AlphaGo or how DeepMind taught machine to win

Gadetsky Artyom

CS NRU HSE

30 September 2016 y.

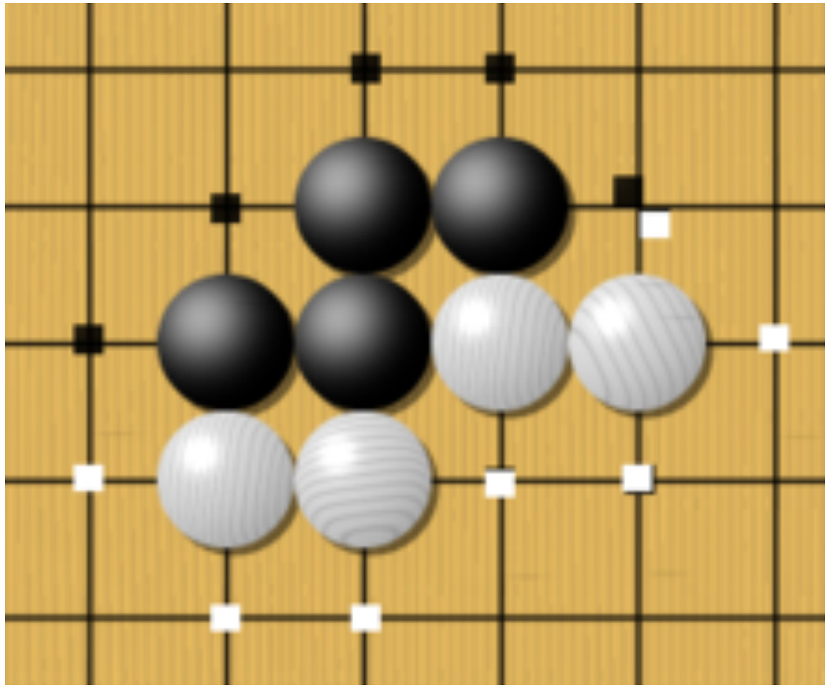
Go equipment

- Game board – goban
- Stones of two different colors, e.g. black and whites
- Bowls for storing stones
- Game clock at competitions

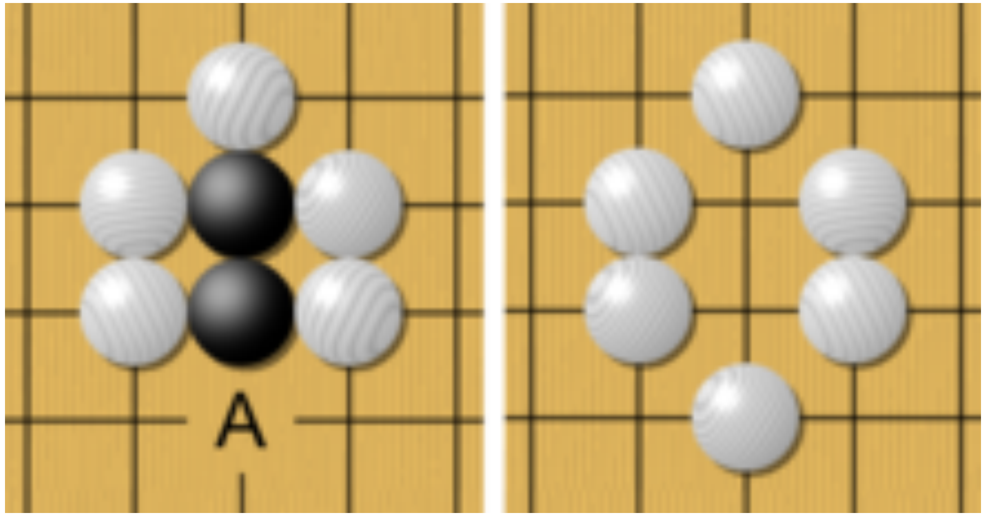
Rules of Go: basics

- Black goes first
- Stones are placed on vacant points of the grid
- Vertically and horizontally adjacent stones of the same color form a chain
- A vacant point adjacent to a stone is called a *liberty* for that stone
- Komi - awarding White some compensation for second move

