Deep Reinforcement Learning: Tic-Tac-Toe

Kevin Liang

Duke University

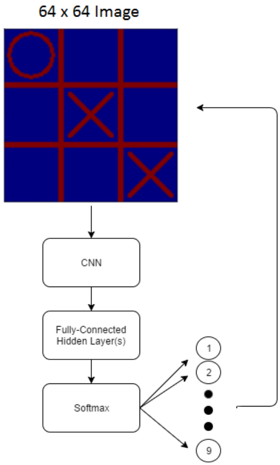
16 September 2016

Introduction

- Goal: Have an agent learn how to play Tic-tac-Toe using deep reinforcement learning
- Tic-Tac-Toe game
  - Players X and O take turns placing marks on a 3x3 grid
  - Player that achieves a 3-in-a-row wins
  - If the board fills up without a winner: draw
- Toy example to get started before tackling more complicated settings (eg. Atari games of OpenAI Gym)

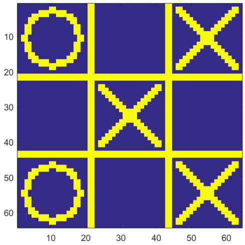
Agent 1: Deep Reinforcement Learner

- Agent is given a 64x64 image of the state of the board, not the 3x3 state space directly.
- Agent not informed of rules. Does not know:
  - what leads to a win/loss
  - own identity
  - marking an already occupied spot is an illegal move
- Use direct policy search (policy gradients) to update weights
- Action stochasticity: multinomial distribution



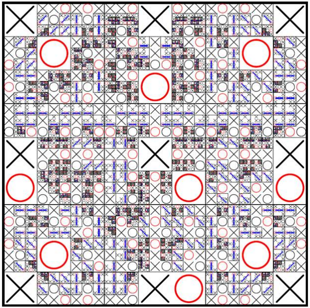
Agent 1: Architecture

- Input: 64x64 image
- CNN: 8 x (3,3), ReLU
  CNN: 16 x (3,3), ReLU
  MaxPool: (2,2)
- CNN: 16 x (3,3), ReLU
  CNN: 16 x (3,3), ReLU
  MaxPool: (2,2)
- CNN: 16 x (3,3), ReLU
  CNN: 16 x (3,3), ReLU
  MaxPool: (2,2)
- Fully Connected: 16 x 8x8 $\rightarrow$ 30, ReLU
- Fully Connected: 30 $\rightarrow$ 9
- Softmax



Agent 2: "Optimal" opponent

- Agent is given the 3x3 state space directly.
- Agent knows rules. Knows:
  - what leads to a win/loss
  - own identity
  - marking an already occupied spot is an illegal move
- Add a "difficulty" parameter that controls how often the "optimal" agent makes a random (legal) move, instead of always following rules
  - Allows for the deep RL agent to occasionally win



Agent 2: "Architecture"

Newell and Simon Tic-Tac-Toe Rules:

- Rule 1: If agent can win, then make a 3-in-a-row
- Rule 2: If opponent can win, block the winning move
- Rule 3: Make a fork (two 2-in-a-rows)
- Rule 4: Block an opponent's fork, while simultaneously making a 2-in-a-row if possible
- Rule 5: Take the center
- Rule 6: If opponent has a corner, take the opposite corner
- Rule 7: Take an empty corner
- Rule 8: Take an empty side

Reinforcement Learning Set-up - Agent 1's perspective

- *State*: 64x64 image of the board
- *Action*: 1 of 9 moves corresponding to the 9 spaces on the board
- *Reward*: Given at the end of a game. One of four outcomes:
  - Win: *+1* - The agent successfully made 3-in-a-row
  - Draw: *0* - The board filled up without either player winning
  - Loss: *-1* - The opponent made 3-in-a-row
  - Broken: *-10* - The agent broke a rule by trying to play a symbol where one already had been placed

Training Regimen

```
Initialize net params θ randomly;
while not converged do
    Initialize images x, actions a, labels l, durations t, player
     identities z to [ ];
    for k = 1, ..., M do
        x_k, a_k, l_k, t_k, z_k = Play game with θ held constant;
        if Broken Rule then
            Discard all but last frame
        end
        Append frames of game k to x, a, l, t, z;
    end
    loss = f(x, a, l, t, z);
    θ = ADAM(loss, θ);
end
```

Loss Expression

Player 1: Deep Reinforcement Learner Player 2: Optimal Player

- Loss for Player j, $j \in \{1, 2\}$:

$$loss_j = -\frac{1}{N} \sum_{i \in z_i = j} \gamma^{t_i} p(y_i = a_i | x_i) r_{l_i} \tag{1}$$

- Total loss (zero-sum game):

$$loss = loss_1 - loss_2 \tag{2}$$

x_i = image of board in frame i

a_i = action taken after frame i

r_l = reward of game result l

z_i = player presented with frame i

N = # of frames in training set

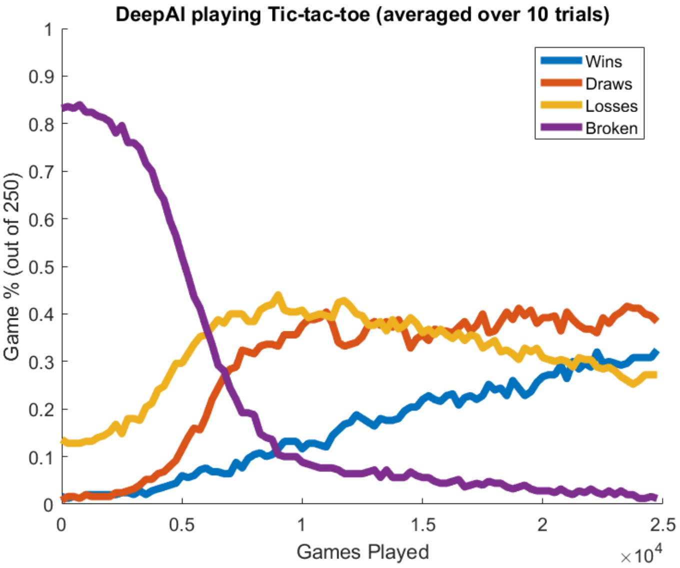
y_i = output of network softmax after frame i

l_i = eventual game result (W, D, L, B) of frame

t_i = duration of game that frame i is part of

γ = discount factor

Results - AI.X = 0.5; AI.O = 0.6



Next Steps

- DGDN Decoder to utilize discarded broken frames
- Knowledge transfer to other tasks:
  - Digital Mammography DREAM Challenge
  - TSA Airport Scanners
- Future RL explorations (and exploitations)