Computing Science (CMPUT) 455

Search, Knowledge, and Simulations

James Wright

Department of Computing Science
University of Alberta
james.wright@ualberta.ca

Fall 2021

455 Today - Lecture 22

- Quiz 11 review
- AlphaGo - overview and early versions
- Coursework:
  - Work on Assignment 4 (due **Tue, Dec 14**)
  - Reading: AlphaGo Zero paper
  - Quiz 12: Reinforcement Learning and AlphaGo

Quiz 11

- 73 attempts, average 93%
- Lowest: Q2: 65%; Q18: 87%

Question 2

Which of the following statements are true when we choose to use linear activation functions such as $f(x) = x$ in your NN?

Question 2

Which of the following statements are true when we choose to use linear activation functions such as $f(x) = x$ in your NN?

- Adding more neurons per layer (both width and depth) to your network no longer increases the capacity of your network.

Question 2

Which of the following statements are true when we choose to use linear activation functions such as $f(x) = x$ in your NN?

- Adding more neurons per layer (both width and depth) to your network no longer increases the capacity of your network.
  - **True**: Linear combinations of linear functions are linear. In other words, extra layers and extra weights don't do anything.

Question 2

Which of the following statements are true when we choose to use linear activation functions such as $f(x) = x$ in your NN?

- Adding more neurons per layer (both width and depth) to your network no longer increases the capacity of your network.
  - **True**: Linear combinations of linear functions are linear. In other words, extra layers and extra weights don't do anything.
- Since the derivative of $f(x) = x$ is just $f'(x) = 1$, chain rule is simplified because you can skip the derivative for the activation function.

Question 2

Which of the following statements are true when we choose to use linear activation functions such as $f(x) = x$ in your NN?

- Adding more neurons per layer (both width and depth) to your network no longer increases the capacity of your network.
    - **True**: Linear combinations of linear functions are linear. In other words, extra layers and extra weights don't do anything.
- Since the derivative of $f(x) = x$ is just $f'(x) = 1$, chain rule is simplified because you can skip the derivative for the activation function.
    - **True**: E.g., ReLU is $f(x) = x$ for $x \geq 0$, so this is actually an advantage.

Question 2

Which of the following statements are true when we choose to use linear activation functions such as $f(x) = x$ in your NN?

- Adding more neurons per layer (both width and depth) to your network no longer increases the capacity of your network.
  - **True**: Linear combinations of linear functions are linear. In other words, extra layers and extra weights don't do anything.
- Since the derivative of $f(x) = x$ is just $f'(x) = 1$, chain rule is simplified because you can skip the derivative for the activation function.
  - **True**: E.g., ReLU is $f(x) = x$ for $x \geq 0$, so this is actually an advantage.
- Your NN will no longer be able to perform classification.

Question 2

Which of the following statements are true when we choose to use linear activation functions such as $f(x) = x$ in your NN?

- Adding more neurons per layer (both width and depth) to your network no longer increases the capacity of your network.
  - **True**: Linear combinations of linear functions are linear. In other words, extra layers and extra weights don't do anything.
- Since the derivative of $f(x) = x$ is just $f'(x) = 1$, chain rule is simplified because you can skip the derivative for the activation function.
  - **True**: E.g., ReLU is $f(x) = x$ for $x \geq 0$, so this is actually an advantage.
- Your NN will no longer be able to perform classification.
  - **False**: If your problem can be linearly classified, your NN will still work.

Question 18

Compared to move evaluation with simple features, Deep Convolutional NN typically return a larger number of good moves for one input position.

Question 18

Compared to move evaluation with simple features, Deep Convolutional NN typically return a larger number of good moves for one input position.

- **False:** Deep CNNs typically return a very confident prediction of a small number of actions, whereas move evaluation with simple places moderate weight/probability on a larger number of moves. (*why?*)

AlphaGo Introduction

- High-level overview
- History of DeepMind and AlphaGo
- AlphaGo components and versions
- Performance measurements
- Games against humans
- Impact, limitations, other applications, future

About DeepMind

- Founded 2010 as a startup company
- Bought by Google in 2014
- Based in London, UK, **Edmonton (from 2017)**, Montreal, Paris
- Expertise in reinforcement learning, deep learning and search

DeepMind and AlphaGo

nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last – a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO
CONSERVATION
SONGBIRDS À LA CARTE
Illegal harvest of millions of Mediterranean birds
PAGE 452
RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 459
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card 40 years on
PAGE 462
NATUREASIA.COM

Image source:

`https://www.nature.com`

- A DeepMind team developed AlphaGo 2014-17
- Result: Massive advance in playing strength of Go programs
- Before AlphaGo: programs about 3 levels below best humans
- AlphaGo/Alpha Zero: far surpassed human skill in Go
- Now: AlphaGo is retired
- Now: Many other super-strong programs, including open source
- All are based on AlphaGo, Alpha Zero ideas

DeepMind and UAlberta

- UAlberta has deep connections
- Faculty who work part-time or on leave at DeepMind
  - Rich Sutton, Michael Bowling, Patrick Pilarski, Csaba Szepesvari (all part time)
- Many of our former students and postdocs work at DeepMind
  - David Silver - UofA PhD, designer of AlphaGo, lead of the DeepMind RL and AlphaGo teams
  - Aja Huang - UofA postdoc, main AlphaGo programmer
  - Many from the computer Poker group
- Some are starting to come back to Canada (DeepMind Edmonton and Montreal)

State of Computer Go before AlphaGo

Summary of previous work, and of our course so far

- Search - MCTS, quite strong
- Simulations - OK, hard to improve
- Knowledge
    - Good for move selection
    - Considered **hopeless for position evaluation**
- Strength: about 3 handicap levels below best humans
- Quick progress considered unlikely - see next slide

Online Betting Market - Computer Go Program Beats Human Champion

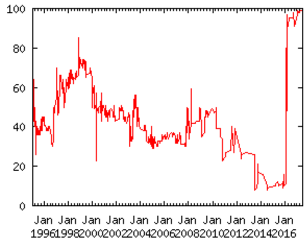


Image source:

`http://www.ideosphere.com/`

`fx-bin/Claim?claim=GoCh`

- Online "play-money" betting market
- Claim: A machine will be the best Go player in the world, sometime before the end of 2020
- Before AlphaGo, chances were considered low, and falling
- First Nature paper changed it around completely

AlphaGo Versions and Publications

- Worked on Go program for about 2 years, mostly kept secret
- DCNN paper published in 2015 (see previous lecture)
- Fall 2015
  Match **AlphaGo Fan** vs
  European champion Fan Hui (2 dan professional)
  - AlphaGo won 5:0 in official games,
    lost some other games
  - Match kept secret until January 2016
- January 2016
  Article in Nature describes this version of AlphaGo

AlphaGo Versions and Publications (continued)

- March 2016
  Match **AlphaGo Lee** vs top-level human player Lee Sedol, wins 4 - 1
- January 2017
  AlphaGo Master plays 60 games on internet vs top humans, wins 60 - 0
- May 2017
  Match vs world #1 Ke Jie, wins 3 - 0
  AlphaGo retires from match play
- October 2017
  AlphaGo Zero article in Nature
  Learns without human game knowledge
- December 2017/December 2018
  Alpha Zero article preprint/final - chess and shogi

AlphaGo Project

Image source: `http://sports.sina.com.cn/go/2016-12-19/`

- AlphaGo project was “Big Science”
- Dozens of developers
- World experts in RL, game tree search and MCTS, neural nets, deep learning
- Many millions of dollars in hardware and computing costs
- Huge engineering effort

AlphaGo vs Fan Hui

Image source:

`https://www.geekwire.com/2016/`

- Fall 2015 - early AlphaGo version **AlphaGo Fan Hui**
- Fan Hui: trained in China, 2 dan professional
- Lives in Europe since 2000, French citizen
- European champion 2013 - 2015
- AlphaGo beat Fan Hui by 5:0 in official games
- Fan Hui won some faster, informal games

AlphaGo Article in Nature

nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO
CONSERVATION
SONGBIRDS À LA CARTE
Illegal harvest of millions of Mediterranean birds
PAGE 452
RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 459
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card 40 years on
PAGE 462
NATUREASIA.COM

Image source:

`https://www.nature.com`

- Nature and Science are the top two scientific journals
- Papers on computing science are rare
- Papers on games are very rare
- Games articles from our dept:
  - Checkers Is Solved, Science 2007
  - Heads-up limit hold'em poker is solved, Science 2015
- January 2016
  AlphaGo on title page of Nature
- Describes version that beat Fan Hui

AlphaGo Fan Design

As described in January 2016 article in Nature

- Search: MCTS (pretty standard, parallel MCTS)
  - Modified UCT combines simulation result and value network evaluation
- Simulation (rollout) policy: relatively normal
  - Uses small, fast network for policy
- Supervised Learning (SL) policy network
  - DCNN, similar to their 2015 paper
  - Learns move prediction from master games
  - Improved in details, more data

AlphaGo Fan Design (continued)

