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 23

- AlphaGo Zero
- USRI Evaluations open Dec 1–**Thu Dec 9**
- Coursework:
  - Work on Assignment 4 (due **Tue, Dec 14**)
  - Reading: AlphaGo Zero paper
  - Quiz 12: Reinforcement Learning and AlphaGo

USRI (Universal Student Ratings of Instruction)

- You should have received an email
- If not, use this link: `https://p20.courseval.net/etw/ets/et.asp?nxappid=UA2&nxmid=start`
- Important part of evaluating this course
- Part of instructor and TA's annual evaluation
- Please fill it out!

AlphaGo Zero

- October 2017 article in Nature
- Mastering the game of Go without human knowledge
- New simplified architecture
- Learns entirely from self-play
- No human knowledge beyond the basics such as rules of game
- Stronger than previous AlphaGo versions
- Far super-human skill

Main Technical Changes in AlphaGo Zero

- New training method tailored for improving MCTS
- Self-learning from predicting searched moves and outcomes
- New network architecture: resnets
- New network architecture: combine policy and value nets into one net with two “heads”
- Does not need large distributed system anymore, strong performance on “one machine”

Human Knowledge in Zero

- Rules of Go, legal moves
- Hard limit of $19 \times 19 \times 2 = 722$ moves on game length
- Tromp-Taylor scoring
- Input has same 2-d grid structure as Go board
- Uses rotation and reflection invariance of Go rules for training
- MCTS search parameters optimized

Human Knowledge Not Used in Zero

Some examples of knowledge used in many other Go programs, but not in Zero

- Avoid eye-filling moves (as in `Go1`)
- Patterns
- Tactics, atari, selfatari
- Human game records
- Rules for simulation policy
  No simulations outside of tree used in Zero

Zero has to learn many of these basics, and then much much more.

AlphaGo Zero's Search

- Search - still MCTS
- Used in two different ways:
  - **For learning (new)**
  - For playing
- The most impressive innovation in Zero is how search is used to improve learning, and learning in turn improves search

Main Components of AlphaGo Zero - Knowledge

Knowledge

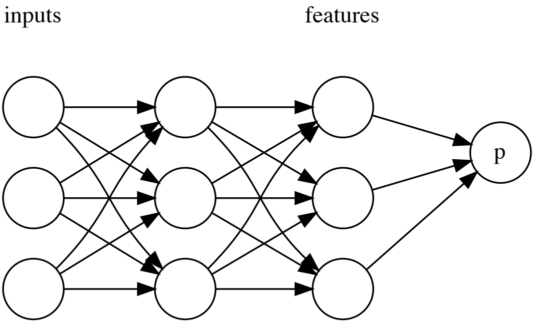
- All knowledge created by machine learning from self-play
- New network architecture
  - Knowledge represented by deep residual neural net
  - Combines policy and value nets into one net with two "heads"
  - Both move and position evaluation learned together
- No more simulations (rollouts) to end of game!
  - MCTS tree growth controlled **only** by neural net knowledge (plus real end of game states reached in the tree)

Knowledge Representation

- Deep residual neural network (He et al 2015)
- Learns two types of knowledge simultaneously
- Policy head
  - Learns good moves for the search
- Value head
  - Learns evaluation function - probability of winning
- Most of network is shared between both (**why?**)

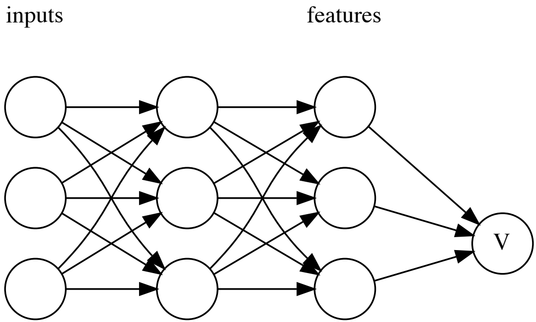
Knowledge Representation

- Deep residual neural network (He et al 2015)
- Learns two types of knowledge simultaneously
- Policy head
  - Learns good moves for the search
- Value head
  - Learns evaluation function - probability of winning
- Most of network is shared between both (**why?**)



