MASTERING TIC TAC TOE USING SELF PLAY AND REINFORCEMENT LEARNING

CONSIDERING 3X3 MAZE

MOTIVATION

- In March 2016, Deepmind's AlphaGo beat world champion Go player Lee Sedol 4–1
- 18th October 2017, AlphaGo Zero, that had defeated AlphaGo 100–0
- 5th December 2017, DeepMind released another paper showing how AlphaGo Zero could be adapted to beat the world-champion programs StockFish and Elmo at chess and shogi
- An algorithm for getting good at something without any prior knowledge of human expert strategy was born

OVERVIEW

- Generate training data using the current best player through self play to train the second best and then evaluate it's performance against the best and if it wins 55% of time then replace it with the best for the next iteration

GOAL

- The goal here is to predict two things
- At each point, which move to take which is actually learning the policy
- At each point, what is the immediate reward which is actually learning the value function

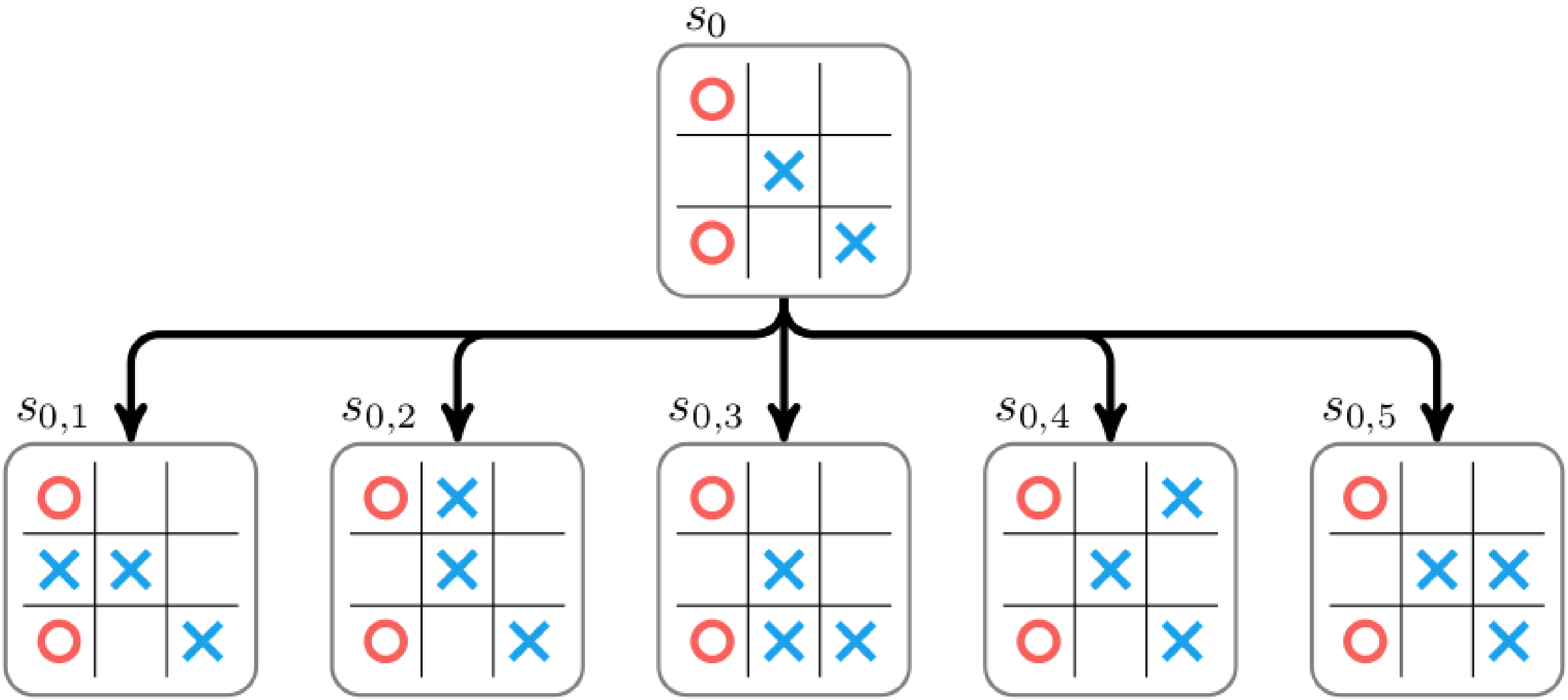
METHODOLOGY

- Self Play
- Network Weights Optimization
- Network Evaluation

SELF PLAY

- Move Selection through Monte Carlo Tree Search
- At each move, the following information is stored
- The game state
- The search probabilities
- The winner

