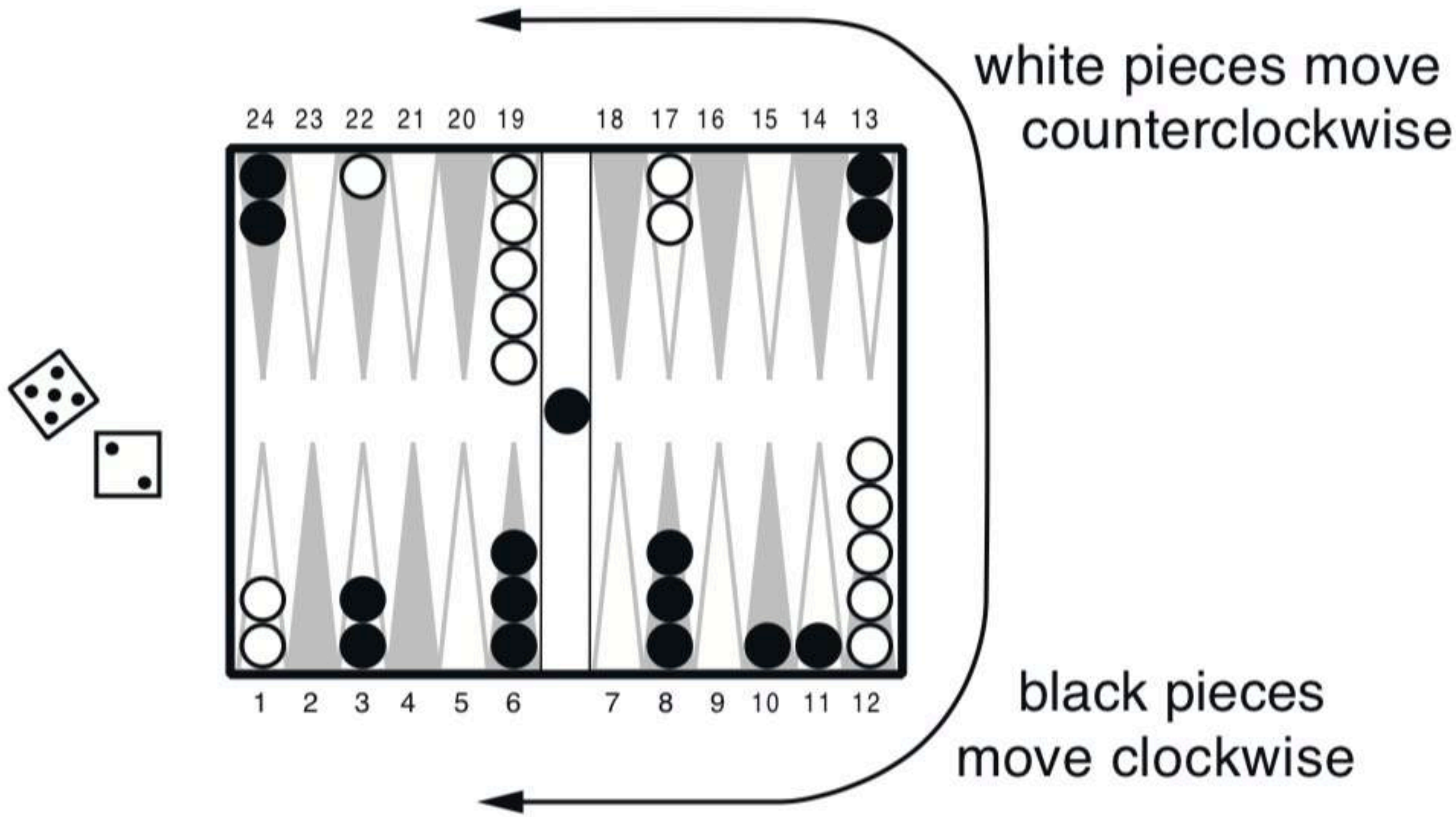
Application and Case Studies

Reinforcement Learning Seminar
19 Jan 2018

Outline

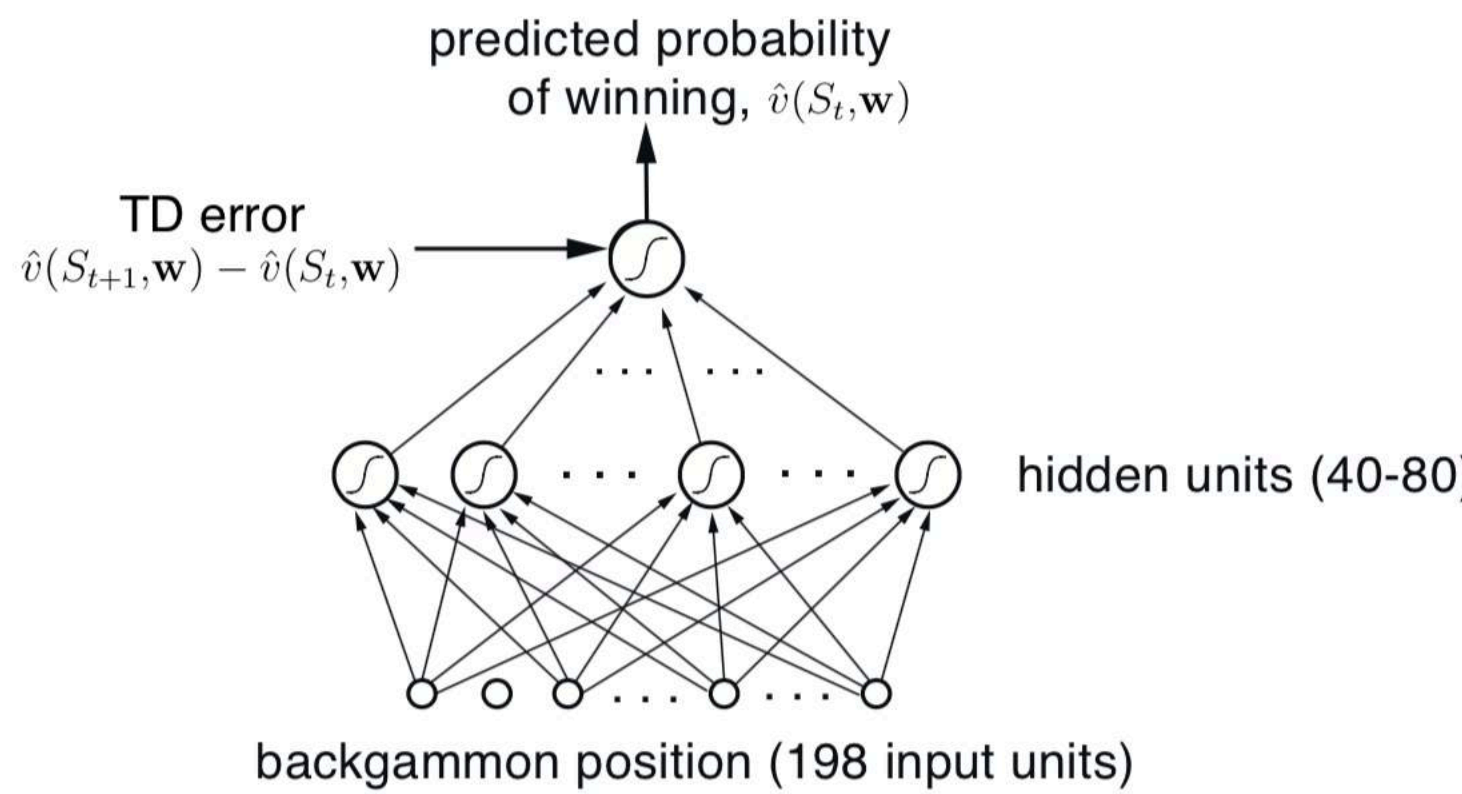
- Backgammon (双陆棋)
- Checkers (西洋跳棋)
- Daily-Double Wagering in *Jeopardy!*
- Optimal Memory Control
- Human-level Video Game Play
- Game of Go (围棋)
- Personalized Web Services
- Thermal Soaring (热动力滑翔)

Backgammon Rules



http://www.247backgammon.org

Tesauro's TD-Gammon 0.0



$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\Big[R_{t+1} + \gamma\hat{v}(S_{t+1},\mathbf{w}_t) - \hat{v}(S_t,\mathbf{w}_t)\Big]\mathbf{z}_t$$

$$\mathbf{z}_t \doteq \gamma\lambda\mathbf{z}_{t-1} + \nabla\hat{v}(S_t,\mathbf{w}_t)$$

gradient from BP

198 input units:
4(num of pieces)*2(black/white)*24(points)
+ 2(pieces removed from board) + 2(pieces on the bar)
+ 2(black/white turn)

