Monte-Carlo Game Tree Search, Machine Learning and Deep Learning

Tsan-sheng Hsu

徐讚昇

tshsu@iis.sinica.edu.tw

http://www.iis.sinica.edu.tw/~tshsu

Abstract

- **Combining domain dependent knowledge obtained using machine learning and deep learning techniques.**
  - **Node expansion**
  - **Better simulation policy**
  - **Better position evaluation**
- **Conclusion:**
  - **Combining the power of machine learning and deep learning, Monte-Carlo approach is now above human players for computer Go.**

Domain dependent refinements

- **Main technique:**
  - **Adding domain knowledge.**
- **We use computer Go as an example here.**
- **Refinements come from machine learning and/or deep learning via training and prediction.**
  - **During the expansion phase:**
    - ▷ *Special case: open game.*
    - ▷ *General case: use domain knowledge to expand only the nodes that are meaningful with respect to the game considered, e.g., Go.*
  - **During the simulation phase: try to find a better simulation policy.**
    - ▷ *Simulation balancing for getting a better playout policy.*
    - ▷ *Other techniques are also known.*
- **Prediction of board evaluations, not just good moves.**
  - ▷ *Combined with UCB score to form a better estimation on how good or bad the current position is.*
  - ▷ *To start a simulation with a good prior knowledge.*
  - ▷ *To end a simulation earlier when something very bad or very good has already happened.*

Warnings

- **We only tell you how machine/deep learning are used in game playing by expecting the audience having some prior knowledge.**
- **We do not cover machine/deep learning theory, tool or implementation in this course.**

How domain knowledge can be obtained

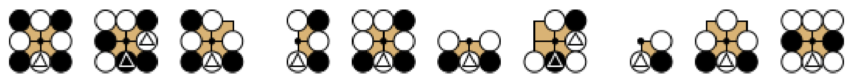
- Via human experts: very expensive to get and very difficult to be complete as proven by studies before year 2004 such as GNU Go.
- Machine learning.
  - (Local) pattern: treat positions as pictures and find important patterns and shapes within them.
    - *K by K sub-boards such as $K = 3$.*
    - *Diamond shaped patterns with different widths.*
    - *...*
  - (Global) feature: find semantics of positions.
    - *The liberties of each stone.*
    - *The number of stones can be captured by playing this intersection.*
    - *...*
  - Advanced knowledge: (higher ordered) information such as history dependent ones, namely cannot be captured using only the current position.
    - *Ko.*
    - *Features include previous several plys of a position.*
    - *A ply that has an effect to be realized a long turns latter.*

3 by 3 patterns

- **[Huang et al'10]**

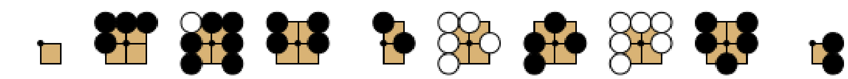
Table 4. 3×3 patterns. A triangle indicates a stone in atari. Black to move.

SB rank	1	2	3	4	5	6	7	8	9	10
MM rank	816	1029	8	1058	1055	403	441	431	960	555
SB γ	47.63	30.85	29.33	29.26	25.53	25.51	25.24	15.72	15.03	14.64
MM γ	1.55	0.95	16.98	0.88	0.89	3.34	3.10	3.15	1.10	2.50

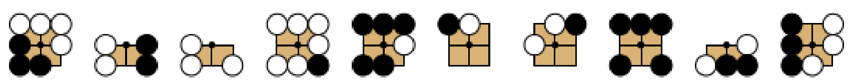


SB rank	1371	951	1870	1519	1941	148	546	3	1486	1180
MM rank	1	2	3	4	5	6	7	8	9	10
SB γ	0.92	1.01	0.43	0.85	0.24	2.35	1.13	29.33	0.86	0.98
MM γ	112.30	52.78	45.68	39.43	30.41	25.52	24.16	16.98	14.66	14.34

SB rank	2008	2007	2006	2005	2004	2003	2002	2001	2000	1999
MM rank	1982	1573	1734	2008	1762	1953	1907	1999	1971	1751
SB γ	0.02	0.02	0.03	0.03	0.04	0.04	0.04	0.04	0.05	0.06
MM γ	0.00	0.21	0.08	0.00	0.07	0.01	0.01	0.00	0.00	0.07



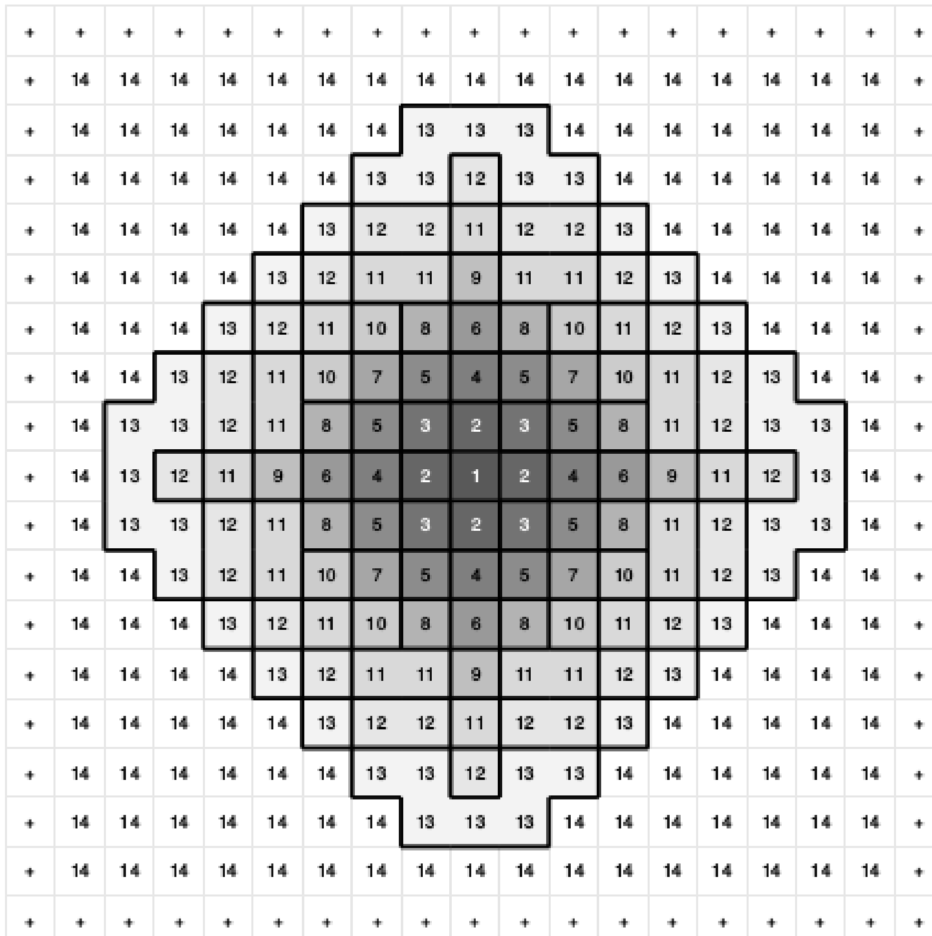
SB rank	2005	1896	1929	251	1910	1818	1874	1969	1915	2001
MM rank	2008	2007	2006	2005	2004	2003	2002	2001	2000	1999
SB γ	0.03	0.36	0.28	1.60	0.34	0.53	0.42	0.16	0.33	0.04
MM γ	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00