Rules of Go: example of chains and liberties

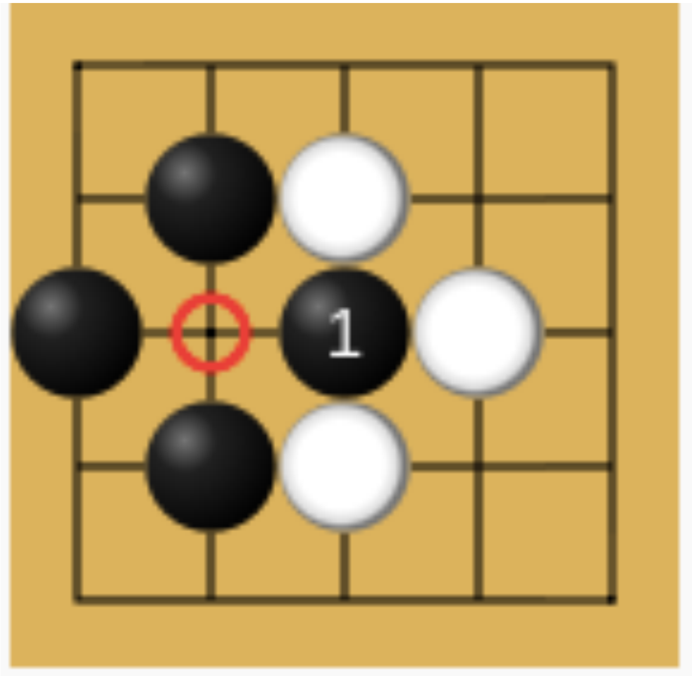


One black chain and two white chains, with their liberties marked with dots.