Other Versions of TD-Gammon

Program	Hidden Units	Training Games	Opponents	Results
TD-Gammon 0.0	40	300,000	other programs	tied for best
TD-Gammon 1.0	80	300,000	Robertie, Magriel, ...	−13 pts / 51 games
TD-Gammon 2.0	40	800,000	various Grandmasters	−7 pts / 38 games
TD-Gammon 2.1	80	1,500,000	Robertie	−1 pt / 40 games
TD-Gammon 3.0	80	1,500,000	Kazaros	+6 pts / 20 games

- **TD-Gammon 1.0: add specialized backgammon features**
- **TD-Gammon 2.0: add selective two-ply search procedure, with 40 hidden units**
- **TD-Gammon 2.1: add selective two-ply search procedure, with 80 hidden units**
- **TD-Gammon 3.0: selective three-ply search procedure, with 169 hidden units**

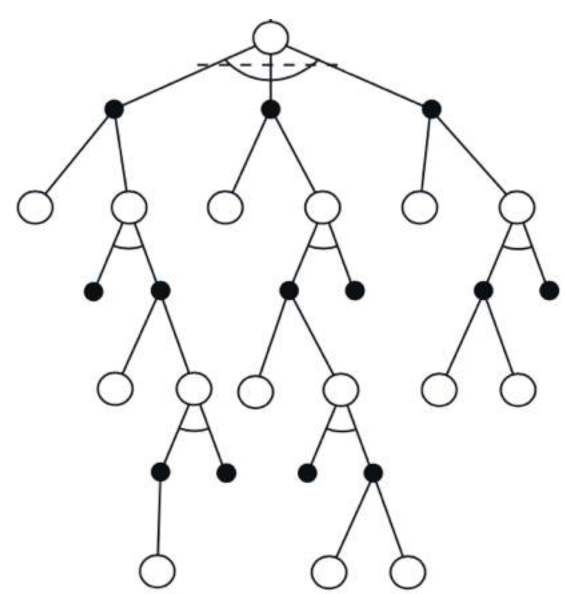
Samuel’s Checkers Player

http://www.247checkers.com/

- minmax procedure: leaf (value function) -> root (backed-up value)

