2-7 Triple Draw Poker: With Learning

Nikolai Yakovenko

2/18/15

EE6894 Deep Learning Class

Overview

- Problem: learn strategy to play 2-7 triple draw poker
- Data: play against existing C-lang program that plays pretty good & really fast
- Why: poker is played by lots of people and for high stakes
- Why learning: heuristic-based algorithms only play so well… and don’t adjust to small changes in the game rules. Which happens a lot.
- Speculate: reinforcement learning (Q-learning) to learn policy for optimizing reward function. Neural net layer between raw inputs and Q-learning algorithm.

2-7 Triple draw

Best hand

Typical hand

Draw cards three times, to make 5 different low cards.

Sample Hand

Opponent does same, and final hands compared.

Why Poker?

- There are 10x games played with same mechanics, different rules
  - Winner for high hand
  - Winner for low hand
  - Winner for badugi
  - Split pot game (½ low hand, ½ badugi)
- Also variations in betting, number of players at the table, etc.
- Could we re-use original problem setup, but learn totally different strategy for each variant?

Poker Data

- Algorithm can play itself.
- Also, I have C-lang program that plays triple draw
  - Brute force tree search, with optimization
  - Optimizes for average value of final hand
  - All final 5-card hands scored 0-1000 heuristic
- In the real world… sites like PokerStars have billions of hands of real play, for most popular variants.

Keep Game Really Simple

- Triple draw, no betting
  - Reward is +-1 for winning the hand
- Triple draw, automatic bet per round (or can fold)
  - Reward is winning the chips if best hand, or opponent folds
- Can even start with single draw.
- Important thing is setup for learning game strategy, directly from game results.

Relevant Research

- For Poker:
  - PokerSnowie: neural net from game-theory-optimal No Limit Hold'em
  - "Limit Hold'em is Solve" – recent academic result (although possibly not accurate)
  - Can play neural-net limit Texas Hold'em machine for real $ in Las Vegas
  - No deep learning, focus on GTO for Hold'em
- Other games:
  - Backgammon: neural nets dominant since the 1990's
  - Go: recent huge breakthrough vs best human players, using CNN
  - Atari: famous DeepMind paper
  - Flappy Bird: great example of Q-learning, for problem with simpler game state

Speculate on Deep Learning

- Reinforcement Learning (Q-learning, for example) to learn a strategy for optimizing rewards
- This requires representing game state *s* and *s'* with full information about cards, actions
- DeepMind paper shows how to turn raw state into useful representation of *s* through neural net layer
- Also shows how to deal with noisy & delayed "rewards"

Reinforcement Learning: Flappy

Inputs

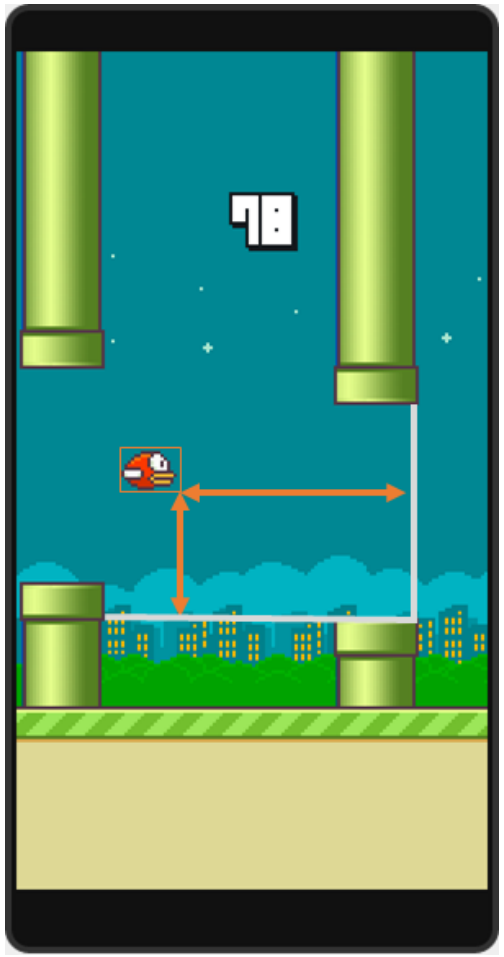
S is a set of states
A is a set of actions
γ the discount
α is the step size

Local

real array $Q[S,A]$
previous state s
previous action a

initialize $Q[S,A]$ arbitrarily
observe current state s
repeat
select and carry out an action a
observe reward r and state s'
$Q[s,a] \leftarrow Q[s,a] + \alpha(r + \gamma \max_{a'} Q[s',a'] - Q[s,a])$
$s \leftarrow s'$ **until** termination

But Flappy State Space is Simpler



- Distance from pipe
- Dead or alive
- Actions: tap or no tap

Conclusion

- Can we simplify poker game, but still keep it the same game?
- And learn a strategy for drawing cards, optimizing the hand, using neural net layer that feeds into reinforcement learning?
- If so… steps to real poker at world-class level, for 100 different game variants… is straightforward.

Thank you!

- Who is interested?
- Flappy Bird result: http://sarvagyavaish.github.io/FlappyBirdRL/
- DeepMind Atari paper: http://arxiv.org/pdf/1312.5602v1.pdf
- 2-7 Triple draw sample hand: http://www.clubpoker.net/2-7-triple-draw/p-263