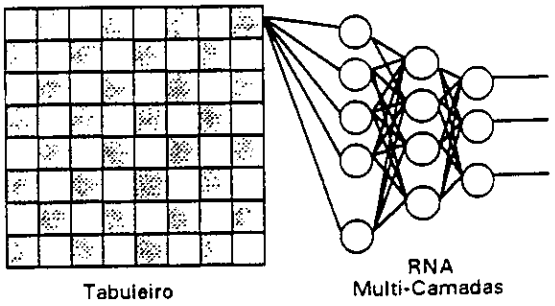

Figura 2 - Modelo Adotado

Foi adotada uma matriz 8x8 de números inteiros. doravante denota por T. para representar o tabuleiro. onde um elemento pode assumir um dos seguintes valores: 0 para posição vazia; 1 para posição ocupada por uma peça do jogador número 1; 2 para peça do jogador número 2; -1 para representar uma *dama* do jogador número 1; e -2 para uma *dama* do jogador número 2.

A matriz T corresponde à entrada da Rede, ou seja, todos os elementos da matriz estão ligados a todos os nós da primeira camada oculta da Rede e vice-versa.

O movimento do usuário é composto por dois pares de coordenadas, que correspondem à linha/coluna inicial e final da peça a ser movida. Como as saídas da Rede. que no nosso modelo utiliza a função sigmóide. estão entre 0 e 1. é necessário que codifiquemos estas coordenadas em seus equivalentes binários (000 a 111). Logo teremos com saida um vetor M de 12 elementos

Desta forma fica estabelecido que temos uma Rede com 64 entradas. 2 camadas ocultas e

12 nós na camada de saída. As quantidades de nós nas camadas ocultas foram estabelecidas com base em testes, adotando-se então 48 nós na primeira camada oculta e 24 na segunda.

Basicamente o que desejamos da Rede é tentar estimar os movimentos do usuário no jogo, ou seja, dada uma matriz **T**, obter como saída o movimento(s) que o usuário fará. Para conseguir isto, devemos treinar a Rede com o maior número possível de pares **T,M** , que são obtidos ao longo de partidas anteriores entre o computador e o usuário em questão. Assim, teremos não só uma "memória" das partidas anteriores, como também uma previsão de possíveis movimentos, isto graças à generalização oferecida pela Rede.

Contudo existe um problema, neste caso específico, que é o fato de que uma RNA Multi-Camadas, num dado instante, dada uma única entrada, matriz **T**, temos somente uma única saída, vetor **M**. Como num jogo, existem diversos movimentos possíveis, e também objetivando explorar ao máximo a capacidade de abstração da Rede, definimos uma função que com base na saída gerada pela Rede, obtém todos os possíveis movimentos derivados da mesma, de acordo com o que se segue:

- Para cada nó de saída temos que o respectivo bit assumirá o valor:

- 0 , se a saída for menor ou igual a 0.1;
- 1 , se a saída for maior ou igual a 0.9;
- 0 e depois 1 , se a saída estiver entre 0.1 e 0.9, ou seja, se tivermos uma saída neste intervalo, primeiro geraremos um movimento assumindo que este bit é 0 e depois outro assumindo que o mesmo bit é 1.

Definimos ainda, um grau de certeza, ou razão de certeza (*ratio*) que corresponderá à confiança que temos deste movimento ser provável, e que é dada por:

$$Ratio = \prod_{i=1}^{12} p_i$$

onde

$p_i = (1 - M_i)$, se for assumido que o bit i é 0;
$p_i = M_i$, se for assumido que o bit i é 1.

Este *ratio* vem do fato de acreditarmos que movimentos já aprendidos pela Rede, e que portanto com pequeno erro, provavelmente ocorrerão novamente, ao passo que situações "desconhecidas" gerarão movimentos "estimados", com valores entre 0.1 e 0.9 , e que portanto são menos prováveis de ocorrer.

Tendo então o conjunto de todos os movimentos, podemos adotar como critério, escolher aquele com maior *ratio* como sendo o movimento que o usuário irá realizar.

Para o aprendizado da RNA Multi-Camadas, adotamos o algoritmo básico proposto por Rumelhart[6], e objetivando uma maior velocidade, utilizamos um fator *momentum* dinâmico, além de restringirmos o treinamento à somente os exemplos em que a Rede falha. Alterações estas propostas por Allred[7].

III - A HEURÍSTICA

Uma heurística decide qual é a peça que o computador irá mover. Podemos expressa-la através do algoritmo abaixo:

1. Dado um nível de jogo, encontrar todos os movimentos possíveis de serem efetuados pelo computador, os quais inicialmente tem um *ratio* igual a 1;
2. Contar quantos destes movimentos podem capturar peças do adversário;
3. Para cada movimento, fazer :
3.1 Contar quantas peças o adversário pode capturar antes e depois do movimento, com isso podemos saber se este é um movimento que entrega uma peça, defende outra(s), ou nenhuma das anteriores;
3.2 Fazer temporariamente o movimento;
3.3 Estabelecer um *ratio* inicial para o movimento, que será:
    - 0.01 se é possível capturar alguma pedra do adversário e o movimento não o faz (peça será "soprada");
    - 1.00*Número de capturas, se o movimento captura pedra(s) do adversário;
    - 0.2 se o após o movimento, o número de peças que o adversário pode capturar é maior do que antes;

- 0.8 se após o movimento, o número de pedras que o adversário pode capturar for menor do que antes;
- 0.6 se nenhuma das condições anteriores for satisfeita.