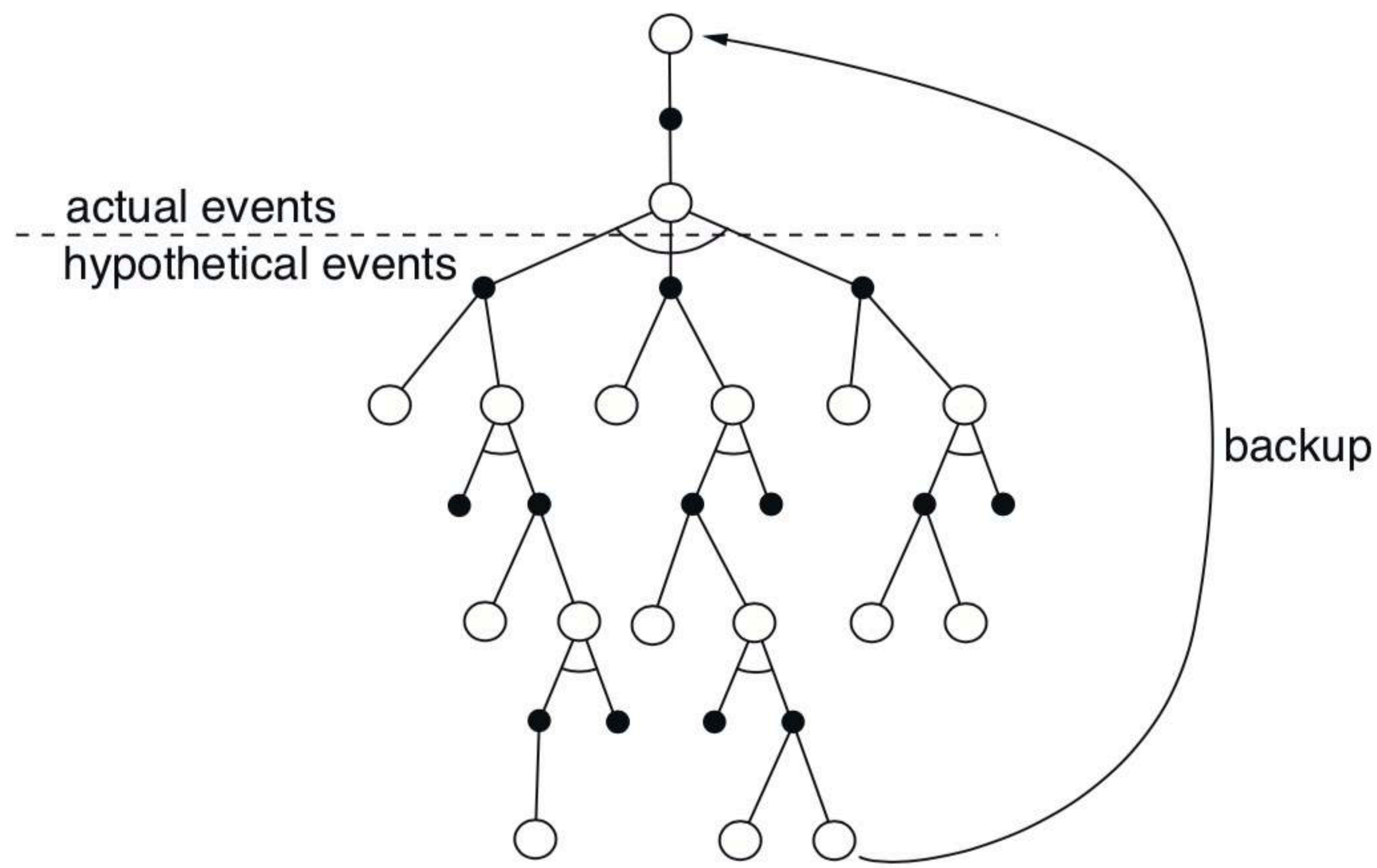
- rote-learning: save a description of each board position encountered during play together with their backup value

- a sense of direction: decreasing a position’s value a small amount each time it was backed up



Samuel's Checkers Player

- learning by generalization: modify the parameters of the value function



Samuel’s Checkers Player

- Problems
  - no rewards upon the end of game -> value function become consistent merely by giving a constant to all positions
  - temporary solution: give *piece-advantage* a large, non-modifiable weight & set other weights back to zero if they gain large absolute values
- Aware the value of a state should equal to the value of likely following state, but there’s no TRUE value defined.

Daily-Double Wagering in *Jeopardy!* : Rules

- First two rounds: select a clue, announce the clue, first buzzing in to answer
- DD: bet more than $5 and less than owned
- Final round: seal the answer and bet
- Information is incomplete

WATSON

$$\hat{q}(s, bet) = p_{DD} \times \hat{v}(S_W + bet, \ldots) + (1 - p_{DD}) \times \hat{v}(S_W - bet, \ldots),$$

