# CS 747, Autumn 2023: Lecture 24

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Autumn 2023

# Application of RL to Go

- AlphaGo
- Summary and outlook

# Application of RL to Go

- AlphaGo
- Summary and outlook

# Main References

- Mastering the game of Go with deep neural networks and tree search.

  David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis, Nature: 529:484–489, 2016.

- Mastering the game of Go without human knowledge.

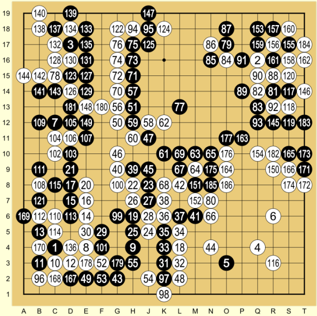
  David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis, Nature, 550:354–358, 2017.

- 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, and Demis Hassabis, Science: 362(6419):1140–1144, 2018.

# 2016: AI Conquers Go

- In 2016, Google DeepMind's AlphaGo program (Silver *et al.*, 2016) defeats Lee Sedol (international champion), 4–1.



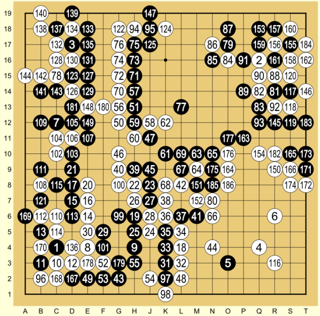
Lee Sedol (B) vs AlphaGo (W) - Game 1

[1]

[1] https://upload.wikimedia.org/wikipedia/commons/thumb/5/56/Lee_Sedol_%28B%29_vs_AlphaGo_%28W%29_-_Game_1.svg/734px-Lee_Sedol_%28B%29_vs_AlphaGo_%28W%29_-_Game_1.svg.png. 

# 2016: AI Conquers Go

- In 2016, Google DeepMind's AlphaGo program (Silver *et al.*, 2016) defeats Lee Sedol (international champion), 4–1.



Lee Sedol (B) vs AlphaGo (W) - Game 1

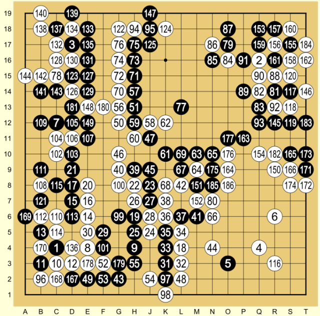
[1]

- 19 $\times$ 19 board; turn-based; black and white stones.
- Surround opponent's stones to capture them.

[1] `https://upload.wikimedia.org/wikipedia/commons/thumb/5/56/Lee_Sedol_%28B%29_vs_AlphaGo_%28W%29_-_Game_1.svg/734px-Lee_Sedol_%28B%29_vs_AlphaGo_%28W%29_-_Game_1.svg.png`. CC image courtesy of Wesalius on WikiMedia Commons licensed under CC-BY-SA-4.0.

# 2016: AI Conquers Go

- In 2016, Google DeepMind's AlphaGo program (Silver *et al.*, 2016) defeats Lee Sedol (international champion), 4–1.



Lee Sedol (B) vs AlphaGo (W) - Game 1

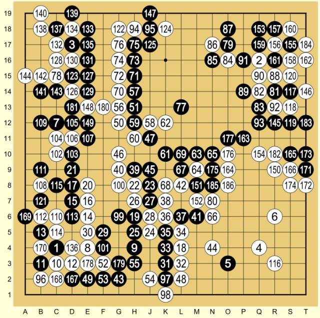
[1]

- $19 \times 19$ board; turn-based; black and white stones.
- Surround opponent's stones to capture them.
- AlphaGo **ingredients**:
  - Value network $v_\theta$.
  - Policy networks $p_\sigma$, $p_\rho$.
  - Rollout policy network $p_\pi$.

[1] `https://upload.wikimedia.org/wikipedia/commons/thumb/5/56/Lee_Sedol_%28B%29_vs_AlphaGo_%28W%29_-_Game_1.svg/734px-Lee_Sedol_%28B%29_vs_AlphaGo_%28W%29_-_Game_1.svg.png`. 

# 2016: AI Conquers Go

- In 2016, Google DeepMind's AlphaGo program (Silver *et al.*, 2016) defeats Lee Sedol (international champion), 4–1.