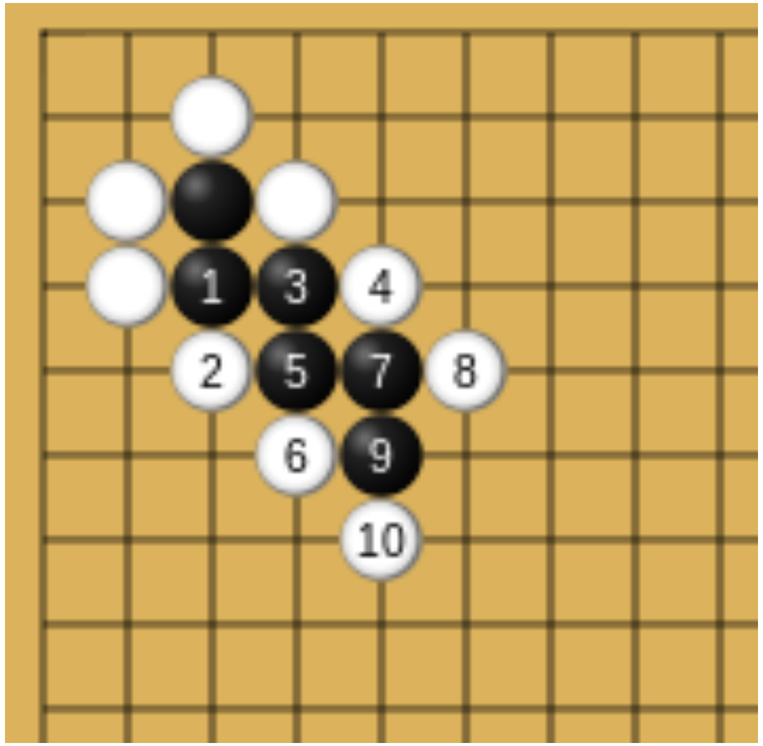
Example of capturing stones

Rules of Go: ko rule



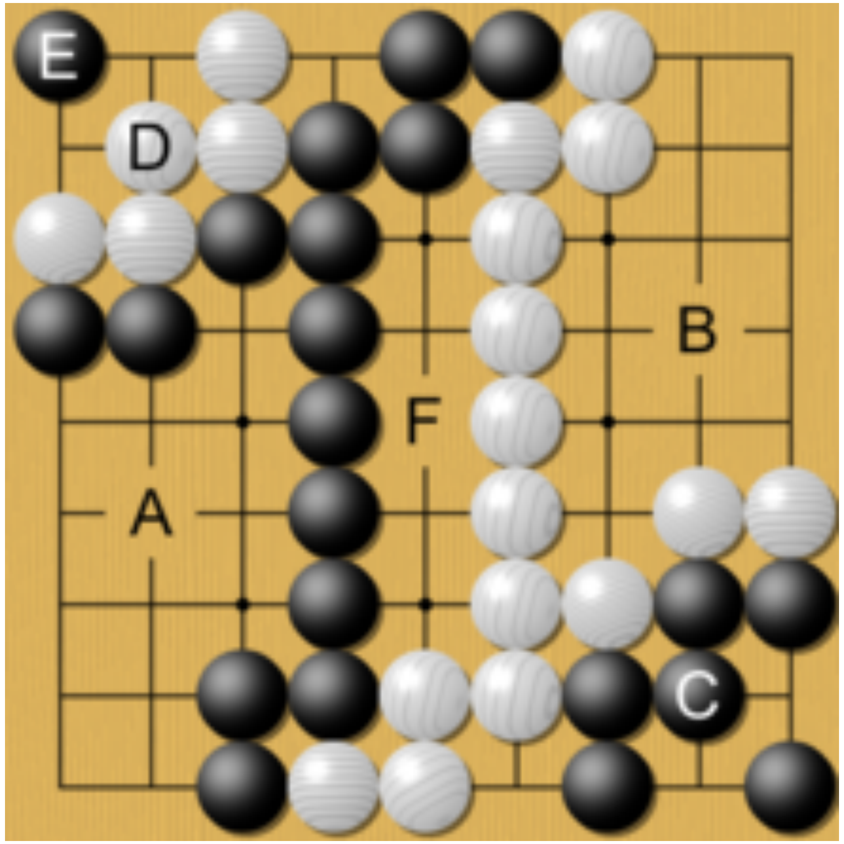
An example of a situation in which the ko rule applies

Basic capturing tactic: ladder



A ladder. Black cannot escape unless the ladder connects to black stones further down the board that will intercept with the ladder.

Rules of Go: scoring



Possible end
of the game

Obstacles to computer performance

- Size of the board (19x19, 361 intersections)
- Number of legal moves
- Much harder in comparison with chess
- Position or move evaluation function