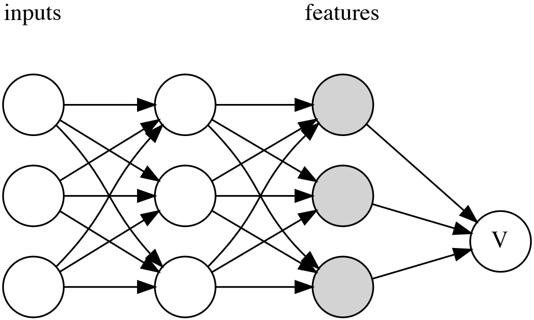
Knowledge Representation

- Deep residual neural network (He et al 2015)
- Learns two types of knowledge simultaneously
- Policy head
  - Learns good moves for the search
- Value head
  - Learns evaluation function - probability of winning
- Most of network is shared between both (**why?**)



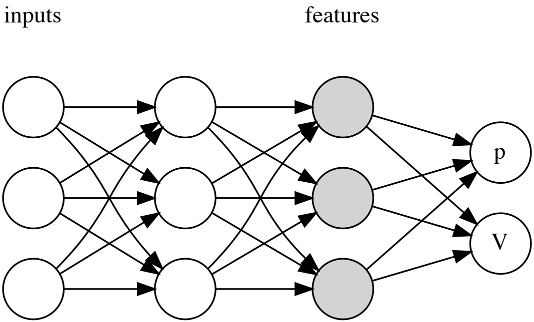
Knowledge Representation

- Deep residual neural network (He et al 2015)
- Learns two types of knowledge simultaneously
- Policy head
  - Learns good moves for the search
- Value head
  - Learns evaluation function - probability of winning
- Most of network is shared between both (**why?**)



Knowledge Representation

- Deep residual neural network (He et al 2015)
- Learns two types of knowledge simultaneously
- Policy head
  - Learns good moves for the search
- Value head
  - Learns evaluation function - probability of winning
- Most of network is shared between both (**why?**)



Deep Residual Neural Network (Resnet)

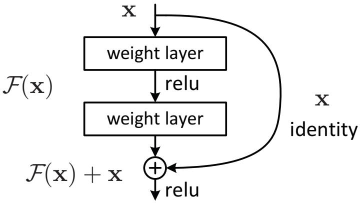


Image source: He et al 2015 article

- Main idea: pass output of previous “block” directly through
- Each block learns a “delta”, a small change to previous output
- Learning small changes is easier
- Can train really deep nets efficiently
- $>$100 layers in image recognition
- In theory, no greater representational power than DCNN
- In practice, learns better

Two-head Architecture

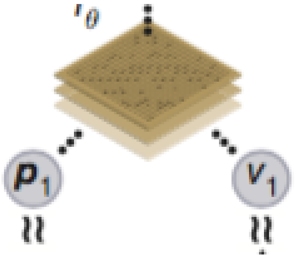
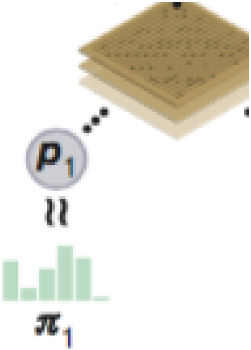


Image source: All further images from AlphaGo Zero paper, unless stated otherwise

$$(p, v) = f_\theta(s)$$

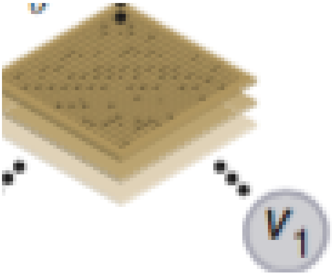
- Deep net
- Input Go position s
- Network weights θ
- Network computes function $f_\theta(s)$
- Two outputs: (p, v)
  - p vector of move probabilities $p(s, a)$ for each move a
  - v value of s