Lee Sedol (B) vs AlphaGo (W) - Game 1

[1]

- $19 \times 19$ board; turn-based; black and white stones.
- Surround opponent's stones to capture them.
- AlphaGo **ingredients**:
  - Value network $v_\theta$.
  - Policy networks $p_\sigma$, $p_\rho$.
  - Rollout policy network $p_\pi$.

  How learned, how used?

[1] https://upload.wikimedia.org/wikipedia/commons/thumb/5/56/Lee_Sedol_%28B%29_vs_AlphaGo_%28W%29_-_Game_1.svg/734px-Lee_Sedol_%28B%29_vs_AlphaGo_%28W%29_-_Game_1.svg.png. CC image courtesy of Wesalius on WikiMedia Commons licensed under CC-BY-SA-4.0.

# 1. Supervised Learning of Policy Networks

- $p_\sigma$ obtained by supervised learning on data (30 million moves) from expert games in KGS Go Server database.
  - 13-layer CNN; 48 hand-designed input features per position, softmax output (over legal actions).
  - Trained using gradient ascent to maximise log-likelihood; accuracy $= 57\%$.

# 1. Supervised Learning of Policy Networks

- $p_\sigma$ obtained by supervised learning on data (30 million moves) from expert games in KGS Go Server database.
  - 13-layer CNN; 48 hand-designed input features per position, softmax output (over legal actions).
  - Trained using gradient ascent to maximise log-likelihood; accuracy $= 57\%$.

- $p_\pi$ trained similarly, to be used for rollouts.
  - Linear + softmax; accuracy $= 24\%$; much faster to compute ($2\mu s$ for forward pass, compared to $3ms$ for $p_\sigma$).

# 2. Self-play, Reinforcement Learning

- $p_\rho(0) = p_\sigma$. Good initial seed.

# 2. Self-play, Reinforcement Learning

- $p_\rho(0) = p_\sigma$. Good initial seed.
  - $p_\rho(0)$ plays many games against $p_\rho(0)$, learns from them to become $p_\rho(1)$.

# 2. Self-play, Reinforcement Learning

- $p_\rho(0) = p_\sigma$. Good initial seed.
  - $p_\rho(0)$ plays many games against $p_\rho(0)$, learns from them to become $p_\rho(1)$.
  - For $i \geq 1$, $p_\rho(i)$ plays many games against $\{p_\rho(0), p_\rho(1), \ldots, p_\rho(i)\}$, learns from them to become $p_\rho(i + 1)$.

    Ensures stable progress in sequence of agents.

# 2. Self-play, Reinforcement Learning

- $p_\rho(0) = p_\sigma$. Good initial seed.
  - $p_\rho(0)$ plays many games against $p_\rho(0)$, learns from them to become $p_\rho(1)$.
  - For $i \geq 1$, $p_\rho(i)$ plays many games against $\{p_\rho(0), p_\rho(1), \ldots, p_\rho(i)\}$, learns from them to become $p_\rho(i+1)$.

    Ensures stable progress in sequence of agents.
  - "Many games" $=$ few thousands; learning steps $=$ few tens.

# 2. Self-play, Reinforcement Learning

- $p_\rho(0) = p_\sigma$. Good initial seed.
  - $p_\rho(0)$ plays many games against $p_\rho(0)$, learns from them to become $p_\rho(1)$.
  - For $i \geq 1$, $p_\rho(i)$ plays many games against $\{p_\rho(0), p_\rho(1), \ldots, p_\rho(i)\}$, learns from them to become $p_\rho(i+1)$.

    Ensures stable progress in sequence of agents.
  - "Many games" $=$ few thousands; learning steps $=$ few tens.

- Learning using REINFORCE with $v_\theta$ subtracted as baseline.

# 2. Self-play, Reinforcement Learning

- $p_\rho(0) = p_\sigma$. Good initial seed.
  - $p_\rho(0)$ plays many games against $p_\rho(0)$, learns from them to become $p_\rho(1)$.
  - For $i \geq 1$, $p_\rho(i)$ plays many games against $\{p_\rho(0), p_\rho(1), \ldots, p_\rho(i)\}$, learns from them to become $p_\rho(i+1)$.

    Ensures stable progress in sequence of agents.
  - "Many games" $=$ few thousands; learning steps $=$ few tens.