- **New: strong RL policy network**
  - Learns by Reinforcement Learning (RL) from self-play
- **New: value network**
  - Learns from labeled game data created by strong RL policy
- Main contributions
  - Reinforcement learning for training DCNN policy net
  - Value network for state evaluation
  - Large, extremely well-engineered learning and playing system

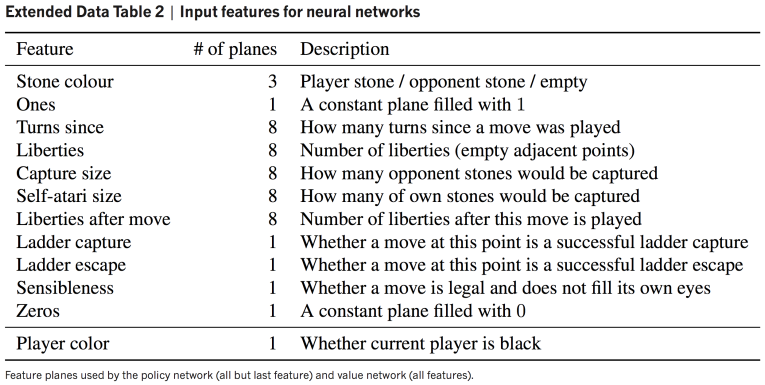
Rollout Policy Network

- Used for simulations in MCTS
- Designed to be simple and fast
- “Linear softmax of small pattern features”
- Probabilistic move selection as in Coulom’s paper, Go4
- Weights trained by RL instead of MM
- Slightly larger patterns/extended features
- 24.2% move prediction rate
- 2μs per move selection (500,000 simulated moves/second)

AlphaGo Fan Deep Network Architecture

- Three “big” networks
    - SL policy network, RL policy network, RL value network
- Same overall design for all three
- 13-layer deep convolutional NN
- 192 filters in convolutional layers
- 4.8ms evaluation on GPU
    - About 200 Evaluations/second/GPU
    - More than 2000x slower than simulation policy evaluation
    - Can learn and represent **much more complex** information

Input Features for NN



Extended Data Table 2 | Input features for neural networks

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

Feature planes used by the policy network (all but last feature) and value network (all features).

Image

source: AlphaGo paper

- Pretty similar input as previous nets

Value Network

- Learn a *state evaluation function*
- Given a Go position
- Computes probability of winning
- No search, no simulation!
- Learns mapping
  - From state s
  - To expected outcome $\mathbb{E}[z]$ of the game
- Network architecture mostly same as policy nets
- Difference: output layer
- Value net outputs single number: evaluation of s
  - Compare policy nets: output = probability distribution

AlphaGo Fan Learning Pipeline and Play

- Training Pipeline
- AlphaGo program
- Performance measurements
- Games against humans

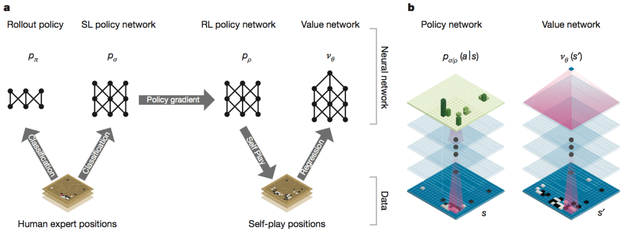
AlphaGo Fan Training Pipeline

Supervised Learning (SL)

- Rollout policy
- SL policy network

Reinforcement Learning

- RL policy network
- Value network



Supervised Learning (SL) Policy Network

- Trained from 30 million positions from the KGS Go Server
- Maximize likelihood of human move
- 57% move prediction when using all simple input features
- 55.7% when using only raw board input
- “Small improvements in accuracy led to large improvements in playing strength”
- MCTS with this network is already much better than all previous Go programs

RL Policy Network

- Learns from self play
- Learning: adjust weights of net to learn from winner of each game
- Exactly same network architecture as SL net
- Weights initialized from SL net
- Then plays many millions of games
- Keeps a pool of previous networks as opponents to avoid overfitting

Rewards and RL Training Procedure

- Goal: learn improved weights ρ for current network
- Play full game
  - Reward z only at end
  - +1 for winning, -1 for losing
- Update weights ρ by stochastic gradient ascent
- Ascent: go in the direction that maximizes expected outcome of game
- Formula from REINFORCE learning method (Williams 1992)

REINFORCE Update Formula

REINFORCE update formula

$$\Delta\rho = \alpha \frac{\partial \log p_\rho(a \mid s)}{\partial \rho} z$$

- $\Delta\rho$ .. change in weight ρ
- α step size for gradient ascent
- $p_\rho(a \mid s)$ .. probability of playing the move a which was played in the game from state s
- z .. game result (+1 or -1)

RL Policy Network vs Other Go Programs

- Tested RL Policy Network as a standalone player
- No search, only one call to network
- Plays move by sampling from p_ρ
- Won 80% against SL policy net
- Won 85% vs MCTS Go program `pachi` which used 100,000 simulations/move (!)