Output 1 of Neural Network: Policy Head



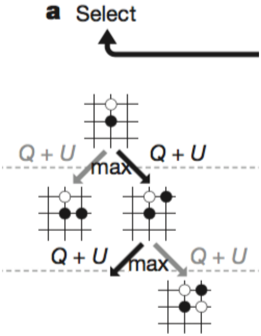
- Learns to predict what the search would do
- How frequently should each move be tried in MCTS?
- Learning goal: minimize *cross-entropy* between
  - Predicted probability of move
  - Frequency of move as selected by MCTS
- Cross-entropy: measures how well one probability distribution can predict another

Output 2 of Neural Network: Value Head



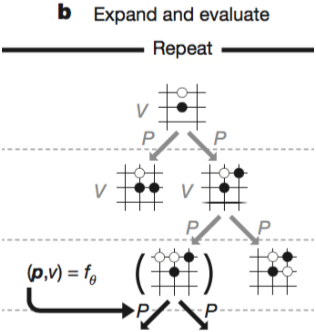
- Given a Go position
- Computes probability of winning
- Static evaluation function
- Trained from selfplay
- Learning goal: Minimize squared error between:
  - Predicted value v
  - Final result z of game

MCTS in AlphaGo Zero - Move Selection



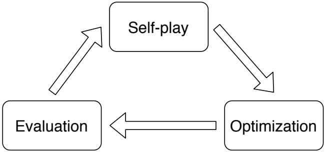
- In tree move selection
- Same formulas as in previous AlphaGo
  - Exploitation term Q
  - Exploration term u
- Meaning of Q is slightly changed
  - Value of simulation ending in in-tree state *s* = value head evaluation of *s*
  - No more simulation beyond the tree, no more evaluation component from rollouts

MCTS in AlphaGo Zero - Evaluation



- Node s expanded
- Single call to neural net
- $(p, v) = f_\theta(s)$
- p = vector of move probabilities, $p(s, a)$ for all moves a from s
- v estimated value of s

Training Pipeline



- Self-play: generate a collection of self-play game records by using MCTS + NN as both players
- Optimization: sample from game records to update the NNs
- Evaluation: play games between updated NN against previous NN. If the new NN wins 55% or more, replace NN used to generate self-play games

Network Optimization

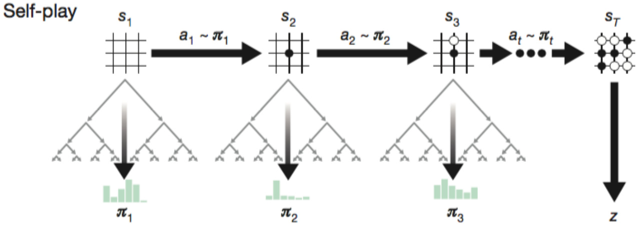
- Error measured by *loss function*
- Combines three terms
  - Error of policy head (cross entropy)
  - Error of value head
  - Regularization term to keep size of weights in check

MCTS Visit Count and Policy π

Meaning of policy π:

- Run MCTS from some state s
- If move a was played $N(s, a)$ times:
- $\pi_t(a) \propto N(s, a)^{1/\tau}$
- Probability is proportional to its "exponentiated visit count"
- Temperature parameter τ controls exploration of low-probability moves
- $\tau = 1$ for early game only, small for rest of game
- What does "proportional" mean?
- Compute values $N(s, a)^{1/\tau}$ for all actions a, then divide by their sum to make them into probabilities

Self-Play Games

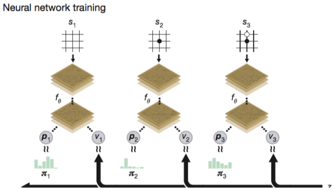


- Play whole game
- For each state s_t in game:
  - Run MCTS on s_t
  - Sample move to play according to number of simulations it received
    - Note difference to regular MCTS: *exploration*!
    - Regular MCTS would always pick the most-simulated move (exploitation)
- Finish game, get outcome z (win = +1 or loss = -1)
- Store tuples (s_t, π_t, z_t) for learning after end of game

