Poker AI: Equilibrium, Online Resolving, Deep Learning and Reinforcement Learning

Nikolai Yakovenko
NVidia ADLR Group -- Santa Clara CA

Columbia University Deep Learning Seminar
April 2017
Poker is a Turn-Based Video Game
Many Different Poker Games

Single Draw Video Poker

2-7 Lowball Triple Draw
(make low hand from 5 cards with multiple draws)

Limit Hold’em

No Limit Hold’em
World Series of Poker [Humans]
Annual Computer Poker Competition [Robots]
One Hand of Texas Hold’em

Private cards

Hero: Ace of hearts, Queen of spades
Oppn: Ace of spades, King of diamonds

Flop (public)

9 of spades, 6 of spades, 9 of diamonds

Turn

9 of clubs

River

2 of spades

Showdown

Flush: A, Q, 6, 9, 2 of spades
Two Pairs: Ace-Ace, Queen-Queen

Betting Round

Betting Round

Betting Round

Betting Round

Best 5-Card Hand Wins
CFR: Equilibrium Balancing

• Abstract Hold’em game to smaller state-space
• Cycle over ever game states
  – Update “regrets”
  – Adjust strategy toward least “regret”
• Converges to Nash equilibrium in the simplified game.
  – Close enough to an equilibrium in the full game

• Winners of every Annual Computer Poker Competition (ACPC) since 2007.
  – Limit Hold’em: 1% of unexploitable (2015)*
  – No Limit Hold’em: defeated top professional players (2017) **

* pre-computed strategy
** in-game simulation on super-computer cluster
CFR: Counterfactual Regret Minimization

Player 1: Random strategy

Regret: folding good hands
Action: bet good hands more

Regret: not bluffing bad hands
Action: bet when can’t win

Player 2: Random strategy

Regret: not folding bad hands
Action: fold bad hands

Regret: not calling bluffs
Action: call with some % vs bluffs

(equilibrium)
CFR: Pre-Compute Entire Strategy

Each point: encodes game state*
- Private cards for player
- Public cards
- Bets made so far

*Opponent can not distinguish between some states
Entangled Game States: Kuhn (3 Card) Poker Example

Heads-up Limit Hold’em Poker is Solved by Bowling, et al [Nature]
Heads-up Limit Hold’em is Solved

Heads-up Limit Hold’em Poker is Solved by Bowling, et al [Nature]
Within 0.1% of Unexploitable by Perfect Response

Heads-up Limit Hold’em Poker is Solved by Bowling, et al [Nature]
Surprising: Equilibrium Strategy is (almost) Binary

Green: raise; Red: fold; Blue: call
Does this work for No Limit Hold’em?
No Quite: NLH is much bigger

Limit Hold’em
• 2 private cards,
• 5 public cards
• 4 rounds of betting
• 3 betting actions
  – Check/Call
  – Bet/Raise
  – Fold
• $10^{14}$ game states

No Limit Hold’em (200 BB)
• 2 private cards
• 5 public cards
• 4 rounds of betting
• Up to 200 betting actions
  – Check/Call
  – Bet/Raise any size
  – Fold
• $10^{170}$ game states
  – Go has $10^{160}$ game states
Bet Sizes: Huge Branching Factor

DeepStack: ... by Moravcik, et al [Science 2017]
Going Off-Tree

Closest or average known state: errors accumulate
Continuous Re-Solving

DeepStack:
- re-solving

Range: probability vector over 1326 unique private cards

(CMU’s Libratus also employs continuous re-solving)
Re-solving Early: Solve Entire Game (Too Big)

DeepStack:
- depth-limited CFR

Estimate values at depth X with a deep neural network (U of Alberta DeepStack)
Figure 3: Deep counterfactual value network. The inputs to the network are the pot size, public cards, and the player ranges, which are first processed into hand clusters. The output from the seven fully connected hidden layers is post-processed to guarantee the values satisfy the zero-sum constraint, and then mapped back into a vector of counterfactual values.
Good enough for (super) human performance?
**Practical Results: Libratus (CMU) and DeepStack (U-Alberta)**

