Connect 4 with Modified A* and a Feed-Forward Neural Network

By:
Taylor Harvin

Outline

- Connect Four
- A* Usage
  - Node Generation Method
  - Heuristic and Step Cost used
    - General structure
    - Pros and Cons
  - Required modifications
- Feed-Forward Neural Network Overview
  - General Structure
  - Equations
  - Connection with A* and Connect 4
- Results
- Connect 4 Demo

Connect Four

- Similar to Tic-Tac-Toe
- Goal: Connect 4 of your color in a row.
- 7 by 6 grid
- 42 slots
- Only a maximum of 7 possible moves on a given turn.
- 69 possible 4 in a row spots.

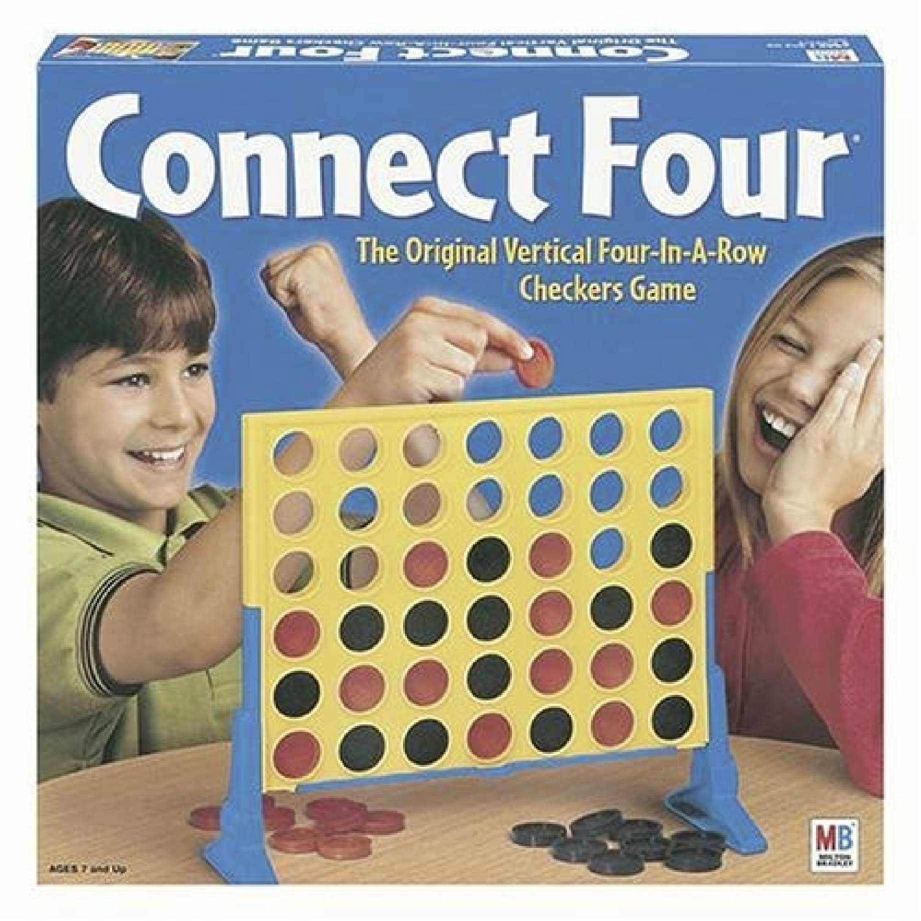
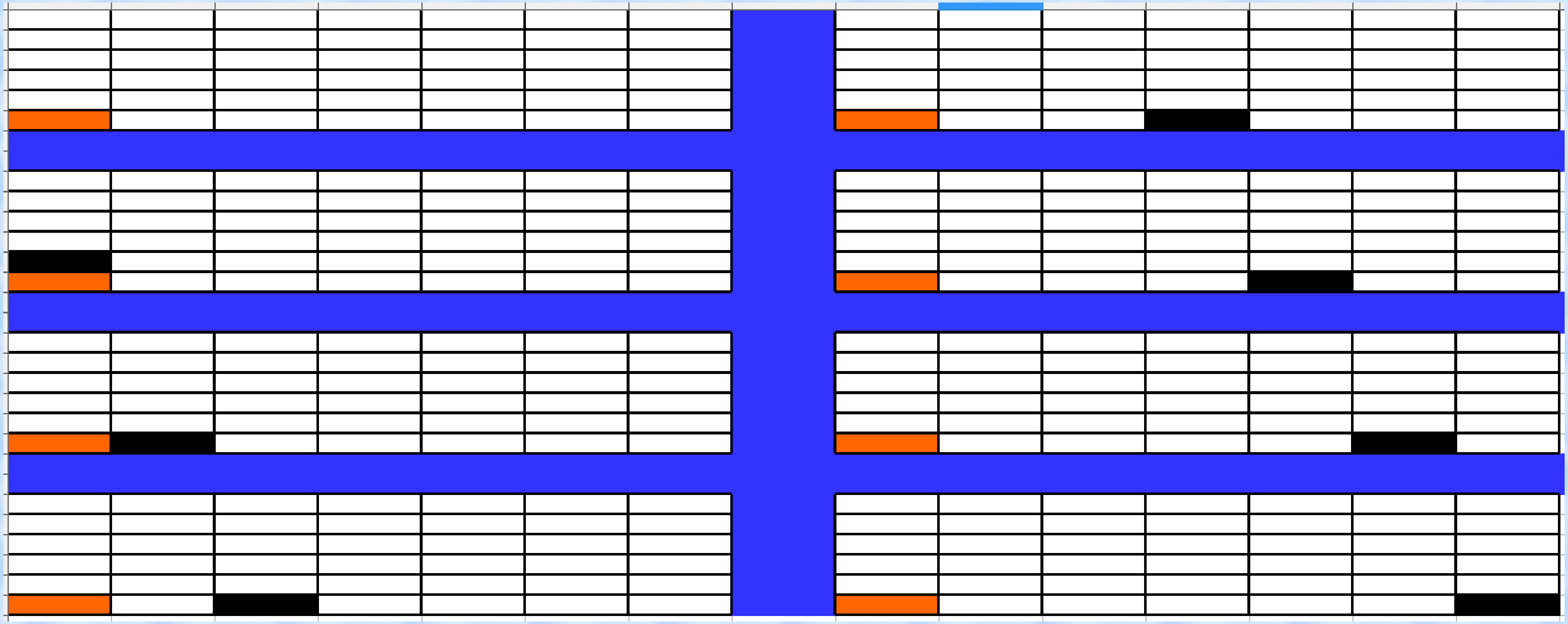