Meaning of Tuples

- (s_t, π_t, z_t)
- s_t = state at time step t
- Game = sequence of states s_1, s_2,...
- π_t = probability distribution derived from visit count of moves in MCTS of s_t
- z_t = result from current player's point of view (z or $-z$, negamax)

Learning from Self-Play Games



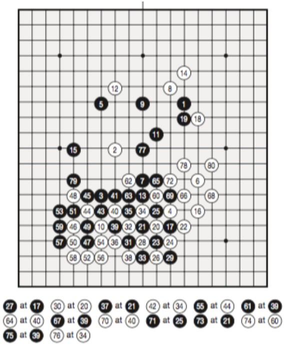
- After each game
- Randomly sample tuple (s, π, z) from all tuples stored from the game
- Adjust net weights θ by gradient descent:
- $(p, v) = f_\theta(s)$
- Make policy p better match π
- Make value v better match z

Training Process

- Most tests with 20 block resnet
- 4.9 million self-play games
- 1600 simulations / move in MCTS
- Update net in minibatches of 2048 game positions

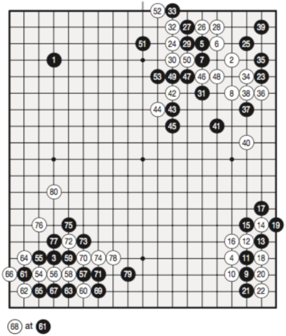
Zero After 3 Hours of Learning

c



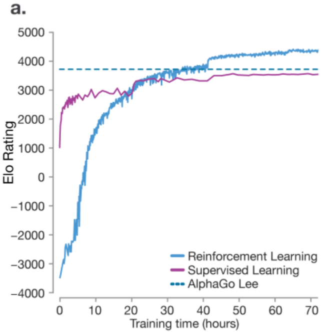
- Net learned for 3 hours
- Quick game, MCTS with 1600 simulations/move
- Learned about capturing stones
- Plays like human beginner
- Clearly better than random

Zero After 70 Hours of Learning



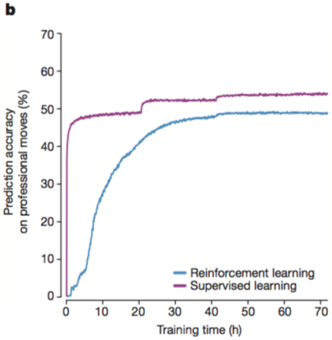
- 70 hours
- Quick game, MCTS with 1600 simulations/move
- Plays super-strong game of Go
- Complex strategies
- Exact score estimates, counting

Comparing Early Learning - RL vs SL and AlphaGo Lee



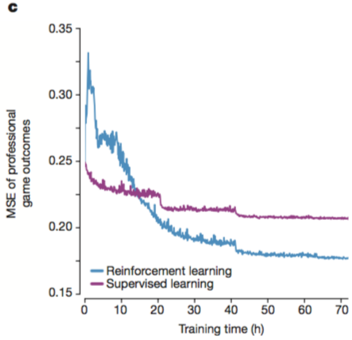
- After 72 hours of training
- Slow games vs AlphaGo Lee
- 2 hours per game per player
- Zero won 100 - 0

Move Prediction



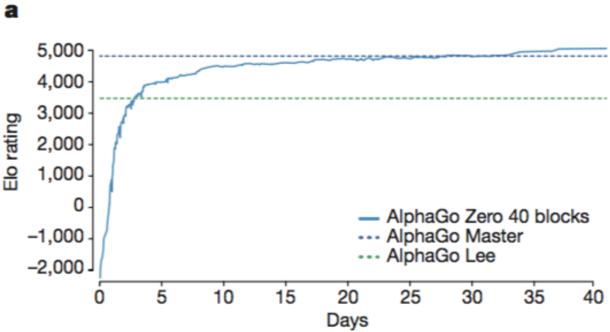
- Move prediction of human professional moves
- Learning improves, then plateaus
- Always stays below SL policy predictions (**why?**)
- Despite lower stats, most moves are very “human-like”