SB rank	11	13	14	15	16	19	25	27	28	32
MM rank	1847	1770	1775	1808	1509	420	900	1857	425	1482
SB γ	14.43	14.15	12.36	12.33	11.71	9.82	8.23	8.11	7.93	7.29
MM γ	0.03	0.07	0.06	0.04	0.28	3.25	1.27	0.03	3.21	0.29

Diamond shaped patterns

- **[Stern et al'06]**



Supervised learning

- **Use supervised learning to get a good prediction on the move to choose when a position is given: a vast amount of expert games with possible annotations are available.**
  - **Training phase.**
    - ▷ *Feed positions and their corresponding actions (moves) in expert games into the learning program.*
    - ▷ *Feature and pattern extraction from these positions.*
  - **Prediction phase.**
    - ▷ *Predict the probability of a move will be taken when a position is encountered.*
- **Many different paradigms and algorithms.**
  - **A very active research area with many applications.**

Reinforcement learning

- Use reinforcement learning to boost the baseline prediction, obtained from supervised learning for example, using self-play or expert annotated games.
  - The baseline one needs to be good enough to achieve some visible improvement.
  - Feed evaluations of positions from the baseline one into the learning program.
    - ▷ *The objective of the learning is different from the supervised learning phase.*
    - ▷ *To learn which move will result in better positions, namely positions with better evaluations.*
- Note that the predictions of moves best matched with the training data, and moves best matched with better positions may be very different.
- Many different paradigms and algorithms.
  - Another very active research area with many applications.
  - Some claimed that no baseline obtained from supervised learning is needed if the rate of convergence is fast enough [Silver et al 2017].

History

- **Using machine learning to aid computer Go programs is not new.**
  - **NeuroGo [Enzenberger'96]: neural network based move predication.**
  - **IndiGo [Bouzy and Chaslot'05]: Bayesian network.**
- **Pure learning approach is very difficult to compete with top computer Go programs with searching before AlphaGo.**
  - **Need to combine some forms of searching.**
- **Computing constraints.**
  - **It is costly, or resource consuming, to do deep learning.**
  - **In 2017, DeepMind team claimed that no supervised learning is needed even the training time is limited in training AlphaGo Zero [Silver et al 2017].**

Combining with MCTS

- **Places that MCTS needs helps.**
  - **The expansion phase: what children to explore when a leaf is to be expanded.**
  - **The simulation phase.**
    - ▷ *Originally almost random games are generated: needs a huge amount of simulated games to have a high confidence in the outcome.*
    - ▷ *Can we use more domain knowledge to get a better confidence using the same number of simulations?*
  - **Position evaluation: to end a simulation earlier or to start a simulation with better prior knowledge.**
- **Fundamental issue: assume we can only afford to use a fixed amount of resources R, say computing power in a given time constraint.**
  - **Assume each simulation takes r_s amount of resources for a strategy s in generating a playout.**
    - ▷ *Hence we can only afford to have $\lfloor \frac{R}{r_s} \rfloor$ playouts.*
  - **How to pick s to maximize c_s, the confidence or quality?**
    - ▷ *Difficult to define confidence or quality.*
  - **Not likely that r_s is linearly proportional to c_s.**

Machine learning

- **Many different framework and theories.**
  - **Decision tree.**
  - **Support vector machine.**
  - **Bayesian network.**
  - **Artificial neural network (ANN).**
  - ...
- **Here we will only introduce Bayesian network and multi-layer artificial neural network which includes convolutional neural network (CNN) and deep neural network (DNN).**
- **For each framework, depending on how the underlying optimization problem is solved, there are many different simplified models.**
  - **We will only introduce some popular models used in game playing.**
  - **There are many open-source or public domain software available.**

Bayesian network based learning (1/3)

- **Bayes theorem:** $P(B \mid A) = \frac{P(A|B)P(B)}{P(A)}$.
  - A: features and patterns
  - B: an action or a move
  - $P(A)$: probability of A happens in the training data set
  - $P(B)$: probability of an action B is taken
  - $P(A \mid B)$: probability of A appears in the training set when an action B is taken.
    - ▷ *This is the training phase.*
    - ▷ *We can compute this information from historical records.*
  - $P(B \mid A)$: when A appears, the prediction of B is taken.
    - ▷ *This is what we want.*
- **Assume there are two actions B_1 and B_2 that one can take in a position with the feature set A, then one uses the values of $P(B_1 \mid A)$ and $P(B_2 \mid A)$ to make a decision.**
  - Take one with a larger value.
  - Take one with a chance proportional to its value.
  - Take one with a chance using the idea of simulating annealing.
  - $\cdots$

Bayesian network based learning (2/3)