Approaches to solving problems

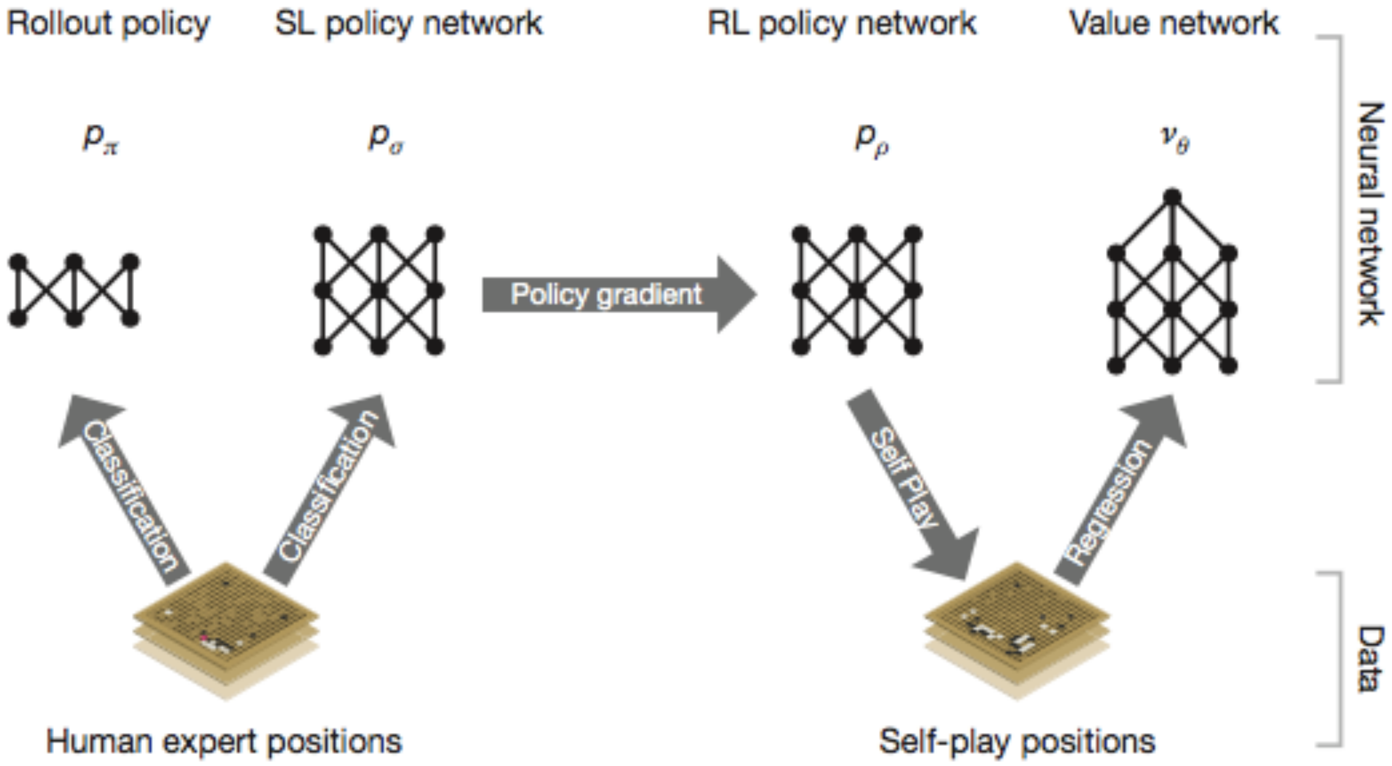
- Go – game of perfect information

- All games of perfect information have optimal value function

- Two general principles to reduce effective search space

  - Reduce depth of the search with position evaluation
  - Reduce breadth of the search with sampling from probability distribution over possible moves from fixed position

AlphaGo learning pipeline



D. Silver et al.
Mastering the game of Go with deep neural networks and tree search

SL policy network: data preparation

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).

D. Silver et al.
Mastering the game of Go with deep neural networks and tree search

SL policy network: architecture of p_σ

- Input Layer: 19x19x48 image stack

- 1 hidden layer:
  - zero padding to 23x23
  - convolving 192 filters of size 5x5 with stride equal to 1
  - applying rectifier unit

- 2-12 hidden layer:
  - zero padding to 21x21
  - convolving 192 filters of size 3x3 with stride equal to 1
  - applying rectifier unit

- Output layer:
  - convolving 1 filter of size 1x1 with stride equal to 1
  - applying softmax to get probability distribution over all possible moves

SL policy network: training

- Each training step:
  - Randomly get m samples from prepared data set
  - Maximize log likelihood of the action by stochastic gradient descent:

$$\Delta\sigma = \frac{\alpha}{m}\sum_{k=1}^{m}\frac{\partial \log p_{\sigma}(a^{k}|s^{k})}{\partial\sigma}$$

Rollout policy: fast linear softmax p_π

Feature	# of patterns	Description
Response	1	Whether move matches one or more response pattern features
Save atari	1	Move saves stone(s) from capture
Neighbour	8	Move is 8-connected to previous move
Nakade	8192	Move matches a *nakade* pattern at captured stone
Response pattern	32207	Move matches 12-point diamond pattern near previous move
Non-response pattern	69338	Move matches 3×3 pattern around move
Self-atari	1	Move allows stones to be captured
Last move distance	34	Manhattan distance to previous two moves
Non-response pattern	32207	Move matches 12-point diamond pattern centred around move

Features used by the rollout policy (first set) and tree policy (first and second set). Patterns are based on stone colour (black/white/empty) and liberties (1, 2, $\geq$3) at each intersection of the pattern.

D. Silver et al.
Mastering the game of Go with deep neural networks and tree search

RL policy network: training p_ρ by self-play

- Each iteration of training consists of **n** games

- p_ρ - network being trained
- $p_{\rho'}$ - network from randomly chosen previous iteration

- Weights ρ and ρ' were initialized to σ from SL policy network

- Every 500 iterations the current parameters ρ were added to the opponent pool.

RL policy network: training p_ρ by self-play

- Play out each game i until termination step T^i
- Score game to get the outcome z_t^i = $\pm r(s_{T^i})$
- Replay games for policy gradient update using SGD
  - $\Delta p = \frac{\alpha}{n}\sum_{i=1}^{n}\sum_{t=1}^{T^i}\frac{\partial log p_\rho(a_t^i \mid s_t^i)}{\partial \rho}(z_t^i - v(s_t^i))$
- 10.000 iterations of 128 games took one day

Value network: overfitting on KGS data

- Data consisting of complete games leads to overfitting due to:
  - difference between successive positions is just one stone
  - but regression target is shared for the whole game
- As the result we get:
  - 0.19 MSE on the training set
  - 0.37 MSE on the test set
  - Memorized outcomes

Value network: let's generate own data

- Each game was generated by successive sampling:

  - Time step $U \sim unif\{1, 450\}$
  - First $t = 1 \ldots U - 1$ moves: $a_t \sim p_\sigma(\cdot\,|s_t)$
  - $a_U \sim \mathrm{Unif}\{1, 361\}$ repeatedly until a_U is legal
  - Remaining $t = U + 1\ \ldots T$ moves: $a_t \sim p_\rho(\cdot\,|s_t)$

- Score game to get the outcome $z_t = \pm r(s_t)$
- Add only $(s_{U+1}\ ;\ z_{U+1})$ to new data set from each game

Value network: architecture of v_θ

- Input to the CNN:
  - the same as in SL policy network
  - plus additional player colour binary feature
- Hidden layers:
  - also the same
  - plus one more convolutional layer is added.
- Output Layer:
  - fully connected layer with 256 units
  - applying single tanh unit

Value network: training v_θ

- Also the same as in SL policy network, but maximizing MSE

$$\Delta\theta = \frac{\alpha}{m}\sum_{k=1}^{m}(z^k - v_\theta(s^k))\frac{\partial v_\theta(s^k)}{\partial\theta}$$

- 50 million mini batches of 32 positions.
- Training took one week

Search algorithm

- Each node s contains edges (s, a) for all legal actions

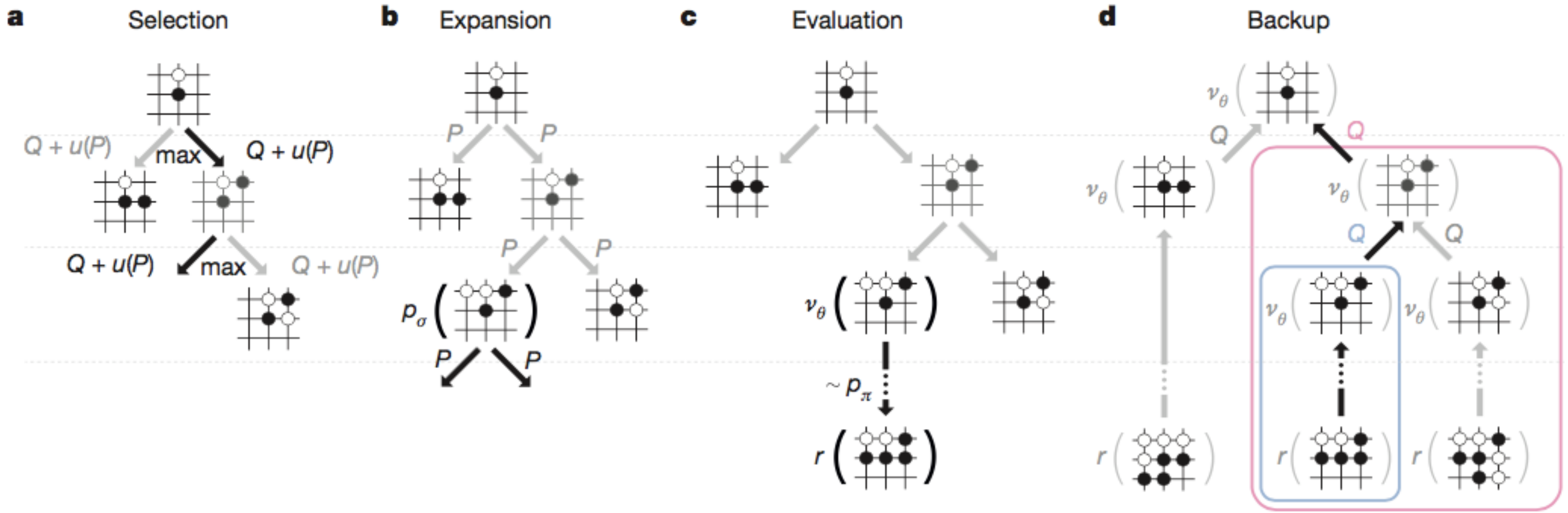
- Each edge stores a set of statistics:

  - P(s, a) – prior probability

  - $W_v(s, a)$ and $W_r(s, a)$ – Monte Carlo estimates of total action value, accumulated over leaf evaluations $N_v(s, a)$ and rollout rewards $N_r(s, a)$

  - Q(s, a) - combined mean action value for that edge.

Search algorithm



D. Silver et al.
Mastering the game of Go with deep neural networks and tree search