Predicting the Outcome of Games



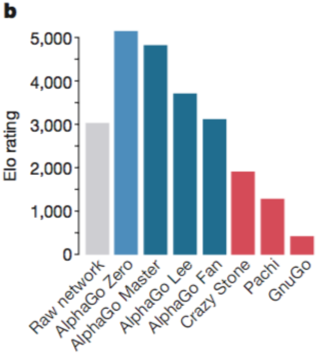
- Predict winner of human professional games
- MSE = mean square error between value net and real outcome
- Compare SL and RL value nets
- SL starts with big advantage
- RL becomes much better than SL with more training

Compare Strength with AlphaGo Lee and Master



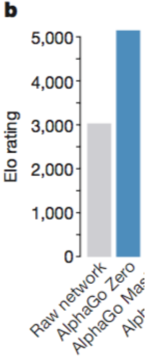
- Strongest version - 40 residual blocks instead of 20
- Trained 29 million games, 40 days
- Learning compared to Elo strength of AlphaGo Lee and AlphaGo Master
- Match Zero 40-block vs Master: 89 wins 11 losses in slow games

Results for Different AlphaGo Versions



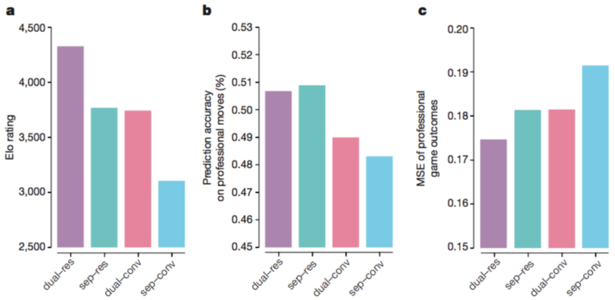
- Compares Elo for different versions of AlphaGo
- Fast games, 5 seconds / move
- Also compares Zero's "raw network" vs full Zero
- Raw network:
    - Evaluate current state *s*
    - Play highest probability move from policy head
    - No search
    - Almost as strong as AlphaGo Fan (with search)

The Importance of Search



- Raw network vs AlphaGo Zero, 5 seconds / move
- With search **2000 Elo stronger**
- That’s 20 skill levels...
- Same gap as between top human professional and weak club player
- Stronger knowledge makes search even stronger
- No “diminishing returns”, value of search remains very high

Comparing Network Architectures



- Evaluate Elo strength, move prediction, MSE of game outcomes
- sep (separate networks) vs dual (one net, 2 heads)
- conv (DCNN) vs res (residual net)
- Clear benefit of dual architecture, sharing most of network
- Clear benefit of residual net over DCNN
- Only exception: sep better in move prediction

Some Limitations of AlphaGo

- AlphaGo only plays 19×19 Go with 7.5 komi
- AlphaGo is not perfect - no proofs, “mastering” vs solving the game
- 5×5 is still the largest solved square board size, since 2002 `http://erikvanderwerf.tengen.nl/5x5/5x5solved.html`
- AlphaGo is not open source - we do not know many of the details

Limitations of an AlphaGo-like Approach to Problem-Solving

- AlphaGo relies on having a perfect model of the game
- Exact rules of game, perfect scoring of outcome, full state of game known
- Model is used for creating many millions of self-play games
- Learning relies on having these games
- Big challenge: how to learn without perfect model
- MuZero (2019) addresses this with some success

Impact of AlphaGo

- Huge impact in media, outside of core AI community
- Often described as a major step towards “machine intelligence”
- Remember main limitation - still needs an exact model to work well

Impact of AlphaGo

- Huge impact in media, outside of core AI community
- Often described as a major step towards “machine intelligence”
- Remember main limitation - still needs an exact model to work well
- Next:
    - Impact on AI and machine learning in general
    - Impact on heuristic search and computer game playing
    - Impact on human Go community

Impact on AI and Machine Learning

- Prime example of how combination of search, simulation and knowledge can achieve spectacular results
- Using deep learning:
  - Search improve knowledge
  - Knowledge improves search
  - Virtuous cycle, positive feedback loop
- Simulation has changed dramatically - in-tree only, controlled completely by neural net evaluations, no more rollouts to end of game.
- AlphaGo (Zero) has dramatically shifted the landscape of what knowledge can do as part of a larger search-based system

Impact on AI and Machine Learning

- 10 years ago, with MCTS there was a similar shift in what search can do
- After current round of progress driven by knowledge, is it time for improving search methods again?
- Main questions:
  - Which other applications of deep learning can profit from adding search and simulation?
  - Which other applications of heuristic search can profit from deep learning?