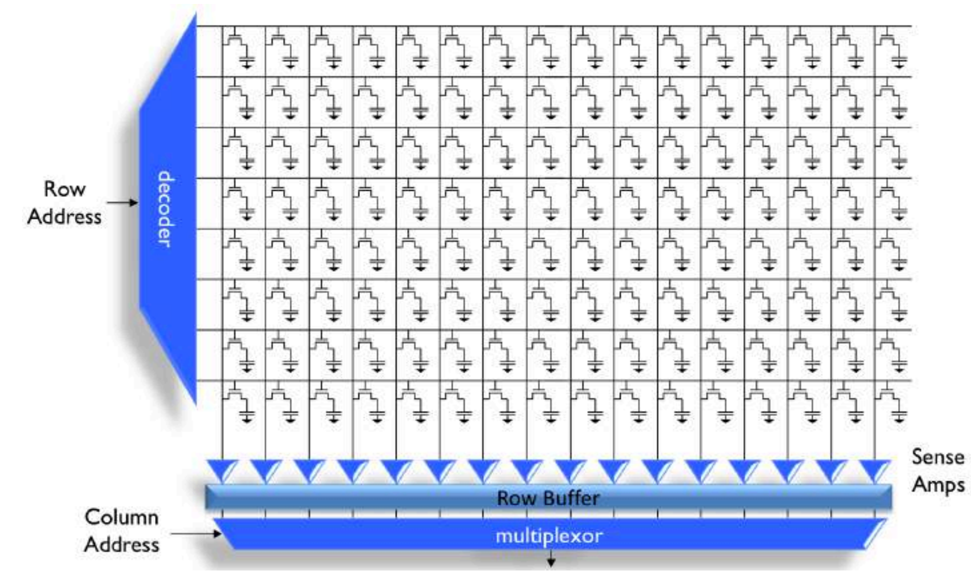
p_{DD}	**estimated from practice data**
$\hat{v}(S_W + bet, \ldots)$	**learn from LR (play against human model)**

- decrease downside risk
    - decrease estimated confidence on itself
    - prevent large bet

WATSON: Result

- win rate from 61% to 67%
- considering DD is needed only 1.5~2 times in each game

Optimizing Memory Control

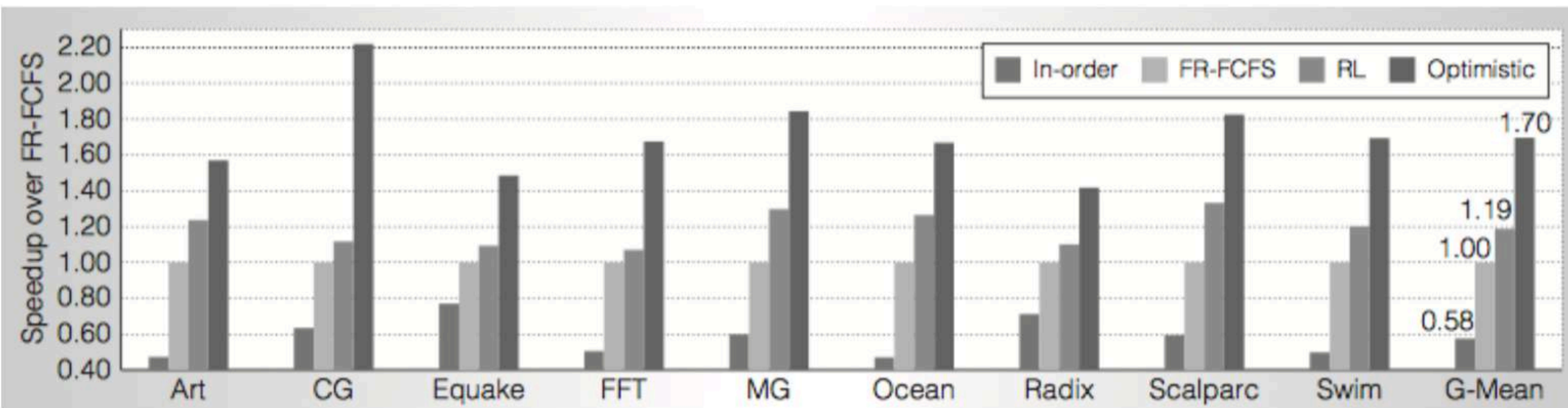


- DRAM structure: w/r via row buffer
- DRAM operation
  - row commands: activate / precharge
  - column commands: read / write
- Objective: minimize latency / maximize throughput
- Planning can minimize latency, e.g. execute several column commands on the same row together

Optimizing Memory Control: Turn into RL Problem

- reward: r/w -> 1 otherwise -> 0
- state: contents of transaction queue
- state feature: 6 integer vector and tile coding
- action: precharge, activate, read, write, noOp
- make system safe from timing and resource restrictions: noOp $A_t \in \mathcal{A}(S_t)$
- SARSA with linear approximation

Optimizing Memory Control: Result



Human-level Video Game Play: Problem Description

- Atari Games: 210x160 pixels 128-color 60Hz video games
- Objective:
  - up to 18 kinds of operations
  - score as high as possible
  - same algorithm and neural network structure for 40+ different games

Figure 1: Screen shots from five Atari 2600 Games: (*Left-to-right*) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

Human-level Video Game Play: Detail

- DQN $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \Big[R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t) \Big] \nabla_{\mathbf{w}_t} \hat{q}(S_t, A_t, \mathbf{w}_t),$
- reward:
  - score increase in the next step: +1
  - score decrease in the next step: -1
  - score unchanged: 0
  - reward can work regardless of different score ranges in different games

Human-level Video Game Play: Detail

- 210x160 pixels 128-color -> 84x84 illuminant pixel and 4 recent frames
- Q(s, a) is given by a neural network
  - input: 84x84x4
  - output: 18 (corresponding to up to 18 operations)
  - structure: Conv(20x20x32) Conv(9x9x64) Conv(7x7x64) FC(512) Out(18) activation: ReLU