- Learning using REINFORCE with $v_\theta$ subtracted as baseline.
  - Reward: $+1$ for win, $-1$ for loss, $0$ for all other states.

# 2. Self-play, Reinforcement Learning

- $p_\rho(0) = p_\sigma$. Good initial seed.
  - $p_\rho(0)$ plays many games against $p_\rho(0)$, learns from them to become $p_\rho(1)$.
  - For $i \geq 1$, $p_\rho(i)$ plays many games against $\{p_\rho(0), p_\rho(1), \ldots, p_\rho(i)\}$, learns from them to become $p_\rho(i+1)$.

    Ensures stable progress in sequence of agents.
  - "Many games" $=$ few thousands; learning steps $=$ few tens.

- Learning using REINFORCE with $v_\theta$ subtracted as baseline.
  - Reward: $+1$ for win, $-1$ for loss, $0$ for all other states.
  - $p_\rho$ (without search) has win record
    - 80% against $p_\sigma$ (without search),
    - 85% against Pachi (independent agent using MCTS).

# 3. Policy Evaluation

- Evaluation function in search should *ideally* be $V^\star$. Instead use $v_\theta$, an approximation of $V^{p_\rho}$ since $p_\rho$ is the best available policy.

# 3. Policy Evaluation

- Evaluation function in search should *ideally* be $V^\star$. Instead use $v_\theta$, an approximation of $V^{p_\rho}$ since $p_\rho$ is the best available policy.
- $v_\theta$ trained using supervised learning; data generated by playing $p_\rho$ against itself.

# 3. Policy Evaluation

- Evaluation function in search should *ideally* be $V^\star$. Instead use $v_\theta$, an approximation of $V^{p_\rho}$ since $p_\rho$ is the best available policy.
- $v_\theta$ trained using supervised learning; data generated by playing $p_\rho$ against itself.
- Trained to minimise mean-squared error with long-term reward (game outcome: $\pm 1$). No bootstrapping.

# 3. Policy Evaluation

- Evaluation function in search should *ideally* be $V^\star$. Instead use $v_\theta$, an approximation of $V^{p_\rho}$ since $p_\rho$ is the best available policy.
- $v_\theta$ trained using supervised learning; data generated by playing $p_\rho$ against itself.
- Trained to minimise mean-squared error with long-term reward (game outcome: $\pm 1$). No bootstrapping.
- Only one training data point per game (from self-play) to eliminate correlated inputs. Training set size: 30 million.

# 3. Policy Evaluation

- Evaluation function in search should *ideally be* $V^\star$. Instead use $v_\theta$, an approximation of $V^{p_\rho}$ since $p_\rho$ is the best available policy.
- $v_\theta$ trained using supervised learning; data generated by playing $p_\rho$ against itself.
- Trained to minimise mean-squared error with long-term reward (game outcome: $\pm 1$). No bootstrapping.
- Only one training data point per game (from self-play) to eliminate correlated inputs. Training set size: 30 million.
- Many orders of magnitude faster to compute than by rollout for similar error thresholds.

# 4. Decision-time Planning

- Uses a version of MCTS in which

$$\text{ucb}(s, a) = Q(s, a) + \text{constant} \times p_\sigma(s, a) \times \frac{\sqrt{visits(s)}}{\text{visits}(s, a) + 1}.$$

- Observe that $p_\sigma$ guides exploration within the tree, but $v_\theta$ (trained to approximate $V^{p_\rho}$) is used for evaluating leaves.

# 4. Decision-time Planning

- Uses a version of MCTS in which

$$\text{ucb}(s, a) = Q(s, a) + \text{constant} \times p_\sigma(s, a) \times \frac{\sqrt{visits(s)}}{\text{visits}(s, a) + 1}.$$

- Observe that $p_\sigma$ guides exploration within the tree, but $v_\theta$ (trained to approximate $V^{p_\rho}$) is used for evaluating leaves.

- Value estimate at leaf $s$ is

$$\frac{1}{2} v_\theta(s) + \frac{1}{2} \times \text{value estimated by rollouts using } p_\pi.$$

# 4. Decision-time Planning

- Uses a version of MCTS in which

$$\text{ucb}(s, a) = Q(s, a) + \text{constant} \times p_\sigma(s, a) \times \frac{\sqrt{\textit{visits}(s)}}{\text{visits}(s, a) + 1}.$$

- Observe that $p_\sigma$ guides exploration within the tree, but $v_\theta$ (trained to approximate $V^{p_\rho}$) is used for evaluating leaves.