- **When the training set is huge and the feature set A is large, it is very time and space consuming to compute everything exactly.**
  - **In many cases, exact computation is impossible.**
    - ***Training data are usually huge in quantity, may contain error, and most of the time incomplete.***
    - ***When there are many features in a position, it is very time and space consuming to compute $P(B \mid A)$.***
- **Use some sort of approximation.**
  - **Assume a position P is consisted of features $\mathcal{P} = \{P_{A_1}, P_{A_2}, \ldots, P_{A_w}\}$.**
  - **For a possible child position B of P, give each feature P_{A_i} a strength or influence parameter $q(B, P_{A_i})$ so that it approximates the probability of $P(B \mid P_{A_i})$.**
  - **Use a function $f(q(B, P_{A_1}), \ldots, q(B, P_{A_w}))$ to approximate the value of $P(B \mid \mathcal{P})$.**

Bayesian network based learning (3/3)

- **Many different models exist to approximate the strength or influence parameter, θ, of a party, player, feature or pattern.**
  - **Bradley-Terry (BT) model.**
    - ▷ *Given 2 players with strengths θ_i and θ_j, $P(i \text{ beats } j) = \frac{e^{\theta_i}}{e^{\theta_i}+e^{\theta_j}}$.*
    - ▷ *Generalized model: Comparisons between teams of players, say odds of players $i+j$ beats both $k+m$ and $j+n+p$ is $\frac{e^{\theta_i+\theta_j}}{e^{\theta_i+\theta_j}+e^{\theta_k+\theta_m}+e^{\theta_j+\theta_n+\theta_p}}$.*
  - **Thurstone-Mosteller (TM) model.**
    - ▷ *Given 2 players with strengths that are Gaussian distributed (or normal distributed) with $\mathcal{N}(\theta_i, \sigma_i^2)$ and $\mathcal{N}(\theta_j, \sigma_j^2)$, $P(i \text{ beats } j) = \Phi(\frac{e^{\theta_i}-e^{\theta_j}}{\sqrt{\sigma_i^2+\sigma_j^2}})$, where $\mathcal{N}(\mu, \sigma^2)$ is a normal distribution with mean μ and variance σ^2, and Φ is the c.d.f. of the standard normal distribution, namely $\mathcal{N}(0, 1)$.*
    - ▷ *Generalized TM model is more involved.*
- **May not be reasonable in real life.**
  - **Does not allow cyclic relations among players.**
  - **Strength of a team needs not to be product of teammate's strength.**
- **We will use mainly BT model to illustrate the ideas here.**

BT model in computing Elo

- **This is also how Elo rating system is computed between players in games like Chess or Go.**
  - **Example: The Elo rating number of player i with strength θ_i is $400\log_{10}(\theta_i)$.**
    - *Assume the Elo ratings of players A, B and C are 2,800, 2,900 and 3,000 respectively.*
    - $P(C \textit{ beats } B) = \frac{10^{3000/400}}{10^{3000/400}+10^{2900/400}} = \frac{10^{7.5}}{10^{7.5}+10^{7.25}} \sim 0.64.$
    - $P(B \textit{ beats } A) = \frac{10^{2900/400}}{10^{2900/400}+10^{2800/400}} = \frac{10^{7.25}}{10^{7.25}+10^{7}} \sim 0.64.$
    - $P(C \textit{ beats } A) = \frac{10^{3000/400}}{10^{3000/400}+10^{2800/400}} = \frac{10^{7.5}}{10^{7.5}+10^{7}} \sim 0.76.$
    - *Note that $P(i \textit{ beats } j) + P(j \textit{ beats } i) = 1$ by assuming no draw.*
  - **Note: In Elo system, we use $10^{\theta_i/400}$ not e^{θ_i}.**
- **Fundamental problem:**
  - **When you have a huge set of data, how to compute the strength parameters using limited amount of resources?**
  - **The problem is even bigger when data may contain some errors and/or incomplete.**

Minorization-Maximization (MM)

- **Minorization-Maximization (MM): an approximation algorithm for the BT model [Coulom'07].**
  - Patterns: all possible, for example $3*3$ patterns, i.e., $3^9 = 19,683$ of them [Huang et al'11].
  - Training set: records of expert games.
- **During the simulation phase, use the prediction algorithm to form a random playout by finding average next moves.**
  - It is easy to have an efficient implementation.
  - Can add some amount of randomness in selecting the moves, such as using the idea of temperature in simulated annealing.
- **Results are very good: 37.86% correctness rate using 10,000 expert games [Wistuba et al'12]**
  - A very good playout policy may not be good enough for the purpose of finding out the average behavior.
    - ▷ *The samplings must consider the average "real" behavior of a player can make.*
    - ▷ *It is extremely unlikely that a player will make trivially bad moves.*
  - Need to balance the amount of resources used in carrying out the policy found and the total number of simulations can be computed.

Simulation balancing (SB)

- **Use the idea of self-play games to boost the performance [Huang et al'11].**
  - **Supervised plus reenforcement learning.**
  - **Feature set can be smaller.**
  - **Normally does not learn what moves are played in expert games, but how good or bad a position is.**
    - ▷ *Some forms of position evaluation.*
- **Results are extremely positive for 9 by 9 Go.**
  - **Against GNU Go 3.8 level 10.**
    - ▷ 59.3% *winning rate using MM against a good baseline of* 50%.
    - ▷ 62.6% *winning rate using SB against a good baseline program of* 50%.
- **Results are not as good on 19 by 19 Go against one using MM along.**
- **Erica, a computer Go program, later improved the SB ideas in [Huang et al'11] won 2010 Computer Olympiad 19x19 Go Gold medal.**

How they were used then

- **Assume using the BT model.**
- **Generation of the pattern database:**
  - **Manually construct.**
  - **Exhaustive enumeration: small patterns such as 3 by 3.**
  - **Find patterns happened more than a certain number of times in the training set.**
    - ▷ *Patterns, for example diamond-shapes, are too large to enumerate.*
- **Training.**
- **Setting of the parameters:**
  - **Assume after training, feature or pattern i has a strength θ_i.**
  - **Let the current position be P with b possible child positions $P_1, \ldots, P_b$.**
  - **Let F_i be the features or patterns occurred in P_i.**
  - **Let the score of P_i be $S_i = \Pi_{j \in F_i} \theta_j$.**
- **Child P_j is chosen with the probability $\frac{S_j}{\sum_{i=1}^{b} S_i}$.**
  - ▷ *The best child is one with the largest score.*