<table>
<thead>
<tr>
<th><strong>Libratus</strong></th>
<th><strong>DeepStack</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design:</strong></td>
<td><strong>Design:</strong></td>
</tr>
<tr>
<td>- No card abstraction</td>
<td>- No card abstraction</td>
</tr>
<tr>
<td>- CFR+ for preflop and flop</td>
<td>- “Continuous resolving” on all streets</td>
</tr>
<tr>
<td>- &quot;Endgame solving” on turn and river</td>
<td>- Depth-limited solving w/ DNN</td>
</tr>
<tr>
<td><strong>Speed:</strong></td>
<td><strong>Speed:</strong></td>
</tr>
<tr>
<td>- Instant preflop &amp; flop</td>
<td>- ~5-10s preflop and flop</td>
</tr>
<tr>
<td>- ~30 seconds turn and river</td>
<td>- ~1-5s turn and river</td>
</tr>
<tr>
<td>- (200 node super-computer)</td>
<td>- Laptop with Torch and GPU</td>
</tr>
<tr>
<td><strong>Results:</strong></td>
<td><strong>Results:</strong></td>
</tr>
<tr>
<td>- $14.1/hand vs top pros</td>
<td>- $48.6/hand vs non-top pros</td>
</tr>
<tr>
<td>- 120,000 hands over 3 weeks</td>
<td>- Upcoming freezeout matches</td>
</tr>
</tbody>
</table>
Libratus Challenge (January 2017)

The humans lost to Libratus AI beat by 4+ σ. Did they also get tired?
Previous ACPC Agents Were Highly Exploitable (And Maybe Still Are)

Results: 1250 mbb/hand = $125/hand – more than folding every hand.
LBR agent = limited best response using 48 bet sizes.
Conclusions & Speculations (04/2017)

• Computers can match top humans at heads-up No Limit Hold’em poker.
• The winning approach is continuous re-solving (similar to chess or Go, but with hand ranges)
• Great tools to measure exploitability and the luck factor (see DeepStack paper for details)

• Can this approach generalize to 3-6 player games?
• Can the online solving become much faster?
• Will this work always require extensive domain expertise?
Can we train a strong poker player with a much smaller strategy?
**Poker-CNN: Cards as 2D Tensors**

Private cards: [AhQs]  
Flop (public): [AhQs] + [As9s6s]  
Turn:  
River: [AhQsAs9s6s9c2s]  
Showdown: Flush

Flush draw: Pair (of Aces)

Flush!
Convnet: Predict Anything You Want

Inputs:
- Private cards
- Public cards
- Pot size
- Position
- Previous bets history

(31 x 17 x 17 3D tensor)

Predict action value:
- Bet, call, fold values
- Action probabilities
- Value by bet size

Surrogate tasks:
- Allin odds
- Opponent hand distribution

Outputs:
- single-trial $ win/loss
- no gradient for bets not made
- no Monte Carlo tree search required
Big Blind: $100
Small Blind: $50

Raise: 81.1%  
Call: 18.9%  
Fold: 0.0%

+$265
+$2616

Odds vs Opponent

Bet Size

$20,000
$20,000
**Odds vs Opponent**

- Bet: 59.0%
- Check: 41.0%
- (Check) +$3,967

**Bet Size**

- 20% pot
- 50% pot
- 1x pot
- 1.5x pot
- 3x pot
- 10x pot

---

**Summary**

- Initial pot: $5,430
- Bet: 86.6%
- Check: 13.4%
- Result: +$3,967
- Final pot: $17,285

---

**Additional Notes**

- The graph on the left shows the distribution of odds against the opponent at different betting percentages.
- The bar chart on the right indicates the betting size proportions for different pot values.
Raise 26.6%  
Call 73.4%  
Fold 0.0%  

Value vs random 91.3%  
Value vs oppon 68.0%  

($13,000 allin call, to win $26,000)  
33.3% odds = break-even
Takeaways

• Pretty good pattern matching, with enough data
  – Naïve network design – and foolish use of pooling
  – Training ≈ 4 million previous ACPC hands