3.4 Se o nível de jogo for superior a 1, e o atual *ratio* do movimento for maior ou igual ao maior *ratio* até o momento encontrado, fazer:

- Solicitar da Rede qual será o próximo movimento a ser efetuado pelo usuário;
- Determinar o *ratio* do movimento do usuário, com base no que foi definido em 3.3;
- O novo *ratio* do atual movimento do computador será dado por:

  *ratio_anterior * (1 - ratio_do_usuário)*
- Se o novo *ratio* ainda for igual ou superior ao maior *ratio*, então decrementar o nível em uma unidade, recursivamente determinar o próximo movimento, multiplicar o atual *ratio* pelo o do próximo movimento, e voltar ao passo 3.4 .

4. O movimento a ser executado é aquele que obteve o maior *ratio* dentre todos.

Tal heurística além de permitir vários níveis de jogo, que no caso representam limitações a árvore de busca, ainda se utiliza da Rede Neural para antecipar qual será o próximo movimento do usuário.

IV - A IMPLEMENTAÇÃO

Com o objetivo de validar o modelo proposto, foi implementada uma versão do Jogo, escrita na linguagem PASCAL e utilizando os princípios que regem a programação orientada à objetos.

Nesta implementação, tal qual definido anteriormente, o tabuleiro foi representado por uma matriz 8x8 de números inteiros (curtos). O movimento foi definido como sendo um registro composto pelas coordenadas inicias e finais da peça a ser movida, além do campo *ratio*. A Rede foi definida a partir das matrizes W11, W12 e W2, para os pesos entre a entrada e a primeira camada oculta, entre esta e a segunda camada oculta, e entre a segunda camada oculta e a camada de saída, respectivamente. As entradas para treinamento foram definidas como sendo listas encadeadas de vetores de 64 elementos, enquanto as saídas como listas encadeadas de vetores de 12 elementos. Os conjuntos de movimentos também foram expressos por listas encadeadas.

Foram criados para cada usuário 3 arquivos. O primeiro contendo a Rede Neural do usuário, ou seja, as matrizes W11,W12 e W2 para a Rede treinada com seus movimentos. Outro arquivo que corresponde ao *log* da última partida jogada pelo usuário, sendo formado por pares do tipo **T** e **M**, ou seja, matriz tabuleiro e movimento efetuado. E um último arquivo que contém uma base de dados sobre todos os pares **T,M** do usuário. É importante observar que como para uma mesma matriz **T** podemos ter movimentos **M** diferentes, em cada partida, adotou-se o seguinte critério de seleção:

- Havendo dois movimentos diferentes para uma mesma disposição do tabuleiro, será escolhido aquele que pertence a uma partida na qual o usuário ganhou. Caso isto não ocorra, ou ambos implicaram em vitória ou derrota, os mesmos serão mantidos na base de dados com os seus respectivos *ratio's*, contudo na aprendizagem, o erro permitido para estes será maior.

Quando um usuário joga pela primeira vez com o programa, temos que a Rede ainda "não o conhece", desta forma, a heurística utilizada ao invés de consultar à Rede qual o próximo movimento do usuário, consulta a ela mesma, recursivamente, qual seria seu próximo movimento se ela fosse o usuário. O mesmo ocorre quando a Rede gera movimentos inválidos.

Sempre após o término de uma partida, a partir do arquivo de *log*, é atualizada a base de dados do usuário, e então a Rede é submetida ao processo de aprendizagem da mesma. Com isto, a cada partida, aumenta o "conhecimento" que a Rede possui do usuário, e conseqüentemente, sua capacidade de "antecipar" seus movimentos.

V - CONCLUSÕES

Com o propósito de validar o modelo, a implementação citada foi submetida a testes com diversos usuários diferentes, e observamos:

- Inicialmente, isto é, após uma única partida, o índice de acerto da Rede foi em média próximo de 17%;
- Para uma quantidade de movimentos na base de dados acima de 120, algo próximo de 7 partidas, o índice de acerto da Rede foi em média de 83%, não superando este valor;
- Para o caso citado acima, o programa em média ganhou quase todas as partidas disputadas, com raras exceções. Estas observadas quando o usuário apresentava uma grande variação de seus movimentos diante de uma mesma situação.

Os resultados obtidos mostraram-se satisfatórios, vindo de encontro às expectativas. Contudo vale ressaltar trabalhos futuros podem ser desenvolidos tendo em mente as seguintes melhorias:

- Aprimoramento do algoritmo de aprendizagem de forma a obter uma maior velocidade;
- Aprendizado durante o jogo, e não em *batch* como feito atualmente;
- Implementação de outros tipos de Redes Neurais;

VI - REFERÊNCIAS

[1] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. Lang, IEEE Trans. on Acoustics, Speech, and Signal Processing, ASSP-37, March, 1989.
[2] G.A. Carpenter, S. Grossberg, Appl. Opt., 26, 4919 (1987).
[3] J.J. Hopfield, D.W. Tank, Biol. Cybern. 52, 141 (1985).
[4] R.A. Francelin, F.A. Gomide, K. Loparo, Proc. IJCNN'91 - Int. Joint Conf. on Neural Networks, 3, 2639 (1991).
[5] G.A. Tagliarini, J.F. Christ, E.W. Page, IEEE Trans. on Computers, 40, 12 (1991).
[6] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Parallel Distributed Processing : Explorations in the Microstructure of Cognition", Vol 1, MIT Press, 1986.
[7] L. G. Allred, G. E. Kelly, Proc. IJCNN'89 - Int. Joint Conf. on Neural Networks, 1, 721 (1989).