Comments

- **Implementation:**
  - Incrementally update the features and patterns found.
  - Use some variations of the Zobrist hash function to efficiently find the strength of a feature or pattern.
- **We only show two possible avenues of using Bayesian network based learning via using the BT model, namely MM and SB.**
  - There are many other choices such Bayesian full ranking.
- **The training phase needs to be done once, but takes a huge amount of space and time.**
  - Usually use some forms of iterative updating algorithms to obtain the parameters, namely the strength vector, of the model.
  - For MM with k distinct features or patterns, n training positions and an average b legal moves for each position, it takes $O(kbn)$ space, and X iterations each of which takes $O(bnkh + k^2bn)$ time, where h is the size of the pattern or feature and X is the number of iterations needed for the approximation algorithm to converge [Coulum'07].
- **The prediction phase takes only $O(kh)$ space and time.**
- **Q: Can the part of feature extraction and weights of multiple features be done automatically?**

Artificial Neural Network

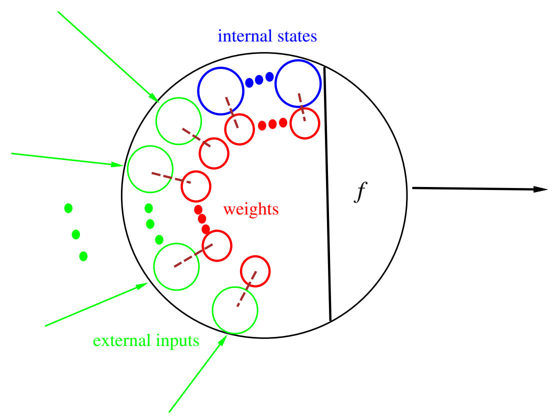
- **Using a complex network of neurons to better approximate non-linear optimizations.**
  - Usually called deep learning when the number of artificial neural network layers is more than 1.
  - Can have different architectures such as CNN or DNN.
- **A hot learning method inspired by the biological process of the animal visual cortex.**
  - Each neuron takes input from possibly overlaid neighboring sub-images of an image, and then assigns appropriate weights to each input plus some values within the cell to compute the output value.
  - This process can have multiple layers, namely a neuron's output can be other neurons' inputs, and forms a complex network.
  - Depending on the network structure, Bayesian network approaches tends to need less resources than the CNN or DNN approach.
  - There are also training phase and prediction phase.
- **Many different tools which can be parallelized using GPU.**
  - Need a great deal of resources to do training and some amount of time to do prediction.

Basics of ANN (1/3)

- **Assume the ith neuron whose output is z_i takes m_i inputs $x_{i,1}, \ldots, x_{i,m_i}$, and has internal states $y_{i,1}, \ldots, y_{i,n_i}$.**
  - **We want to assign weights $w_{i,1}, \ldots, w_{i,m_i+n_i}$ so that**

$$z_i = f((w_{i,1} * x_{i,1}), ..., (w_{i,m_i} * x_{i,m_i}), (w_{i,m_i+1} * y_{i,1}), ..., (w_{i,m_i+n_i} * y_{i,n_i})),$$

    where f is a transformation/activation function that is not hard to compute.
- **Neurons are connected as a inter-connection network where outputs of neurons can be inputs of others.**



Basics of ANN (2/3)

- Sometime for simplicity

$$z_i = f(\sum_{j=1}^{m_i}(w_{i,j} * x_{i,j}) + \sum_{j=1}^{n_i}(w_{i,j+m_i} * y_{i,j})).$$

- f is often called activation function that normalize the value.
  - Examples:
    - ▷ ***Binary step:*** $f(x) = (x \leq 0)?0 : 1$
    - ▷ ***ReLU (Rectified Linear Unit):*** $f(x) = (x < 0)?0 : x$
    - ▷ ...
  - Desired properties in obtaining optimization and consistence:
    - ▷ ***Nonlinear***
    - ▷ ***Continuously differentiable***
    - ▷ ***Monotonic***
    - ▷ ...

Basics of ANN (3/3)