• Struggles with rare cases
  – Under-weights outliers
  – Out of sample situations

• Struggles in big pots
  – Large effect on average results
  – Sparse data

• No attempt to avoid exploitability
Future Work

• More games, more contexts
  – 3-6 player No Limit Hold’em
  – Pot-Limit Omaha (4 private cards instead of 2)
  – Tournament Hold’em
• Learn the CFR internal parameters?
  – Predict opponent hand ranges directly
• Personalize model against an opponent
• Tune hyper-parameter...
  – 100,000 hands per experiment
  – Find ideal network arrangement
  – Exploit flexibility of deep neural nets
Can a DNN learn to imitate strong players in 2+ player games?

Data: high-quality simulation, equilibrium solving, or player logs
Reinforcement Learning
Deep Q-Learning for Atari Games

*Human-level control through deep reinforcement learning* by DeepMind (Nature 2015)
OpenAI Gym: Train & Share RL Agents

Support for Atari games, classic RL problems, robot soccer, Doom [no poker]
**Reinforcement Learning for Games**

**objective:** learned policy maximizes future rewards

\[ R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} , \]

- discount factor \( \gamma \)
- reward change at time \( t' \) \( r_{t'} \)

Reinforcement learning

https://www.slideshare.net/onghaoyi/distributed-deep-qlearning
Faulty Reward Function?

https://blog.openai.com/faulty-reward-functions/
Can Poker Be Solved With RL?

• Yes, with modifications.
  – “Standard” RL is greedy and requires the Markov property
  – Poker decisions can’t be optimized locally
  – Some game-custom local simulation is required

• Heinrich & Silver (DeepMind 2016) match state of the art on Limit Holdem with modified deep RL
  – Deep RL also gives useful “similar context” embedding for poker situations
  – (As should our Poker-CNN)
Deep RL High Watermarks

- Atari games
  - Results keep improving
  - Although OpenAI claims equal/better results on the simpler Atari games with evolutionary algorithms
- AlphaGo – super-human achievement
- RL saves 40% on datacenter cooling – Google
Can I Apply Deep RL to My Problem?

**Pros – Go for it!**
- Clear game-like reward function
- Easy to simulate the environment
- Markov property applies [state is not path-dependent]
- Best path can be deterministic
- Rewards are observable in relatively short sequences
- Hard to compute exact problem gradients, even if solutions easy to compare
- Access to massive machine resources

**Cons – try something else.**
- No clear rewards (self driving car)
- Training data, not training environment
- Limited computational resources
- Possible to compute exact gradients on the problem (MNIST, video classification, etc)
- Not likely that random actions will ever get a positive reward (Deep QRL scored 0.0 on Montezuma’s Revenge for a long time)
Questions for Future Thought

• What are some hard problems that could be solved with Deep RL, given huge resources?
  – Example: component arrangement for microchip manufacture

• Given access to Libratus or DeepStack engine, could you design a deep net to imitate it, or to beat it?
  – With or without online simulation?

• From a small amount of expert training data, can you train a general agent for 2P games like StreetFighter?
  – Could you bootstrap it like AlphaGo?
  – Could you train it so humans can’t tell that it’s a bot?

• What problems would you train with access to a huge GPU cluster?
Thank you! Questions?
References & Further Reading

• DeepStack
  – https://www.deepstack.ai/
  – Watch weekly human vs AI matches on Twitch: https://www.twitch.tv/deepstackai

• Libratus
  – http://www.cs.cmu.edu/~sandholm/

• Poker-CNN
  – Code & models (admittedly needs cleanup) https://github.com/moscow25/deep_draw

• Annual Computer Poker Competition http://www.computerpokercompetition.org/

• Deep Reinforcement Learning
  – DeepMind: https://deepmind.com/research/dqn/
  – OpenAI: https://openai.com/

• NVidia Applied Deep Learning Research Group
  – Open requisition: https://sv.ai/nvidia-senior-research-scientist-deep-learning/