Reinforcement Learning

Todd Sierens[1,2]

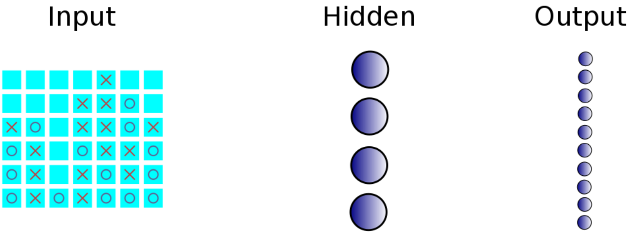
[1]Perimeter Institute for Theoretical Physics
[2]University of Waterloo

May 5, 2016

Outline

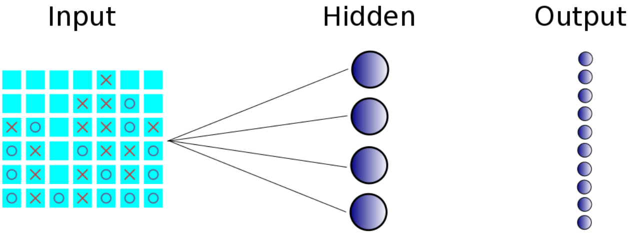
1. Neural Networks
    - Simple Perceptron
    - Multilayer Perceptron
    - Convolutional Neural Network
2. Machine Learning
3. Neural Networks
    - Simple Perceptron
    - Multilayer Perceptron
    - Convolutional Neural Network
4. Temporal Difference Learning
5. results
6. The End

Neural Network



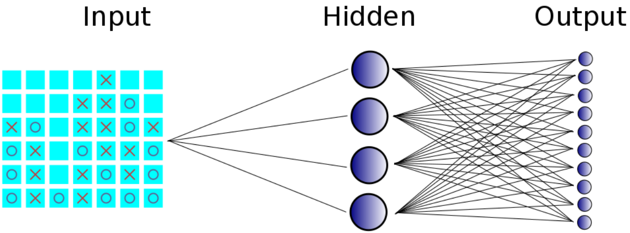
				×		
			×	×	○	
×	○		×	×	○	×
○	×		○	×	×	○
○	×		×	○	×	○
○	×	○	×	○	○	○

Neural Network



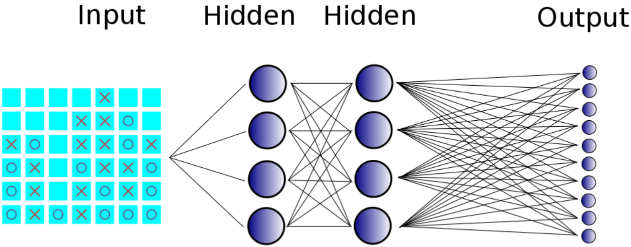
				×		
			×	×	○	
×	○		×	×	○	×
○	×		○	×	×	○
○	×		×	○	×	○
○	×	○	×	○	○	○

Neural Network

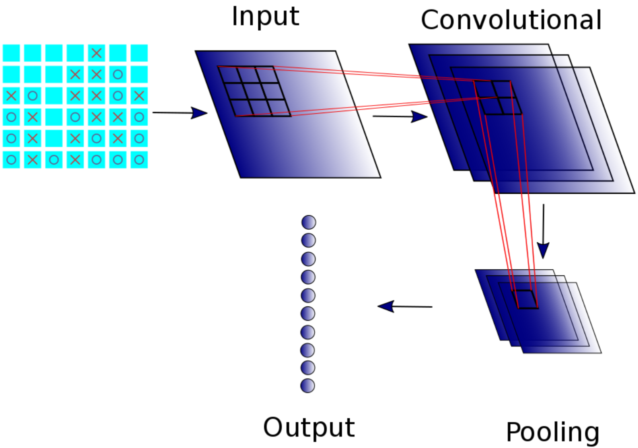


				×		
			×	×	○	
×	○		×	×	○	×
○	×		○	×	×	○
○	×		×	○	×	○
○	×	○	×	○	○	○

Multilayer Perceptron



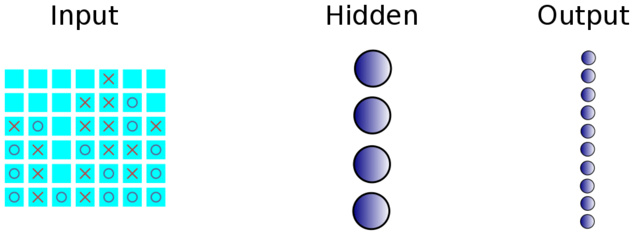
Convolutional Neural Network



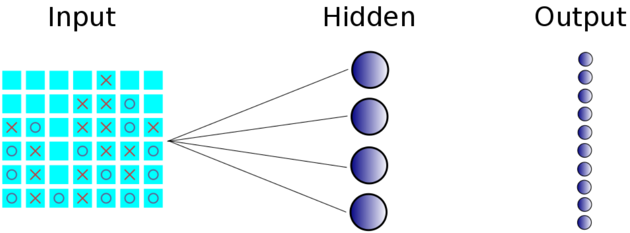
				X		
			X	X	O	
X	O		X	X	O	X
O	X		O	X	X	O
O	X		X	O	X	O
O	X	O	X	O	O	O