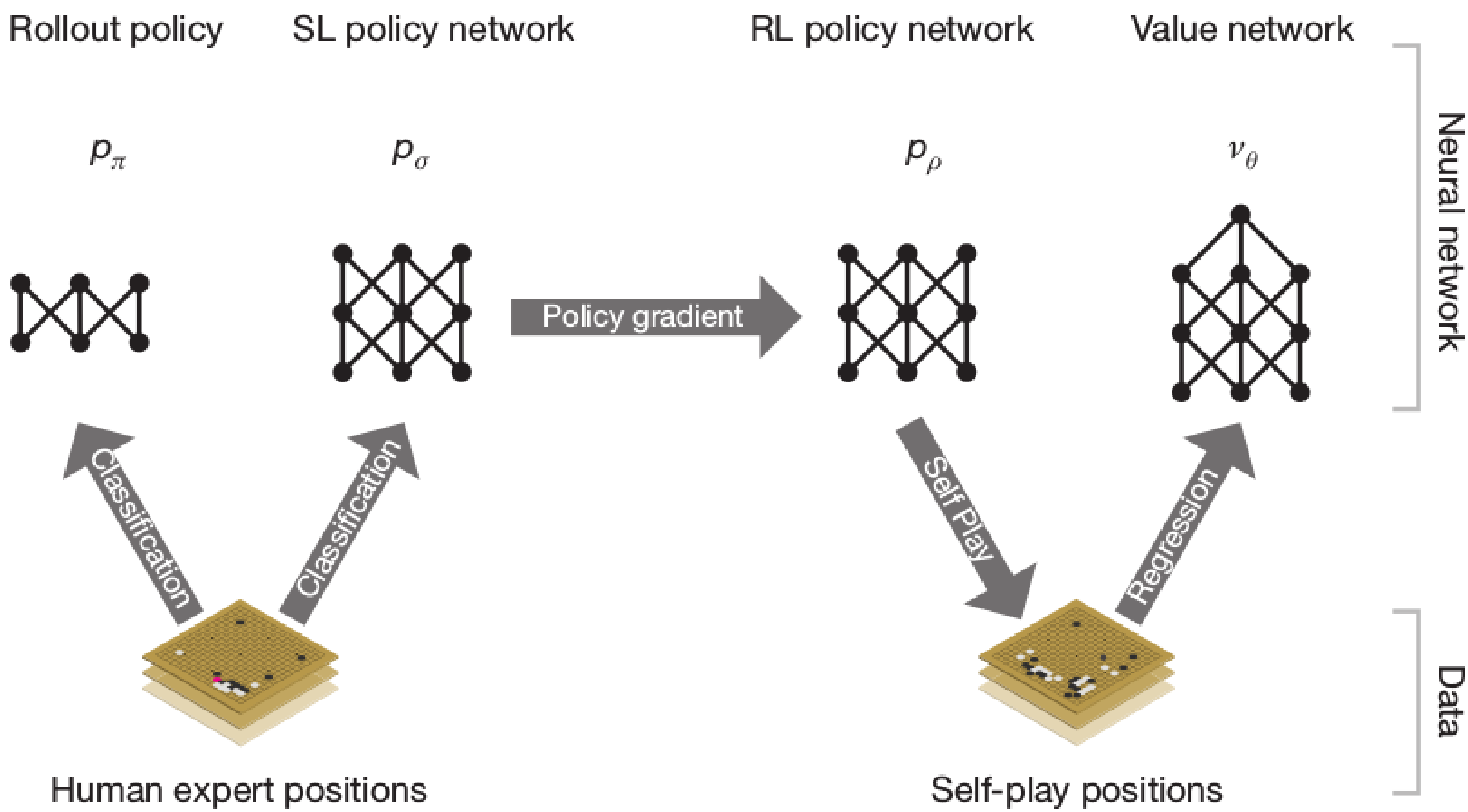
- Measurement of success
  - **Accuracy**: the percentage of your predicted values equal to their actual values.
    - *Accuracy may not be a good indicator of success since not all events, for example false positive and false negative, are equal.*
    - *Example: assume a rare event happened in a training set, then answering all negative's gives you a high accuracy, but useless prediction.*
- When there are multiple input data set, we want to find an assignment of the weights so that some **loss** or **error** function is minimized.
  - The loss or error function can be the average distance, in terms of L_1 or L_2 metric, among the training data set.
  - May want to use some log scale such as **cross entropy**.
- Many different algorithms exist to compute approximated values for the weights.
  - Computation time intensive.
  - Space usage intensive.

Deep learning

- Use artificial neural network of different sizes and structure to achieve different missions in playing 19 by 19 Go [Silver et al'16].
  - Supervised learning (SL) in building policy networks which spell out a probability distribution of possible next moves of a given position.
    - ▷ *A fast rollout policy: for the simulation phase of MCTS, prediction rate is 24.2% using only 2 μs.*
    - ▷ *A better SL rollout policy: 13-layer CNN with a prediction rate of 57.0% using 3 ms.*
  - Reinforcement learning (RL): obtain both a better, namely more accurate, policy network and at the same time a value network for position evaluation.
    - ▷ *RL policy: further training on the top of the previously obtained SL policy using more features and self-play games that achieves an 80% winning rate against the SL rollout policy.*
    - ▷ *Value network: using the RL policy to train for knowing how good or bad a position is.*