Human-level Video Game Play: Contribution

- experience replay: add tuple to replay memory and Q-learning update a mini-batch uniformly sampled from replay memory $(S_t, A_t, R_{t+1}, S_{t+1})$
- advantage:
  - each experience can be learned multiple times
  - reduce variance in weight updating
  - reduce instability induced by experiences based on similar weights

Human-level Video Game Play: Contribution

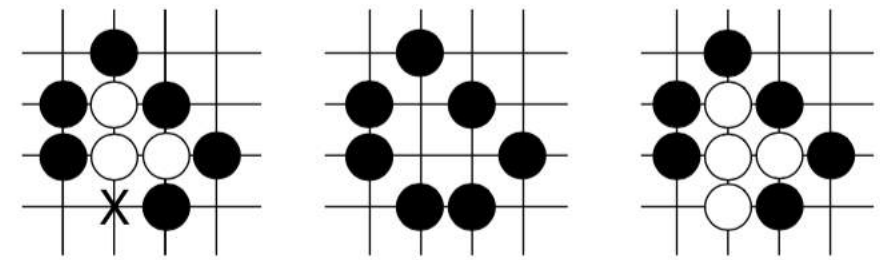
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\Big[R_{t+1} + \gamma \max_a \tilde{q}(S_{t+1}, a, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)\Big]\nabla_{\mathbf{w}_t}\hat{q}(S_t, A_t, \mathbf{w}_t).$$

- use a duplicated network:
  - weights in duplicated network updated every C steps
  - reduce instability

Human-level Video Game Play: Results

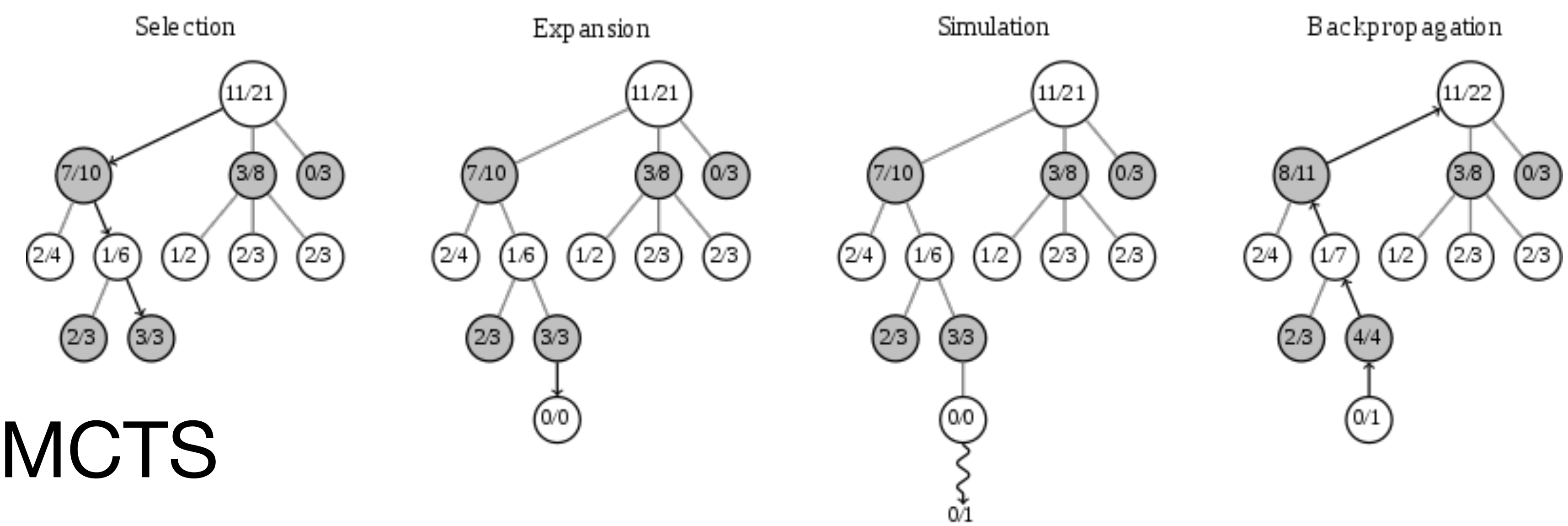
- Training: 50 million frames (38 days of experience)
- Testing: 5min session x 30 (with random initial state)
- Testing for human: 2hrs practice, 5min x 20
- compared by score
- 29/46 games reached or exceeded human level (greater or equal to 75% of human's score)