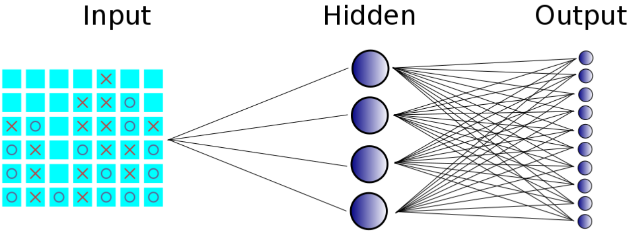
Supervised Learning vs Reinforcement Learning

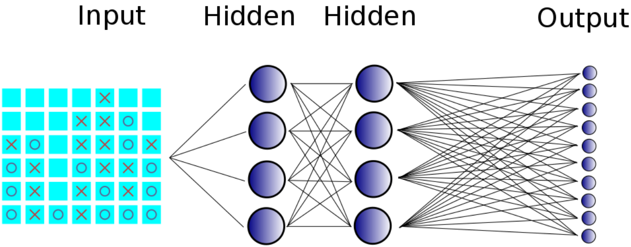
- Supervised Learning
  - Have access to a large set of data with known desired results
  - Adjust model parameters to minimize an objective function
- Reinforcement Learning
  - Have access to an environment that can be modeled
  - Typically a reward function is used as a signal for how to adjust model parameters
  - Board games present a natural environment that is easily modeled and can provide a reward whenever a game finishes.

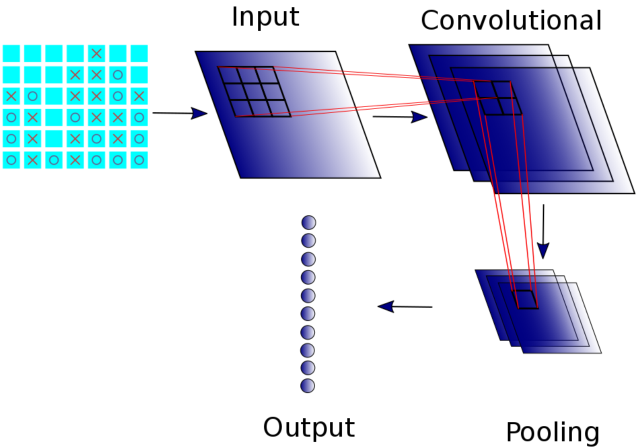


				X		
			X	X	O	
X	O		X	X	O	X
O	X		O	X	X	O
O	X		X	O	X	O
O	X	O	X	O	O	O


Input
Hidden
Output

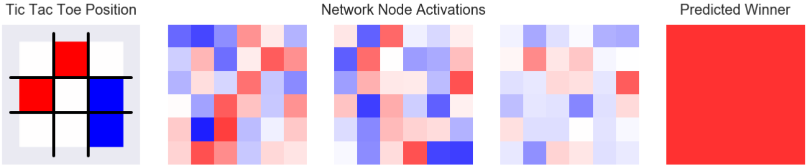

Input
Hidden
Output


Input
Hidden
Hidden
Output


Input
Convolutional
Pooling
Output

Value Network

- A value network is a neural network that evaluates an environment and determines a value to associate to it
- In the case of board games, a value network can be used to determine the probability of winning from any given position
- Here is an example of a multi-layered neural network in action
- The network takes a Tic Tac Toe board as input, and through a succession of node activations the network outputs a prediction on who will win, in this case: the red player