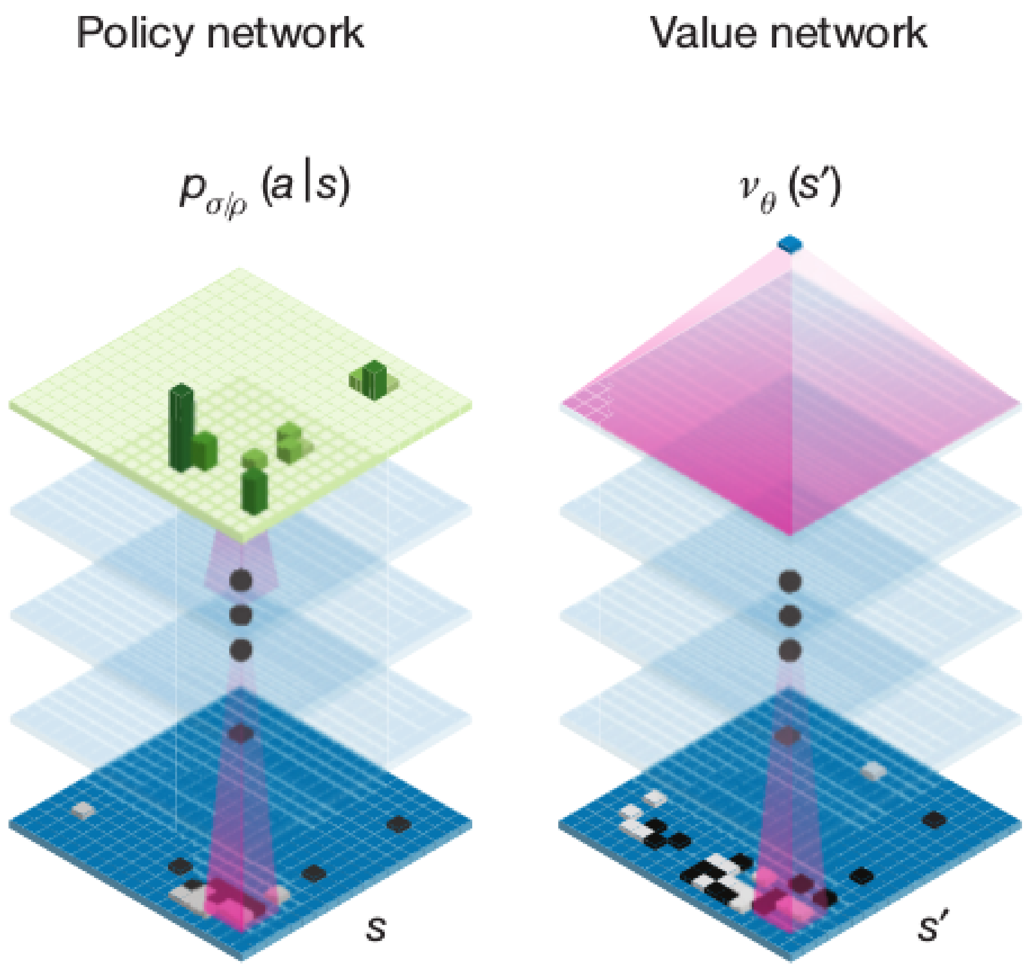
Various networks in AlphaGo

- **[Silver et al'16]**



How networks are obtained by AlphaGo

- **[Silver et al'16]**

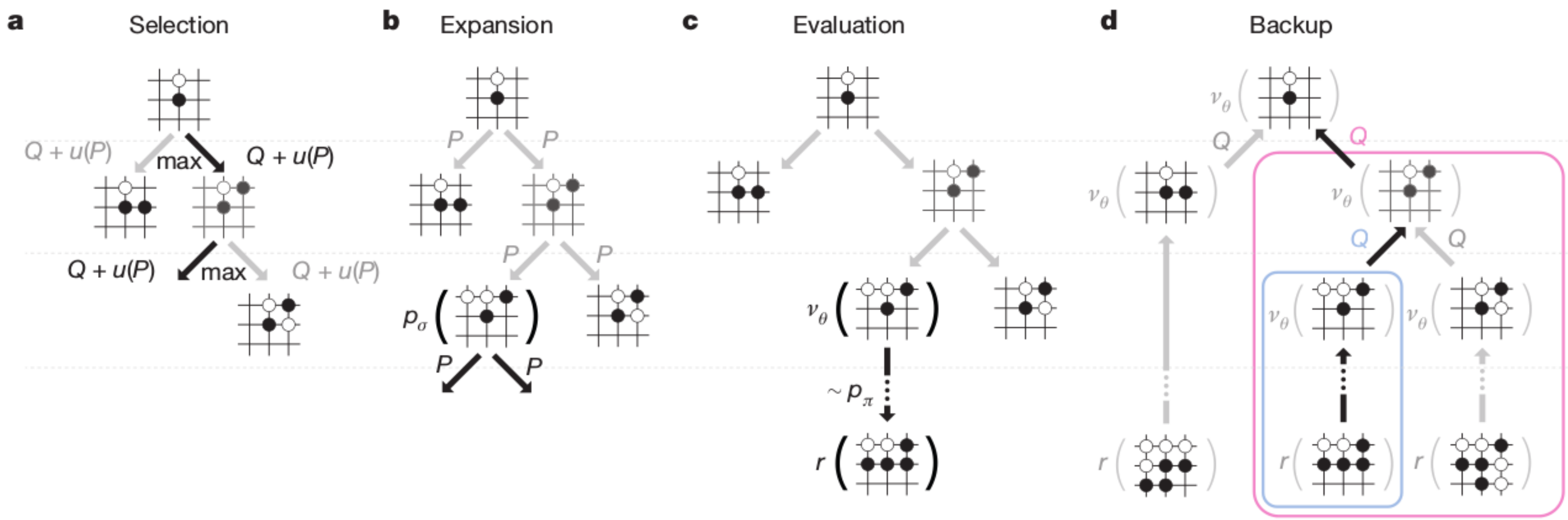


Combining networks

- Use a fast, but less accurate, SL Rollout policy to do the simulations.
  - Need to do lots of simulations.
- Use a slow, but more accurate, SL policy in the expansion phase.
  - Do not need to do node expansions too often.
- Use a slow, resource consuming and complex, but more informatic RL policy to construct the value network.
  - Do not need to do node evaluations too often.
- Using a combination of the output from the value network and the current score from the simulation phase, one can decide whether to end a simulation earlier or not.

How networks are used in AlphaGo

- **[Silver et al'16]**



Comments (1/3)

- **A very good tradeoff in performance and amount of resources used.**
  - **A less accurate but fast rollout policy is used with MCTS so that in the tree search part the correctness rate can be increased.**
    - ▷ *Need to do lots of simulations so each cannot take too much time.*
  - **Use a slow but more accurate policy for tasks such as expansion that do not need to carry out many times.**
  - **Use reinforcement learning in obtaining a value network to replace the role of designing complicated evaluation functions.**
- **Now is the way to go for computer Go!**
  - **Performance is extremely well and is generally considered to be over human champion.**
  - **Lots of legacy teams such as Zen and Crazystone are embracing ANN.**
  - **New teams such as Darkforest developed by Facebook, Fine Art developed by Tencent, and CGI developed by NCTU Taiwan, are catching up before 2016.**
- **Latest (after 2017)**
  - ▷ *Darkforest has turned open sourced in 2016.*
  - ▷ *After the introducing of AlphaGo Zero in 2017, it is difficult for others to catch up.*

Comments (2/3)

- **This approach can be used in many applications such as medical informatic which includes medical image and signal reading.**
  - **Anything that is pattern related and has lots of data collected with expert annotations.**
- **Take a lot of computing resources for computer Go.**
  - **More than 100,000 features and patterns.**
  - **More than 40 machines each with 32 cores and a total of more than 176 GPU cards whose power consumption is estimated to be in the order of 10^3 KW.**
  - **AlphaGo Zero claims to use much less resources.**
- **More studies are needed to lower the amount of resources used and to do transfer learning, namely duplicate the successful experience on one domain to another domain.**

Comments (3/3)