The Game of Go: Problem Description



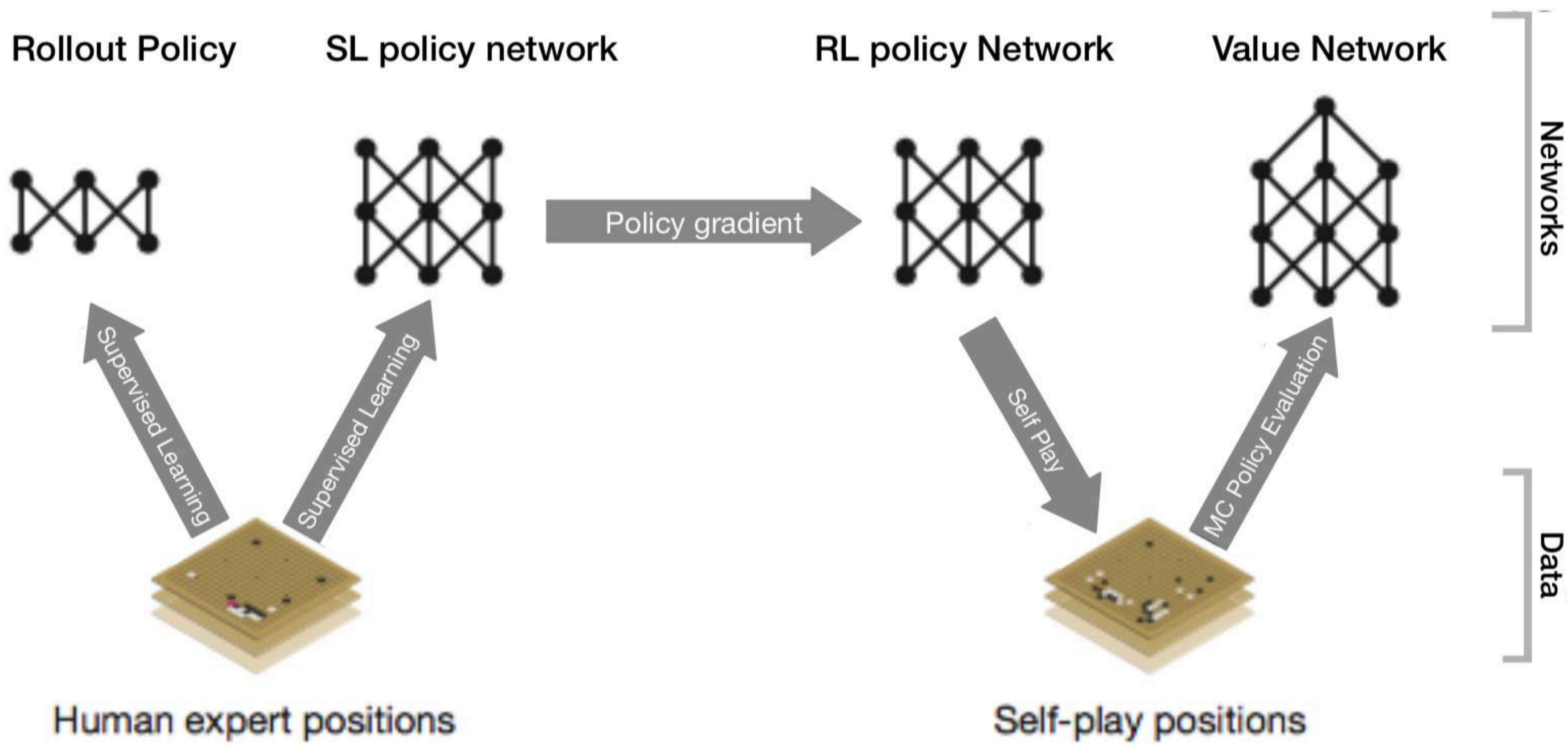
- Game of Go
- Difficulty:
  - Search space is significantly large
  - Not easy to find a simple evaluation function

The Game of Go: Runtime Frame Work



- APV-MCTS
  - expansion: by SL policy network
  - simulation: by rollout policy
  - evaluation: searched reward together with a value network $v(s) = (1-\eta)v_\theta(s) + \eta G,$