Tic Tac Toe Position

Network Node Activations

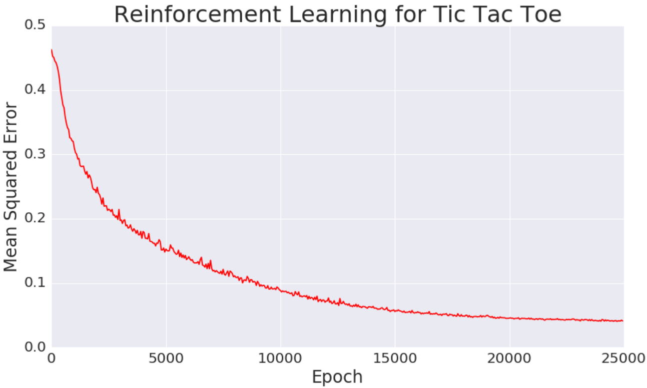
Predicted Winner

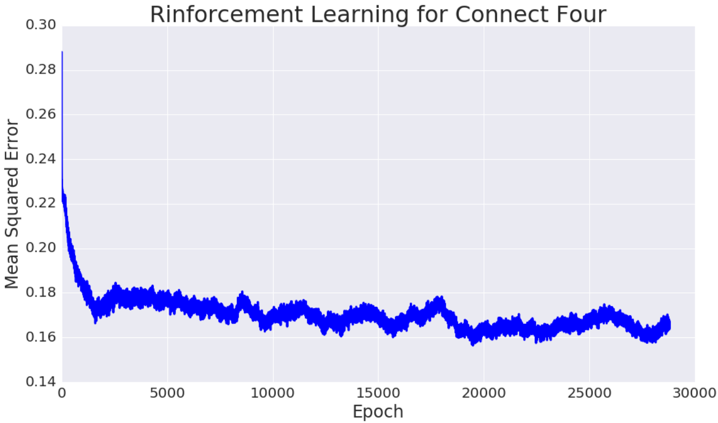
- TD(λ) Equation

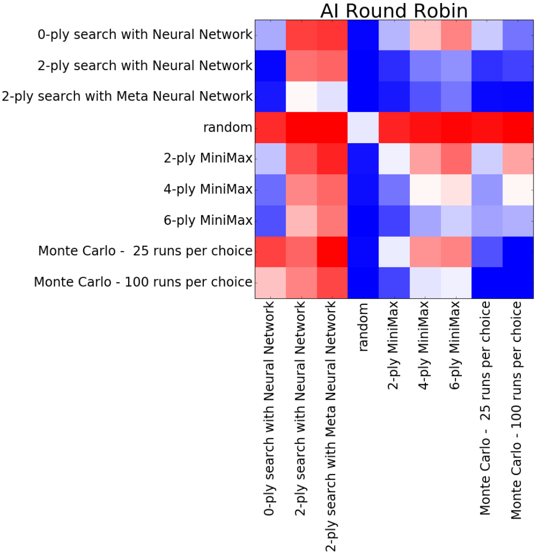
$$T_n = \quad (1-\lambda) \quad V_n + \lambda \quad r$$

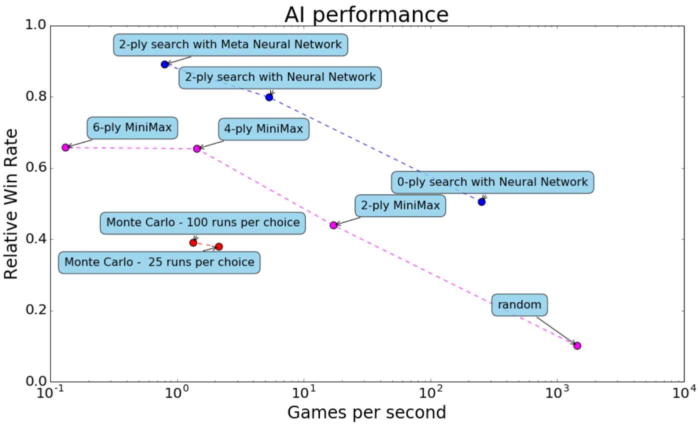
- TD(λ) Equation

$$T_n = \sum_{n=0}^{N_0} (1-\lambda)\lambda^n V_n + \lambda^{N_0} r$$


Reinforcement Learning for Tic Tac Toe
Mean Squared Error
0.0
0.1
0.2
0.3
0.4
0.5
Epoch
0
5000
10000
15000
20000
25000


Rinforcement Learning for Connect Four
Mean Squared Error
0.30
0.28
0.26
0.24
0.22
0.20
0.18
0.16
0.14
Epoch
0
5000
10000
15000
20000
25000
30000


AI Round Robin
0-ply search with Neural Network
2-ply search with Neural Network
2-ply search with Meta Neural Network
random
2-ply MiniMax
4-ply MiniMax
6-ply MiniMax
Monte Carlo - 25 runs per choice
Monte Carlo - 100 runs per choice
0-ply search with Neural Network
2-ply search with Neural Network
2-ply search with Meta Neural Network
random
2-ply MiniMax
4-ply MiniMax
6-ply MiniMax
Monte Carlo - 25 runs per choice
Monte Carlo - 100 runs per choice


AI performance
Relative Win Rate
Games per second
1.0
0.8
0.6
0.4
0.2
0.0
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
2-ply search with Meta Neural Network
2-ply search with Neural Network
6-ply MiniMax
4-ply MiniMax
0-ply search with Neural Network
2-ply MiniMax
Monte Carlo - 100 runs per choice
Monte Carlo - 25 runs per choice
random

The End

The End