Training the Value Net

- Value net computes a function $v_\theta(s)$
- Function arguments:
  - input state s
  - Value net weights θ
- Output: $v_\theta(s)$ is a single real number
- Training: minimize squared error between
  - Value net prediction $v_\theta(s)$
  - True game outcome z_i

$$\text{Squared error} = \sum_i (v_\theta(s_i) - z_i)^2$$

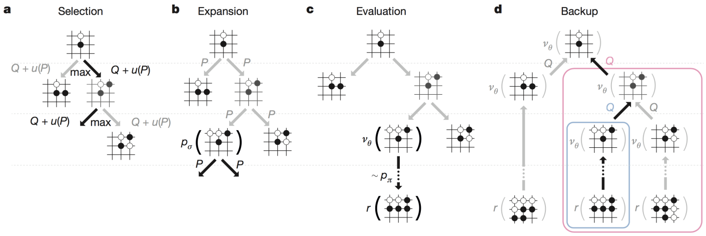
- Measure mean squared error (MSE):
  - Squared error / number of training items

Training Data for Value Net

- Played 30 million self-play games using the RL policy player
- Randomly sample *one single* state s from each game
    - Why only one state? Because of overfitting problems
- Label s with the outcome z of the game
- Result: 30 million training points (s_i, z_i)
- Learn the value net by trying to predict z_i from s_i
- This was called “reinforcement learning of value networks” in the paper
- It is really supervised learning from game data
- Game data was produced by the RL policy

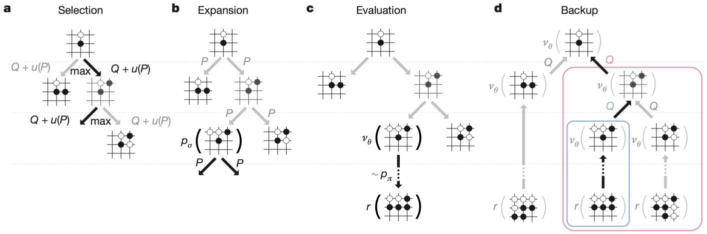
MCTS in AlphaGo Fan Player

- Player uses MCTS - search and simulations
- Neural nets used as (very strong) in-tree knowledge
  - Policy net guides tree search
  - Value net helps evaluate leaf nodes
- 3 values stored on each edge (s,a) of game tree
- Winrate Q, visit count N, prior probability P from policy net



In-tree Move Selection

- Choose move which maximizes $Q + u$
- Winrate Q, exploitation term
- u exploration term, with multiplicative knowledge
- Decay by $u = C\frac{P}{1+N}$
- Exploration constant C



Computing $V(s_L)$

- Value estimate $V(s_L)$ of leaf node s_L
- $V(s_L)$ is weighted average of two different evaluations
- $v_\theta(s_L)$ = Value network evaluation of s_L
- z_L = win/loss result of single simulation from s_L
- Combined by $V(s_L) = (1 - \lambda)v_\theta(s_L) + \lambda z_L$
  - $\lambda = 0.5$, equal weight

Which Prior Probability *P* to Use?

- Remember meaning of knowledge term in MCTS:
  - Give prior to good moves that should be searched
- Problem of RL policy in AlphaGo Fan:
  - It was optimized to play well
  - It learned a very strong prior for a **single** move
  - It was not optimized to produce a good **set** of moves to search in MCTS

SL vs RL policy network in MCTS

- Search with RL policy was too narrow
- Quality of other candidate moves not as good as in SL network
- Result: they ended up using the SL policy network, not the RL policy network in AlphaGo Fan
- SL net worked better in MCTS, even though it is much weaker in move prediction
- SL net gave a better set of good moves to search, not just a single strong move
- In AlphaGo Zero, this problem was finally solved by defining a different learning target for the policy net

Evaluation and Parallel Search

- Evaluating policy and value networks is expensive!
- Asynchronous multi-threaded search
- Simulations run on CPUs
- Policy and value networks evaluated on GPUs
- In match vs Fan Hui:
  40 search threads, 1,202 CPUs and 176 GPUs

Elo Model of Playing Strength

- Numerical rating scale
- Developed for chess
- Works well for many games of skill
- Basic idea: difference in rating corresponds to winning probability
- In Go, 100 Elo is roughly 1 stone handicap
- Formula: Difference d, win probability $1/(1 + 10^{d/400})$
- Example: 200 Elo stronger = wins about 80% of games