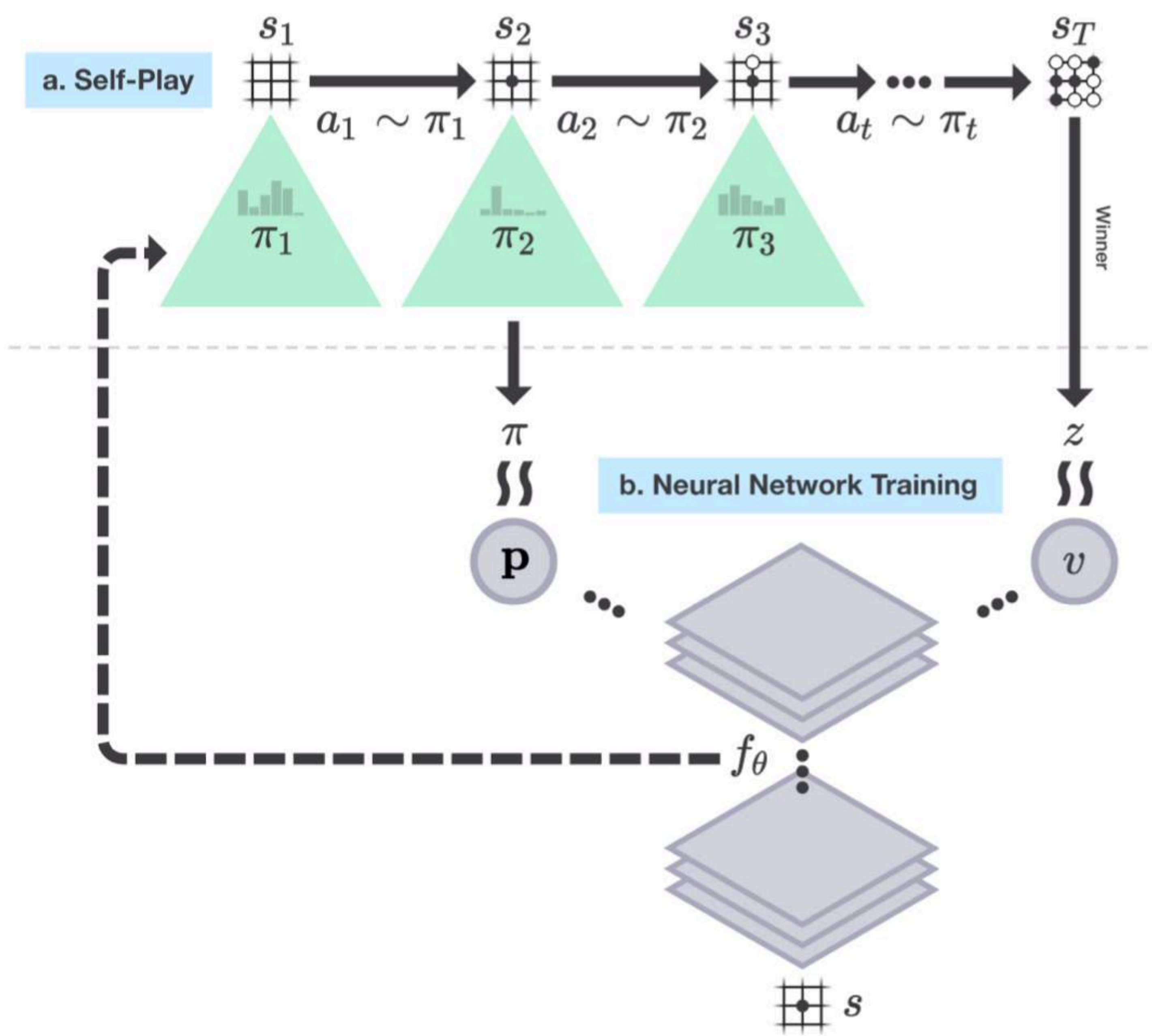
The Game of Go: AlphaGo Pipeline



The Game of Go: Detail

- input feature: 19x19x48 many special designed feature for the game of go - binary/integer value
- self-play against a randomly selected policies produced by earlier iterations of learning algorithm -> prevent overfitting

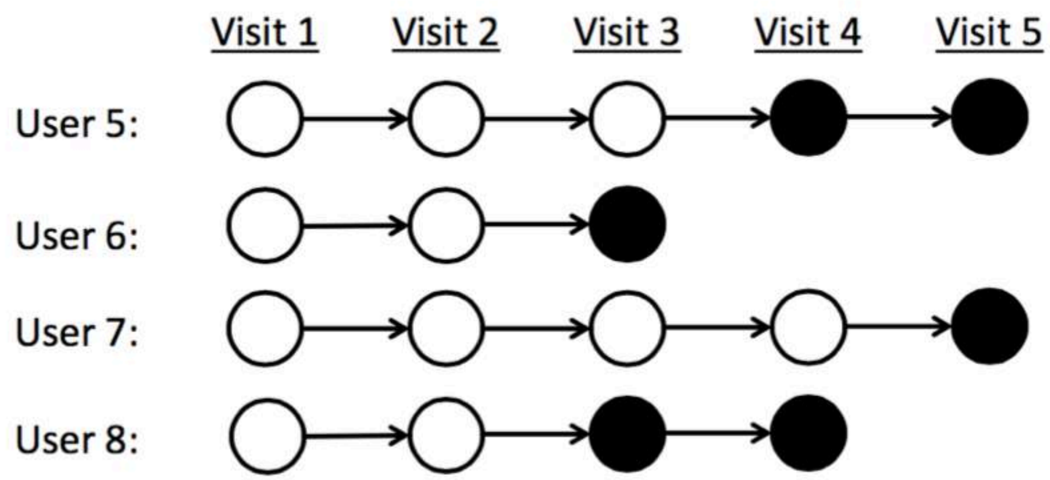
The Game of Go: AlphaGo Zero



Personalized Web Services

$$\text{CTR} = \frac{\text{Total \# of Clicks}}{\text{Total \# of } \textbf{Visits}}$$

$$\text{LTV} = \frac{\text{Total \# of Clicks}}{\text{Total \# of } \textbf{Visitors}}$$