Image borrowed from www.amazon.com

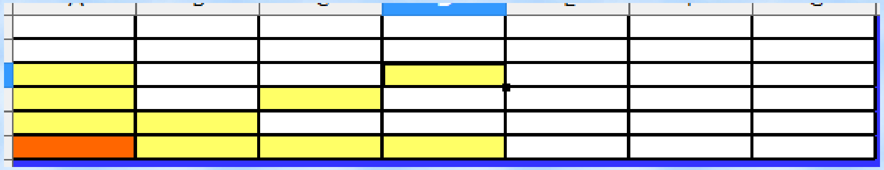
A* Usage

- Player goes first (Top left board—red)
- Below is one expansion example.
  - 7 possible computer moves.



Heuristic and Step Cost

- Step Cost => number of computer moves from the start point.
- **Full Cost Example**
  - Cost of 9 + 3*(2^1) (yellow => empty spots)



🟨			🟨			
🟨		🟨				
🟨	🟨					
🟧	🟨	🟨	🟨			

Heuristic

- Looks at all 69 possible 4 in a row options.
  - 4 Empty => 0
  - Red > 0
    - Red^2 + Empty Count
  - Black > 0
    - Empty Count

Advantages/Problems with the Heuristic and A* Usage

Advantages

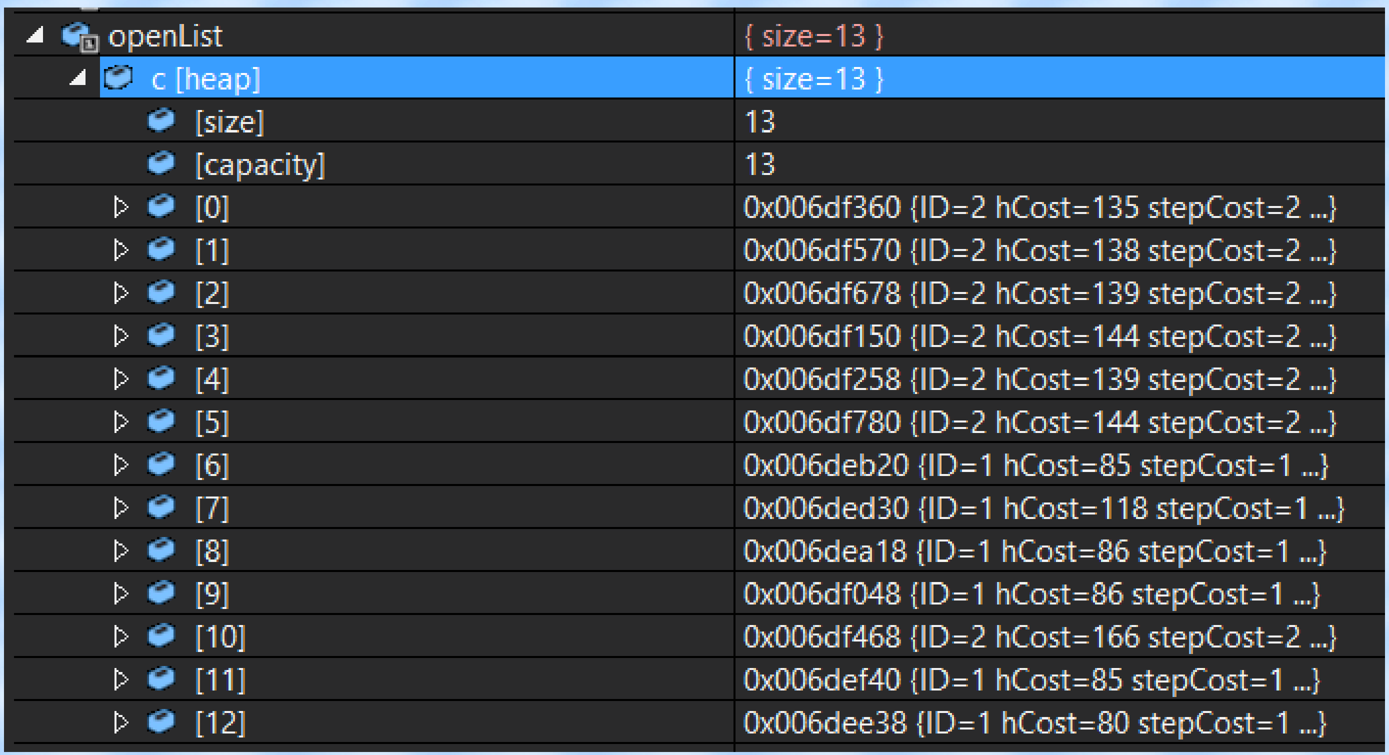
- Provides a relatively accurate description of the state of the board.
- Provides an exponentially growing red flag as the main player gets closer to winning while still maintaining a focus on finding a goal point.

Problems

- Only one expansion level can be performed without a method to guess the player's next move at each state.
- **Even with a guessing method**, each level/move past the start point grows in cost to the point of forcing a breadth first search.
  - Ultimately leading to long processing and bad moves.

Modifications to A*

- Rather than ordering the open list purely by the cost, it also orders by node level.
- Still allows for backtracking to any previous nodes if the level proves to be bad.
- Sort of like an optimized depth first search.
- Options for guessing player move at each node generation:
  - Random guess
  - Feed-Forward Neural Network –> Preferred method