Impact on Heuristic Search and Computer Game-playing

- Dramatic shift to much stronger knowledge
- With each major advance in one of the three areas - search, knowledge, simulations -
- Need to rethink all heuristic search systems
- AlphaGo is only the beginning
- Much work ahead to fully exploit the power of stronger knowledge
- Can we learn stronger knowledge for other, harder problems
  - Video games (e.g. Atari games, Starcraft)
  - More open real-world problems with less well-defined rules

Impact on Human Professional Go Community

- Human professionals study AlphaGo and other AI games intensely
- Try out many AlphaGo-inspired openings
- Some pros are worried for their jobs
- Less interest in human tournaments?
- Can pro-level phone replace human teachers?

Impact on Human Amateur Go Community

- Temporary boost in excitement and visibility for the game of Go
- Having strong computer opponents (and online play) helps individuals in small communities without a Go club
- Will cheating become a problem, as in chess?
- Goals:
  - Turn programs into tools for teaching Go
  - Explain programs' moves in human terms

Impact on Computer Go Community

- Mission accomplished? Game over?
- Taking chess as example
  - Public interest in programming Go will fade
  - A core group of enthusiasts will keep going
  - Everyone will use programs as study tools
  - Level of humans will improve from studying with programs
- It is now possible for a single person to write a professional-level Go program in a year
- Recently, several such new programs

Beyond AlphaGo

- Computer Go Today
- Applications to other games
- Other types of applications?
- Current research: improving the techniques

Computer Go Today

- Half a dozen professional level Go programs
- Strongest: Tencent's FineArt
  - Can give top human professionals two stones handicap (!) in fast games
- Computer Go Server for automated testing
  - `http://www.yss-aya.com/cgos/`
- Computer Go Tournaments: `http://www.computer-go.info/events/index.html`

Leela Zero

Leela Zero

- Strongest open source Go program:
- Public reimplementation of AlphaGo Zero
- Smaller network for faster learning
- Reached top pro level in a few months, still improving rapidly
- Community effort, over 400 participants donate CPU and GPU cycles
- Over 13 million games played
- Improved by 14000 Elo from random play at beginning

Leela Zero Learning Curve

Recent Strength Graph (Full view.)

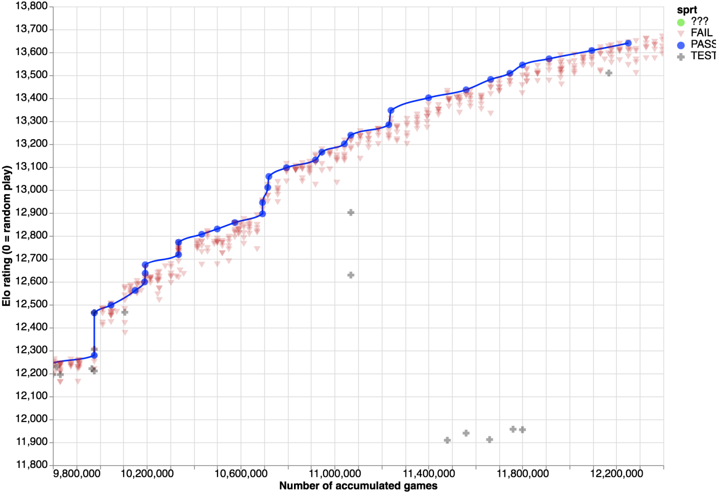


Image source: `http://zero.sjeng.org`

Summary

- Reviewed AlphaGo Zero in detail
- Discussed impact, state of computer Go after AlphaGo