- Value estimate at leaf $s$ is

$$\frac{1}{2} v_\theta(s) + \frac{1}{2} \times \text{value estimated by rollouts using } p_\pi.$$

- Standard version: 48 CPUs, 8 GPUs.
- Distributed version: 1200 CPUs, 176 GPUs.

# Performance

- AlphaGo wins nearly 100% of games against other competitive (MCTS-based) agents.

# Performance

- AlphaGo wins nearly 100% of games against other competitive (MCTS-based) agents.
- Defeats Fan Hui (winner of the 2013, 2014, and 2015 European Go championships) 5–0 in formal match.

# Performance

- AlphaGo wins nearly 100% of games against other competitive (MCTS-based) agents.
- Defeats Fan Hui (winner of the 2013, 2014, and 2015 European Go championships) 5–0 in formal match.
- Program subsequently improved, with help from Fan Hui!

# Performance

- AlphaGo wins nearly 100% of games against other competitive (MCTS-based) agents.
- Defeats Fan Hui (winner of the 2013, 2014, and 2015 European Go championships) 5–0 in formal match.
- Program subsequently improved, with help from Fan Hui!

- In March 2016, AlphaGo defeats Lee Sedol (winner of 18 international titles), 4–1 in formal match.

# Performance

- AlphaGo wins nearly 100% of games against other competitive (MCTS-based) agents.
- Defeats Fan Hui (winner of the 2013, 2014, and 2015 European Go championships) 5–0 in formal match.
- Program subsequently improved, with help from Fan Hui!

- In March 2016, AlphaGo defeats Lee Sedol (winner of 18 international titles), 4–1 in formal match.
  - Move 37 by AlphaGo in Game 2 initially thought to be a “mistake” by commentators, but now praised in the Go world as “beautiful”! Unlike a move a human would play.

# Performance

- AlphaGo wins nearly 100% of games against other competitive (MCTS-based) agents.
- Defeats Fan Hui (winner of the 2013, 2014, and 2015 European Go championships) 5–0 in formal match.
- Program subsequently improved, with help from Fan Hui!

- In March 2016, AlphaGo defeats Lee Sedol (winner of 18 international titles), 4–1 in formal match.
  - Move 37 by AlphaGo in Game 2 initially thought to be a "mistake" by commentators, but now praised in the Go world as "beautiful"! Unlike a move a human would play.
  - Humans tend to optimise for win margin; AlphaGo optimises win probability.

# Performance

- AlphaGo wins nearly 100% of games against other competitive (MCTS-based) agents.
- Defeats Fan Hui (winner of the 2013, 2014, and 2015 European Go championships) 5–0 in formal match.
- Program subsequently improved, with help from Fan Hui!

- In March 2016, AlphaGo defeats Lee Sedol (winner of 18 international titles), 4–1 in formal match.
  - Move 37 by AlphaGo in Game 2 initially thought to be a "mistake" by commentators, but now praised in the Go world as "beautiful"! Unlike a move a human would play.
  - Humans tend to optimise for win margin; AlphaGo optimises win probability.

- Watch "AlphaGo - The Movie":
  `https://www.youtube.com/watch?v=WXuK6gekU1Y`.

# Getting Sleeker and Stronger

- AlphaGo Zero (Silver *et al.*, 2017)
  - Can be trained/run on single machine with 4 TPUs.
  - Tabula rasa learning; no bootstrapping from expert games.
  - Only raw features (black/white/empty) used.
  - No rollouts in MCTS.
  - Value and policy in single network (only outputs differ).
  - AlphaGo Zero beats AlphaGo Lee 100–0!

# Getting Sleeker and Stronger

- AlphaGo Zero (Silver *et al.*, 2017)
  - Can be trained/run on single machine with 4 TPUs.
  - Tabula rasa learning; no bootstrapping from expert games.
  - Only raw features (black/white/empty) used.
  - No rollouts in MCTS.
  - Value and policy in single network (only outputs differ).
  - AlphaGo Zero beats AlphaGo Lee 100–0!

- AlphaZero (Silver *et al.*, 2018)
  - Can be trained to play multiple games (chess, shogi, Go).
  - Trained/run on single machine.
  - In a few (tens of) hours of training, defeats champion agent for each game (Stockfish, Elmo, AlphaGo Lee).