AlphaGo Fan Playing Strength - 2016 Nature Paper

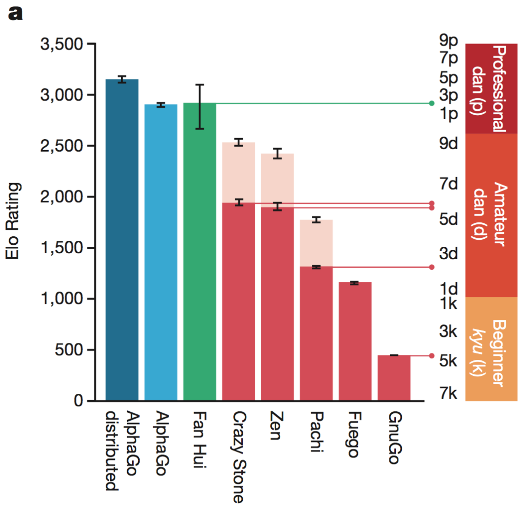


Image source: AlphaGo paper

Engineering and Technical Contributions

- Massive amounts of self-play training for the neural networks
- Massive amounts of testing/tuning
- Parallel training algorithms
- Large-scale parallel asynchronous search
- Large hardware:
- 1202 CPU, 176 GPU used vs Fan Hui

AlphaGo vs Lee Sedol

Image source:

`http://time.com/4257406/go-google-alphago-lee-sedol/`

- 5 game match in March 2016
- First ever match computer vs top player on 19 $\times$ 19 board, with no handicap
- Lee Sedol was the world's top Go player for more than a decade
- Won large number of international tournaments
- Still very strong at time of match, about world #3
- AlphaGo won the match 4:1
- Lee won game #4

AlphaGo Movie

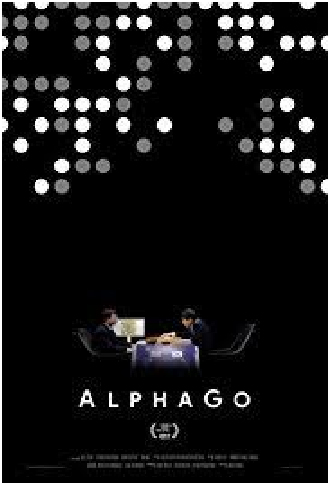


Image source: `http://www.imdb.com/title/tt6700846/`

- `https://www.alphagomovie.com`
- Story of AlphaGo development and match vs Lee Sedol
- Director Greg Kohs
- Very good movie, focus on human aspects

AlphaGo Lee

- Version used for Lee Sedol match, March 2016
- Much more RL training than for Fan Hui match
- Trained from **AlphaGo selfplay games**,
  not from RL policy games anymore
- Larger neural net
- Better hardware to evaluate neural nets
- About 2000 CPU, 48 TPU used
- 1 TPU: maybe 30x faster than 1 GPU
- Strength: 3 handicap stones stronger than AlphaGo Fan in self-play

AlphaGo Master

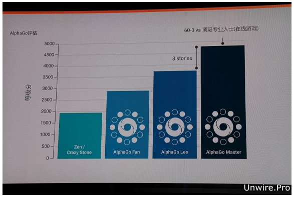


Image source:

`https://unwire.pro/2017/05/25`

- Played online end of December 2016 - early January 2017
- 60 fast games, many against top human players
- Score 60 wins no losses

AlphaGo Master Architecture

- Mostly same architecture as AlphaGo Zero (next lecture)
- Some parts were still the same as in previous AlphaGo, different from Zero:
  - Still uses handcrafted input features
  - Still uses rollouts as part of evaluation
  - Still uses initialization by SL net
- Reduced hardware:
  4 TPU, “single machine”
- Strength:
  - Far surpasses humans
  - 3 handicap stones stronger than AlphaGo Lee in direct match

AlphaGo vs Ke Jie

Image source:

- Match in May 2017 at “Future of Go Summit”
- AlphaGo played world #1 Ke Jie
- AlphaGo won 3:0
- Program: similar to AlphaGo Master
- Also on reduced hardware: 4 TPU, “single machine”

Summary

- Discussed the earlier versions of AlphaGo
- AlphaGo Fan, AlphaGo Lee, AlphaGo Master
- Main architecture: MCTS + neural networks for knowledge
- Policy net for biasing in-tree move selection
- Value net plus simulations for evaluating leaf nodes in tree
- Massively parallel implementation on CPU for simulations + GPU or TPU for nets
- Quantum leap in performance of Go-playing programs
- Reached, then surpassed level of best human Go players