A STATE AND ITS CHILDREN



REPRESENTATION

- 0 represents no move has been played at that position
- 1 represents player 1
- 2 represents player 2

STATE

- At any instance, the condition of the game is a state depending on the representation
- Here, it is the complete maze
  0000000001
  0000000002
- Along with a turn indicator
- Above shown is the initial state of the game under consideration

CHILDREN

- Possible moves from a state are considered as its children
- Children of initial state of the game are

1 0 0	0 1 0	0 0 1	0 0 0	0 0 0	0 0 0
0 0 0	0 0 0	0 0 0	1 0 0	0 1 0	0 0 1
0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0

0 0 0	0 0 0	0 0 0
0 0 0	0 0 0	0 0 0
1 0 0	0 1 0	0 0 1

CHILDREN

- A child stores five values
- N → number of times the action has been taken
- W → total value of the state
- Q → mean value of the state
- P → policy function prediction
- V → value function prediction

MONTE CARLO TREE SEARCH

- Given a state, explore the tree until a leaf node is reached by selecting the best child
- At leaf node, predict probabilities of each of it's children
- Back propagate from the leaf to the root which in this case is the input and update the values of N, W and Q
- Repeat this cycle for a certain time period
- Select the best action among the lot

CHILD SELECTION

- A child is selected based on the equation
- Q+U
- Q $\rightarrow$ The mean value of the state
- U $\rightarrow$ A function of P and N that increases if an action hasn't been explored much, relative to the other actions, or if the prior probability of the action is high

CHILD SELECTION

- U in this case is P/N
- So the equation becomes (P/N)+Q
- Because early on in the simulation, U should dominate (more exploration), but later Q is more important (exploitation)
- So a slight modification is needed which is
- (epsilon*(P/N))+Q

VALUE UPDATE

- During back propagation, the values are updated as follows
- N = N+1
- W = W+V
- Q = W/N

EPISODE

- A game played from start to finish is an episode

DATA GENERATION

- Once an episode has ended, all the steps which were performed will be given labels
- All the steps taken by the winning player will be assigned +1 and to all other steps -1 will be assigned.
- Similarly every move will be assigned a probability provided by the CNN

DEEP NEURAL NETWORK

- It's a network with 1 convolution layer and 40 residual layers with a double headed output, a value head and a policy head
- Value head giving a single prediction in the range [-1,1], actually trying to predict the outcome of the game
- Policy head giving 9 probabilities, actually trying to predict the probability distribution among the children
- For faster convergence, I divided the model in to one each for the different heads

DEEP NEURAL NETWORK

- The loss function in case of value head is mean square error while in case of policy head, it is softmax crossentropy

NETWORK WEIGHTS OPTIMIZATION

- Once data has been generated using the current best player through self play, now it’s time to retrain the second best using the data generated by the best
- Sample a mini batch from the data
- Retrain the current network on these positions

NETWORK EVALUATION

- After training, it's time to evaluate the performance of the retrained network against the best so far and if the retrained network wins 55% of the games, it will become the new best and for the next iteration, roles will be changed as the new best will generate data for training of the previous best

SOME HYPER PARAMETERS

- Number of training epochs : 1
- Number of training iterations : 20
- Number of games during self play : 100
- Number of games during evaluation : 40
- Batch Size : 256
- Sample Size : 512
- Monte Carlo Simulations : 100
- Regularization Constant : 0.0001
- Learning Rate : 0.01
- Momentum : 0.9
- Optimizer : SGD

RESULTS

- After 100 games, defeated untrained algorithm with 40-0
- After 200 games, defeated first best with 40-0