# Getting Sleeker and Stronger

- AlphaGo Zero (Silver *et al.*, 2017)
  - Can be trained/run on single machine with 4 TPUs.
  - Tabula rasa learning; no bootstrapping from expert games.
  - Only raw features (black/white/empty) used.
  - No rollouts in MCTS.
  - Value and policy in single network (only outputs differ).
  - AlphaGo Zero beats AlphaGo Lee 100–0!

- AlphaZero (Silver *et al.*, 2018)
  - Can be trained to play multiple games (chess, shogi, Go).
  - Trained/run on single machine.
  - In a few (tens of) hours of training, defeats champion agent for each game (Stockfish, Elmo, AlphaGo Lee).

- 2019: Lee Sedol retires from professional play.

# Application of RL to Go

- AlphaGo
- Summary and outlook

# Review of Recent RL Successes

- (Atari 2600 DQN $\cup$ AlphaGo) $\cap$ CS 747:
  Bandits/UCB, TD learning, Function approximation, Policy gradient methods, Batch RL, Multiagent RL, MCTS.

# Review of Recent RL Successes

- (Atari 2600 DQN $\cup$ AlphaGo) $\cap$ CS 747:
  Bandits/UCB, TD learning, Function approximation, Policy gradient methods, Batch RL, Multiagent RL, MCTS.
- Neural networks a good choice of representation in many interesting domains (vision, audio, speech input).

# Review of Recent RL Successes

- (Atari 2600 DQN $\cup$ AlphaGo) $\cap$ CS 747:
  Bandits/UCB, TD learning, Function approximation, Policy gradient methods, Batch RL, Multiagent RL, MCTS.
- Neural networks a good choice of representation in many interesting domains (vision, audio, speech input).
- Other applications of Deep RL: self-driving cars, speech/dialogue systems, computer games, algorithm discovery.

# Review of Recent RL Successes

- (Atari 2600 DQN $\cup$ AlphaGo) $\cap$ CS 747:
  Bandits/UCB, TD learning, Function approximation, Policy gradient methods, Batch RL, Multiagent RL, MCTS.
- Neural networks a good choice of representation in many interesting domains (vision, audio, speech input).
- Other applications of Deep RL: self-driving cars, speech/dialogue systems, computer games, algorithm discovery.

- Successes have popularised RL, viewed as a fundamental ingredient of autonomous decision-making systems.
- Published literature in RL has exploded in the last 5–10 years.

# Review of Recent RL Successes

- (Atari 2600 DQN $\cup$ AlphaGo) $\cap$ CS 747:
  Bandits/UCB, TD learning, Function approximation, Policy gradient methods, Batch RL, Multiagent RL, MCTS.
- Neural networks a good choice of representation in many interesting domains (vision, audio, speech input).
- Other applications of Deep RL: self-driving cars, speech/dialogue systems, computer games, algorithm discovery.

- Successes have popularised RL, viewed as a fundamental ingredient of autonomous decision-making systems.
- Published literature in RL has exploded in the last 5–10 years.

- Usually needs lots of computation, data (hence simulators).
- Not the method of choice across all domains.

# Next Steps for Interested Students

- This course has covered theory and practice of basic topics related to RL.

# Next Steps for Interested Students

- This course has covered theory and practice of basic topics related to RL.

- Next set of readings: papers on selected topics.

  Inverse RL, partial observability, reward shaping; design and analysis of RL algorithms, . . . .

# Next Steps for Interested Students

- This course has covered theory and practice of basic topics related to RL.
- Next set of readings: papers on selected topics.

  Inverse RL, partial observability, reward shaping; design and analysis of RL algorithms, . . . .
- Practical experience: write code, run experiments!

  Many tasks in public domain; competitions to develop intelligent agents.

# Next Steps for Interested Students

- This course has covered theory and practice of basic topics related to RL.

- Next set of readings: papers on selected topics.

  Inverse RL, partial observability, reward shaping; design and analysis of RL algorithms, ....

- Practical experience: write code, run experiments!

  Many tasks in public domain; competitions to develop intelligent agents.

- Related courses/areas to explore:

  Game theory and multiagent systems; on-line learning; neural networks and deep learning; linear optimisation, MDPs, stochastic approximation; cognitive science, neuroscience; robotics; ....