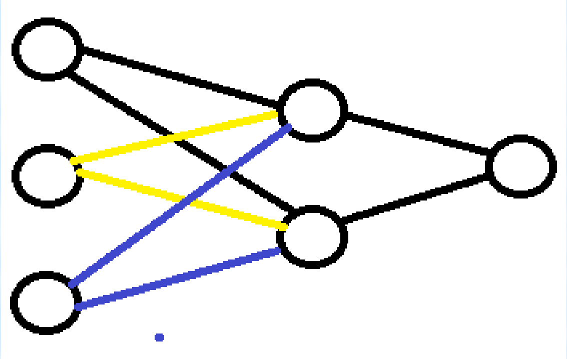


```
◢ openList                 { size=13 }
  ◢ c [heap]               { size=13 }
        [size]             13
        [capacity]         13
    ▷   [0]                0x006df360 {ID=2 hCost=135 stepCost=2 ...}
    ▷   [1]                0x006df570 {ID=2 hCost=138 stepCost=2 ...}
    ▷   [2]                0x006df678 {ID=2 hCost=139 stepCost=2 ...}
    ▷   [3]                0x006df150 {ID=2 hCost=144 stepCost=2 ...}
    ▷   [4]                0x006df258 {ID=2 hCost=139 stepCost=2 ...}
    ▷   [5]                0x006df780 {ID=2 hCost=144 stepCost=2 ...}
    ▷   [6]                0x006deb20 {ID=1 hCost=85 stepCost=1 ...}
    ▷   [7]                0x006ded30 {ID=1 hCost=118 stepCost=1 ...}
    ▷   [8]                0x006dea18 {ID=1 hCost=86 stepCost=1 ...}
    ▷   [9]                0x006df048 {ID=1 hCost=86 stepCost=1 ...}
    ▷   [10]               0x006df468 {ID=2 hCost=166 stepCost=2 ...}
    ▷   [11]               0x006def40 {ID=1 hCost=85 stepCost=1 ...}
    ▷   [12]               0x006dee38 {ID=1 hCost=80 stepCost=1 ...}
```

Feed-Forward Neural Network

- Similar to Biological neural network.
- 2 to N layers
  - Input
    - Normalized values being analyzed.
  - Hidden (0 to N)
    - First one has every node connected to every input node.
    - Helps with feature detection in different applications.
  - Output
    - Result of the analysis of the input.
    - Connects with every node in either the previous hidden layer nodes or input layer (if 0 hidden layers).
- Every connection has a double weight value.

- Basic Neural Network
  - Each line represents a double weight value.
  - Each node is a value generated from input (except the input layer).



Feed-Forward Neural Network Formulas

- Neuron output calculation

$$f(x_i, w_i) = \phi(\sum_i (w_i \cdot x_i))$$

- Sigmoid Activation Function

$$\phi(x) = \frac{1}{1 + e^{-x}}$$

- Output Layer Deltas

$$\delta_i = (\hat{y}_i - y_i)\phi'_i$$

- Sigmoid Derivative

$$\phi'(x) = \phi(x)(1 - \phi(x))$$

- Hidden Layer Deltas

$$\delta_i = \phi'_i \sum_k w_{ki}\delta_k$$

- Weight Update

$$\Delta w_{(t)} = -\epsilon \frac{\partial E}{\partial w_{(t)}} + \alpha \Delta w_{(t-1)}$$