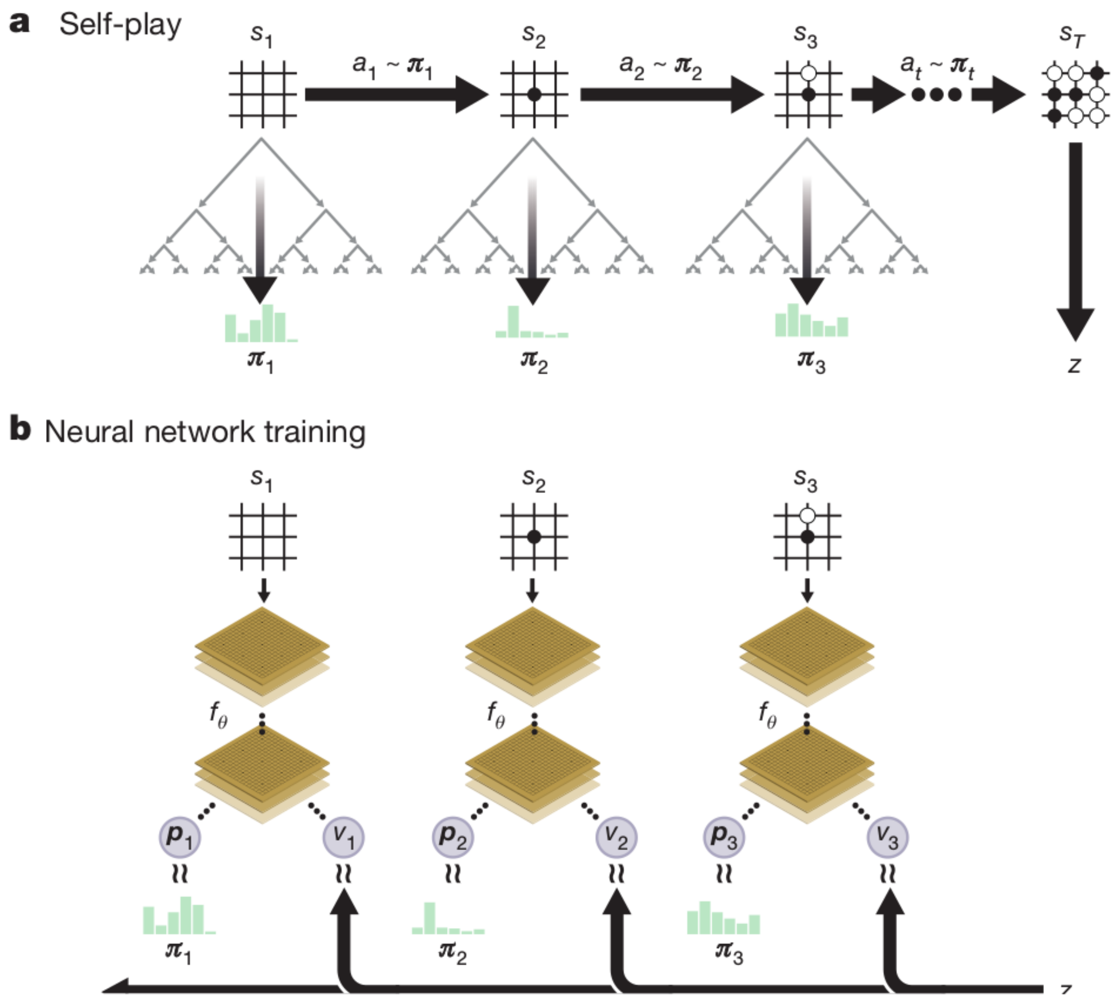
- **We only know it works by building the ANN, but it is almost impossible to explain how it works.**
  - **Very difficult to debug if a silly bug occurs.**
  - **Very difficult to "control" it to act the way you want/expect to.**
  - **It is an art to find the right coefficients and tradeoff.**
- **We also describe some fundamental techniques and ideas in the part of combining machine learning.**
  - **Other machine learning tools are also available and used.**
- **Using machine learning or MTCS along won't solve the performance problem in computer Go. However, the combination of both does the magic.**

AlphaGo Zero

- **Latest result: AlphaGo zero uses no supervised learning to achieves the top of computer Go at an Elo rating of 5185 [Silver et al. 2017].**
- **Main methods:**
  - **Trained solely by self-play reinforcement learning, starting from random play, without any supervision or use of human data.**
  - **Uses only the black and white stones from the board as input features.**
  - **Uses a single neural network, rather than separate policy and value networks.**
  - **Uses a simpler tree search that relies upon this single neural network to evaluate positions and sample moves, without performing any Monte Carlo rollouts.**
- **Contribution:**
  - **A new reinforcement learning algorithm that incorporates lookahead search inside the training loop, resulting in rapid improvement and precise and stable learning.**

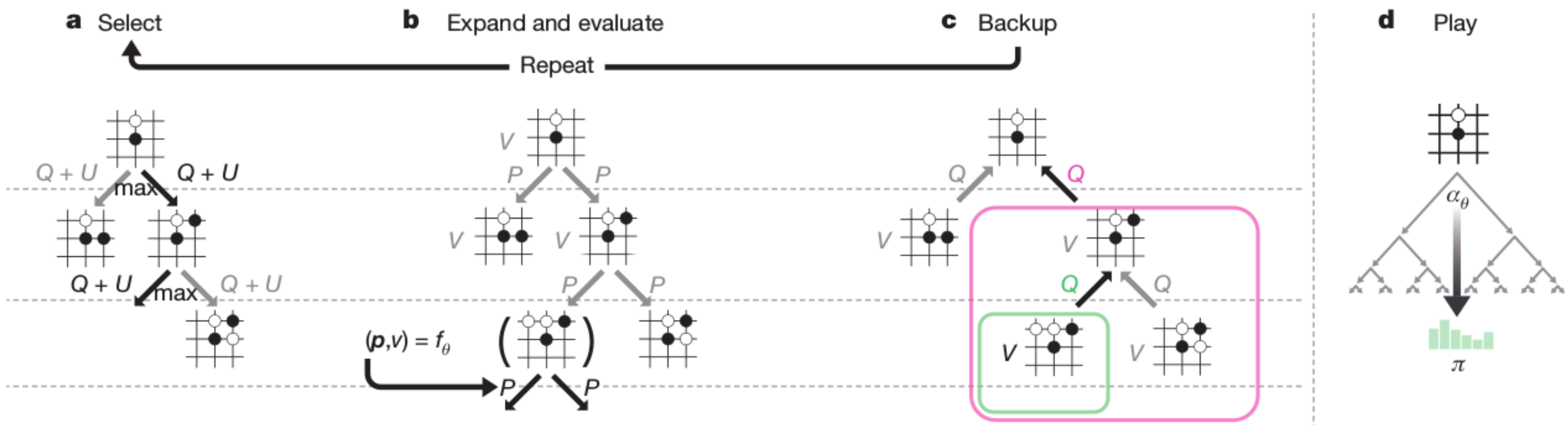
Training while self-playing

- **[Silver et al'17]**



MCTS and training together

- **[Silver et al'17]**



Comments

- Updating the network each ply you do in a self-play.
- Fast stabilizing in just 72 hours.
- Helped by special hardware and the total power consumption is greatly reduced.
  - A single machine with 4 TPU's.
- Is this a unique experience or something can be used in many other applications?