$$\text{CTR} = \frac{6}{17} \approx 0.35$$

$$\text{LTV} = \frac{6}{4} = 1.5$$

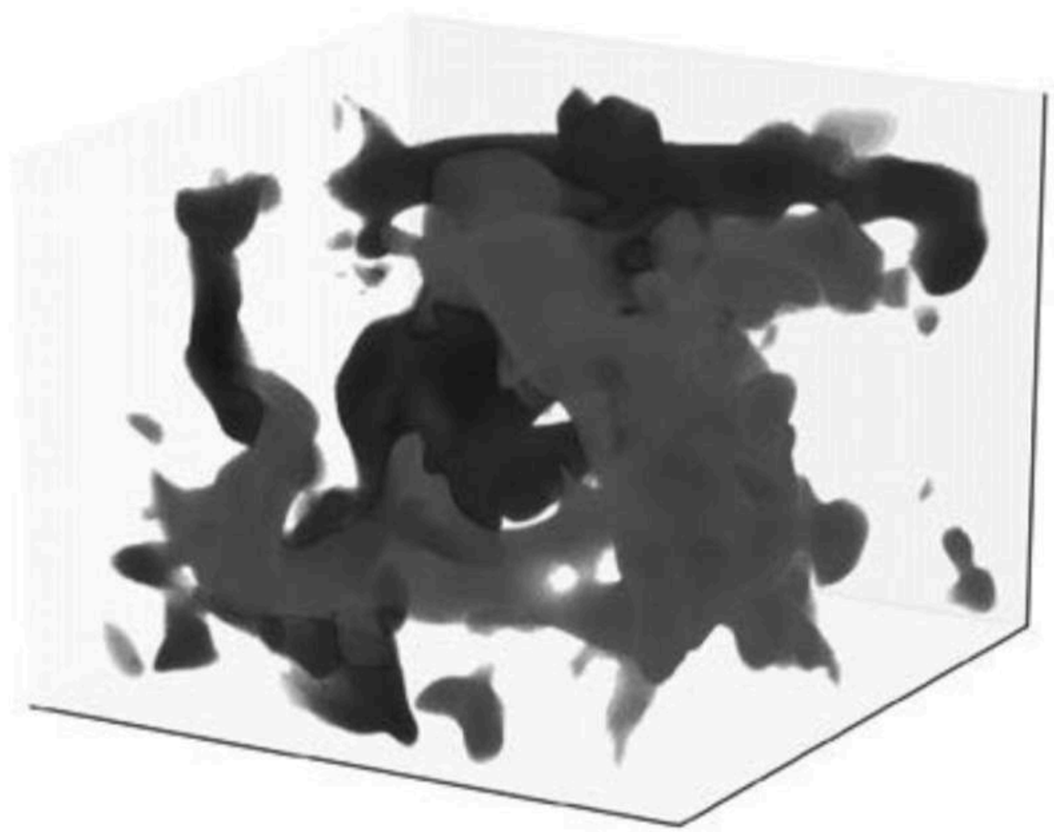
Greedy Optimization → **LTV Optimization**

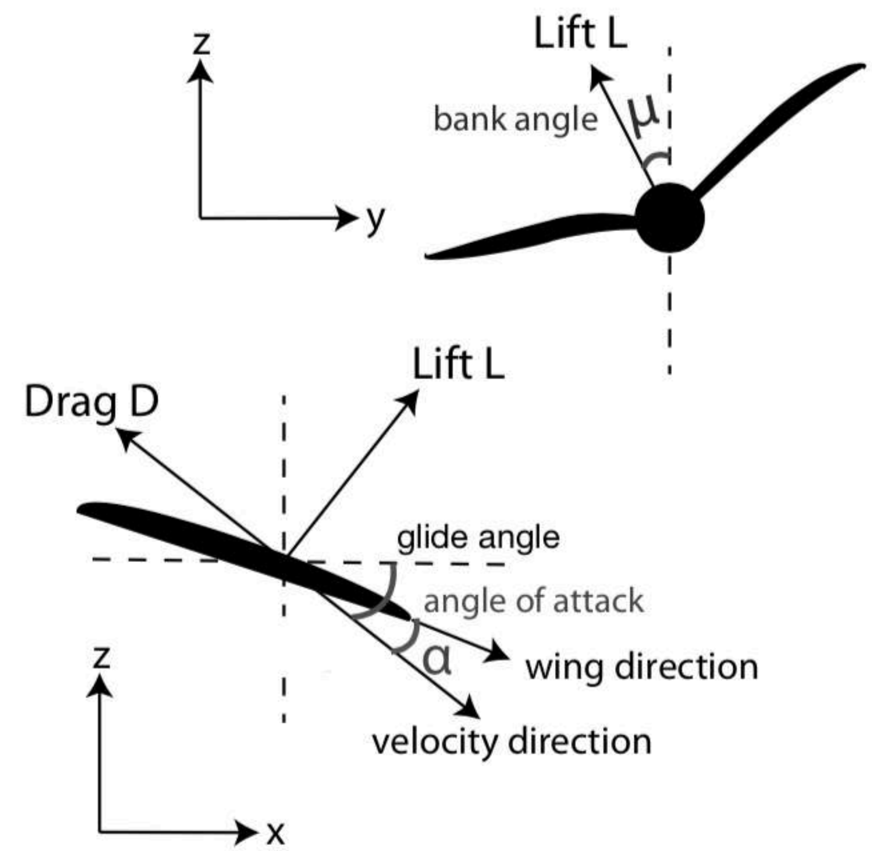
$y = \mathbf{X}_{\text{train}}(\text{reward})$
$x = \mathbf{X}_{\text{train}}(\text{features})$
$\bar{x} = \text{informationGain}(x, y)$ {feature selection}
$\text{rf}_a = \text{randomForest}(\bar{x}, y)$ {for each action}
$\pi_e = \text{epsilonGreedy}(\text{rf}, \mathbf{X}_{\text{test}})$

$r = \mathbf{X}_{\text{train}}(\text{reward})$ {use recurrent visits}
$x = \mathbf{X}_{\text{train}}(\text{features})$
$y = r_t + \gamma \max_{a \in A} Q_a(x_{t+1})$
$\bar{x} = \text{informationGain}(x, y)$ {feature selection}
$Q_a = \text{randomForest}(\bar{x}, y)$ {for each action}
$\pi_e = \text{epsilonGreedy}(Q, \mathbf{X}_{\text{val}})$

(iteratively)

Thermal Soaring





State feature: local vertical wind speed, local vertical wind accelerate, torque by wind, local temperature
Actions: (increase/decrease) (bank angle/ angle of attach) (0, 2.5°, 5°)
Objective: gain as much altitude as possible
Method: SARSA (Simulation by 2.5min episodes with 1s time step in 1km³ box)

Summary

- Backgammon
- Checkers
- Daily-Double Wagering in *Jeopardy!*
- Optimal Memory Control
- Human-level Video Game Play
- Game of Go
- Personalized Web Services
- Thermal Soaring