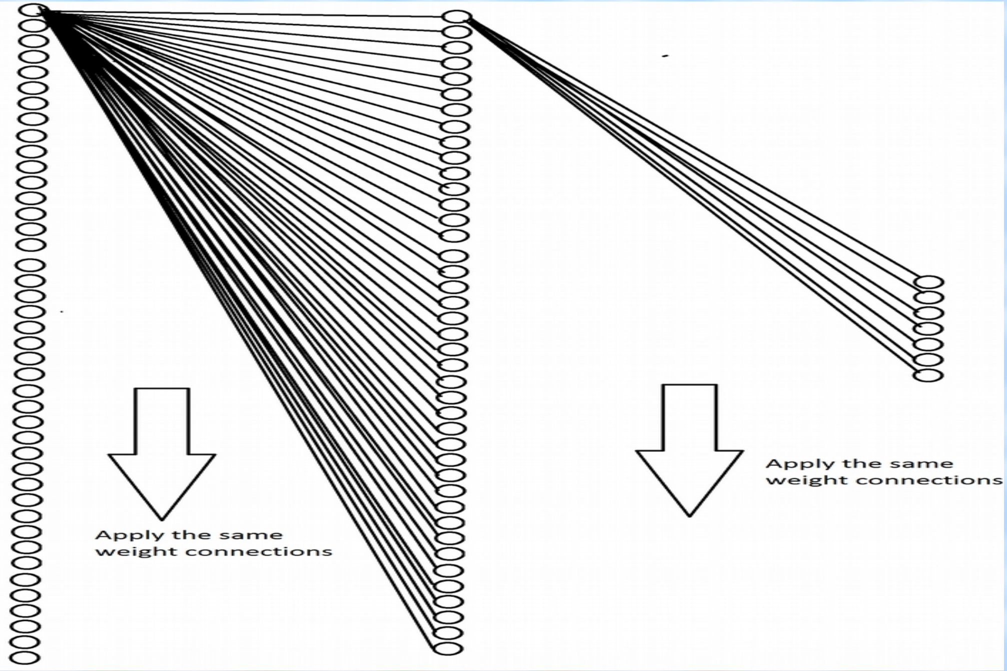
- Global Neural Network Error

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

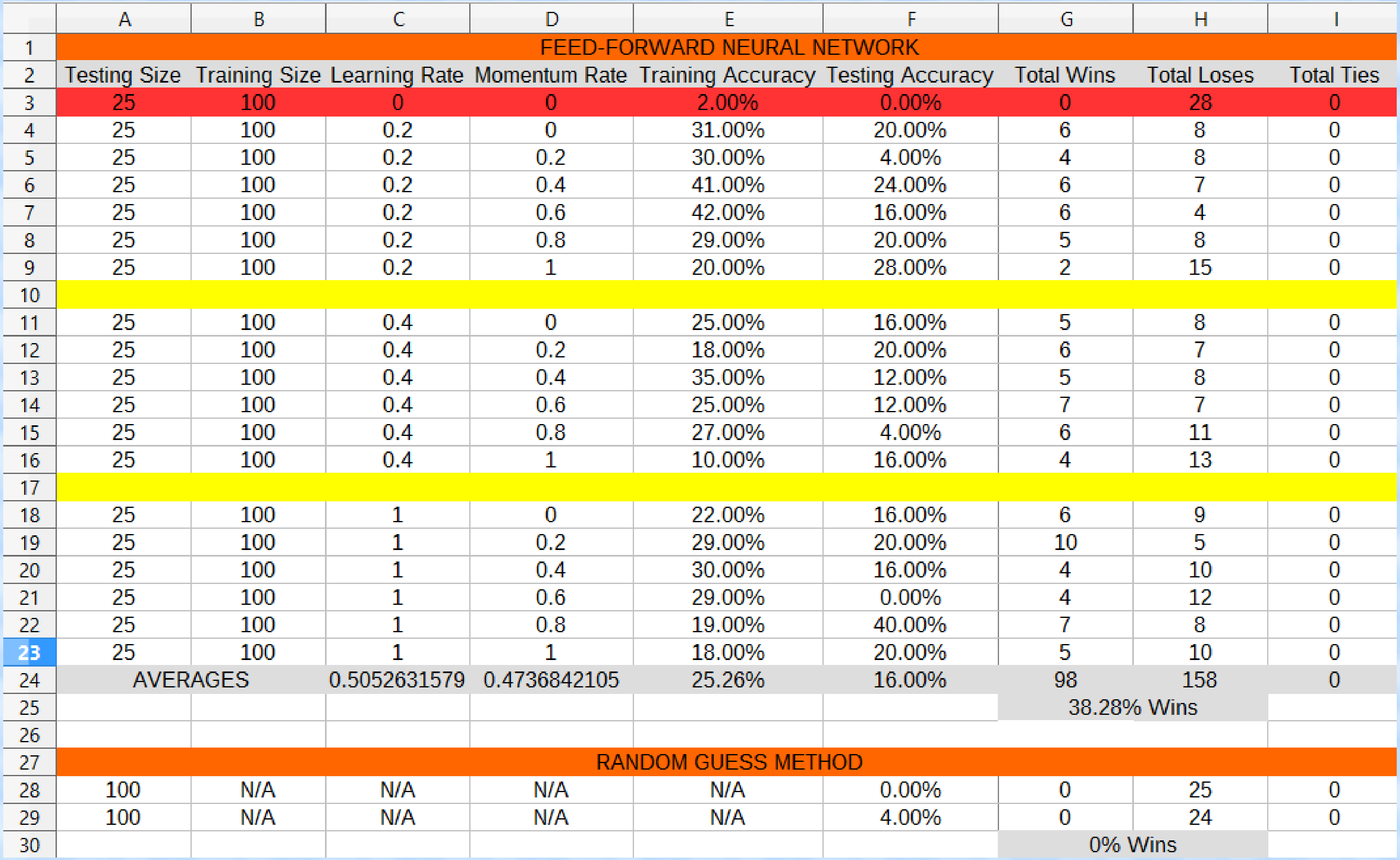
A* and FFNN for Connect 4

- For each node generated in A*, the state of the board generated is fed to the FFNN and a player move guess is applied to the current state before the heuristic cost is generated.
  - If trained properly, provides a very accurate simulation of the rest of the game.
    - Results in a more intelligent move, while still allowing room for realistic bad moves.

- Inputs => all 42 slots
- Outputs => user play guess (7 nodes)
  - Hidden => 41 general nodes
    - Note: new version has no hidden layer


Apply the same weight connections
Apply the same weight connections

Example Results



	A	B	C	D	E	F	G	H	I
1	FEED-FORWARD NEURAL NETWORK								
2	Testing Size	Training Size	Learning Rate	Momentum Rate	Training Accuracy	Testing Accuracy	Total Wins	Total Loses	Total Ties
3	25	100	0	0	2.00%	0.00%	0	28	0
4	25	100	0.2	0	31.00%	20.00%	6	8	0
5	25	100	0.2	0.2	30.00%	4.00%	4	8	0
6	25	100	0.2	0.4	41.00%	24.00%	6	7	0
7	25	100	0.2	0.6	42.00%	16.00%	6	4	0
8	25	100	0.2	0.8	29.00%	20.00%	5	8	0
9	25	100	0.2	1	20.00%	28.00%	2	15	0
10									
11	25	100	0.4	0	25.00%	16.00%	5	8	0
12	25	100	0.4	0.2	18.00%	20.00%	6	7	0
13	25	100	0.4	0.4	35.00%	12.00%	5	8	0
14	25	100	0.4	0.6	25.00%	12.00%	7	7	0
15	25	100	0.4	0.8	27.00%	4.00%	6	11	0
16	25	100	0.4	1	10.00%	16.00%	4	13	0
17									
18	25	100	1	0	22.00%	16.00%	6	9	0