Alpha Zero (1/2)

- **A deep learning program to end all programs in game playing!**
  - **Starting from random play and given no domain knowledge except the game rules, Alpha Zero is a general algorithm that masters Chess, Go and Shogi.**
    - ▷ *No need to do supervised learning.*
    - ▷ *MCTS with deep learning beats alpha-beta with a human tuned evaluation function.*
  - **Claim to be as well for games with less-defined rules.**
    - ▷ *I tend to believe this is true!*
  - **"AlphaZero shows that it can learn that knowledge automatically – at least if you have Google's 5,000 TPUs, which is a lot of computing!" — Daylen Yang (a member of the Stockfish Chess program team) [Stetka'18].**
- **The Facebook PolyGames project tried to do a similar magic, but does not seem to be able to derive comparable performace.**

Alpha Zero (2/2)

- **Papers**
  - *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.*
    David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis.
    arXiv:1712.01815, Dec. 5, 2017.
  - *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play*
    David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis.
    Science, 07 Dec 2018: 1140-1144.
  - *"Superhuman" AI Triumphs Playing the Toughest Board Games*
    Bret Stetka
    Scientific American, December 6, 2018.

MuZero

- **Be able to master sophicated games even without building the rules of the games into the learning network.**
  - **Be able to deduce the rule set by observing games palyed.**
  - **Be able to play master level Atari, Go, chess and shogi.**
- **Citation**
  - **Mastering Atari, Go, chess and shogi by planning with a learned model Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap & David Silver *Nature*, volume 588, pages 604–609 (2020)**

Comments (1/2)

- What can be done after MuZero?
  - It is better to know "what cannot be done using MuZero?"
- Beating the best human is only a very small, yet glorious, part of why we started doing Computer game playing researches.
  - A machine beats a human in a certain skill is predictable and inevitable for any skill that can be defined by a limited set of rules.
  - Machines, with a perfect duplicating capacity, are faster and more resourceful every day, but human only gets older and fade away.
- The real purpose of playing games using computers.
  - Enabling computers more useful to human.
    - ▷ *Programming and problem solving skills that can be used in other areas.*
    - ▷ *Helping human to have a better life.*
  - Understanding fundamental structures and properties of games.
    - ▷ *What properties do a game have?*
      Fairness Fun Educational Boundary effects
    - ▷ *What rules or designs to make a game having such properties?*
    - ▷ *Why this position is more difficult to human than others?*
    - ▷ *...*

Comments (2/2)

- **Skills with limited pre-defined rules, including low-level programming, are going to fade away.**
  - **Deep learning models are built for lots of complicated previously unsolvable exactly algorithms.**
  - **More examples:**
    - ▷ *What used to need coding in assembly/machine languages to achieve desirable performance 30 years ago are now replaced by high-level programming languages and maybe sometimes programming languages running in virtual machines or by interpreters.*
    - ▷ *What used to need coding for simple accounting functions 20 years ago are now replaced by simple spreadsheet softwares like EXCEL.*
    - ▷ *What are provided in the std:: library of C++17 are hard codings for most programmers before the year 2011.*
    - ▷ *Python, PHP, C# ...*
- **What is your role, as a human being, in the age of AI emerging or "disrupting"?**
  - **Are we at least part of the revolution or evolution?**

References and further readings (1/4)

* Sylvain Gelly and David Silver. Combining online and offline knowledge in UCT. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 273–280, New York, NY, USA, 2007. ACM.
* David Silver. Reinforcement Learning and Simulation-Based Search in Computer Go. PhD thesis, University of Alberta, 2009.
* Silver, David, Huang, Aja, Maddison, Chris J, Guez, Arthur, Sifre, Laurent, Van Den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, et al. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587):484-489, 2016.
* Silver, David, Schrittwieser, Julian, Simonyan, Karen, Antonoglou, Ioannis, Huang, Aja, Guez, Arthur, Hubert, Thomas, Baker, Lucas, Lai, Matthew, Bolton, Adrian, et al. (2017). Mastering the game of Go without human knowledge. Nature, 550(7676):354359, 2017

References and further readings (2/4)

- Hugues Juille. *Methods for Statistical Inference: Extending the Evolutionary Computation Paradigm*. PhD thesis, Department of Computer Science, Brandeis University, May 1999.
- Maddison, C. J., Huang, A., Sutskever, I., & Silver, D. (2014). Move evaluation in go using deep convolutional neural networks. arXiv preprint arXiv:1412.6564.
- Tian, Y., & Zhu, Y. (2015). Better Computer Go Player with Neural Network and Long-term Prediction. arXiv preprint arXiv:1511.06410.

References and further readings (3/4)

- Shih-Chieh Huang, Rmi Coulom, and Shun-Shii Lin. Monte-Carlo Simulation Balancing in Practice. In H. Jaap van den Herik, H. Iida, and A. Plaat, editors, *Lecture Notes in Computer Science 6515: Proceedings of the 7th International Conference on Computers and Games*, pages 81–92. Springer-Verlag, New York, NY, 2011.
- Stern, D., Herbrich, R., and Graepel, T. (2006, June). Bayesian pattern ranking for move prediction in the game of Go. In Proceedings of the 23rd international conference on Machine learning (pp. 873-880). ACM.
- Wistuba, M., Schaefers, L., and Platzner, M. (2012, September). Comparison of Bayesian move prediction systems for Computer Go. In Computational Intelligence and Games (CIG), 2012 IEEE Conference on (pp. 91-99). IEEE.

References and further readings (4/4)

- Coulom, R. (2007). Computing Elo ratings of move patterns in the game of go. In Computer games workshop.
- B. Bouzy and G. Chaslot, "Bayesian generation and integration of K-nearest-neighbor patterns for 19x19 Go", IEEE 2005 Symposium on Computational Intelligence in Games, Colchester, UK, G. Kendall & Simon Lucas (eds), pages 176-181.
- Enzenberger, M. (1996). The integration of a priori knowledge into a Go playing neural network. URL: http://www. markus-enzenberger.de/neurogo.html.
- Clark, C., & Storkey, A. (2014). Teaching deep convolutional neural networks to play go. arXiv preprint arXiv:1412.3409.