19	25	100	1	0.2	29.00%	20.00%	10	5	0
20	25	100	1	0.4	30.00%	16.00%	4	10	0
21	25	100	1	0.6	29.00%	0.00%	4	12	0
22	25	100	1	0.8	19.00%	40.00%	7	8	0
23	25	100	1	1	18.00%	20.00%	5	10	0
24	AVERAGES		0.5052631579	0.4736842105	25.26%	16.00%	98	158	0
25							38.28% Wins		
26									
27	RANDOM GUESS METHOD								
28	100	N/A	N/A	N/A	N/A	0.00%	0	25	0
29	100	N/A	N/A	N/A	N/A	4.00%	0	24	0
30							0% Wins		

Demo Time

(No Hidden Layer Version)

References

[1] J. Boyan, “Modular Neural Networks for Learning Context-Dependent Game Strategies,” University of Chicago, 1992.

[2] J. Clune, “Heuristic evaluation functions for general game playing,” in AAAI’07 Proceedings of the 22nd national conference on Artificial intelligence - Volume 2, 2007, pp. 1134–1139.

[3] J. Heaton, Artificial Intelligence for Humans Volume 3: Deep Learning and Neural Networks, 1.0 ed. Chesterfield: Heaton Research, Inc., 2015.

[4] B. M. Sathyaraj, L. C. Jain, A. Finn, and S. Drake, “Multiple UAVs path planning algorithms: a comparative study,” Fuzzy Optim. Decis. Mak., vol. 7, no. 3, pp. 257–267, Jun. 2008.

[5] M. O. Schneider and J. L. Garcia Rosa, “Neural Connect 4 - a connectionist approach to the game,” in VII Brazilian Symposium on Neural Networks, 2002. SBRN 2002. Proceedings., 2002, pp. 236–241.

[6] W. Williams, N. Prasad, D. Mrva, T. Ash, and T. Robinson, “Scaling recurrent neural network language models,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5391–5395.

End

Questions?