An Introduction to Reinforcement Learning

Yiping Qi

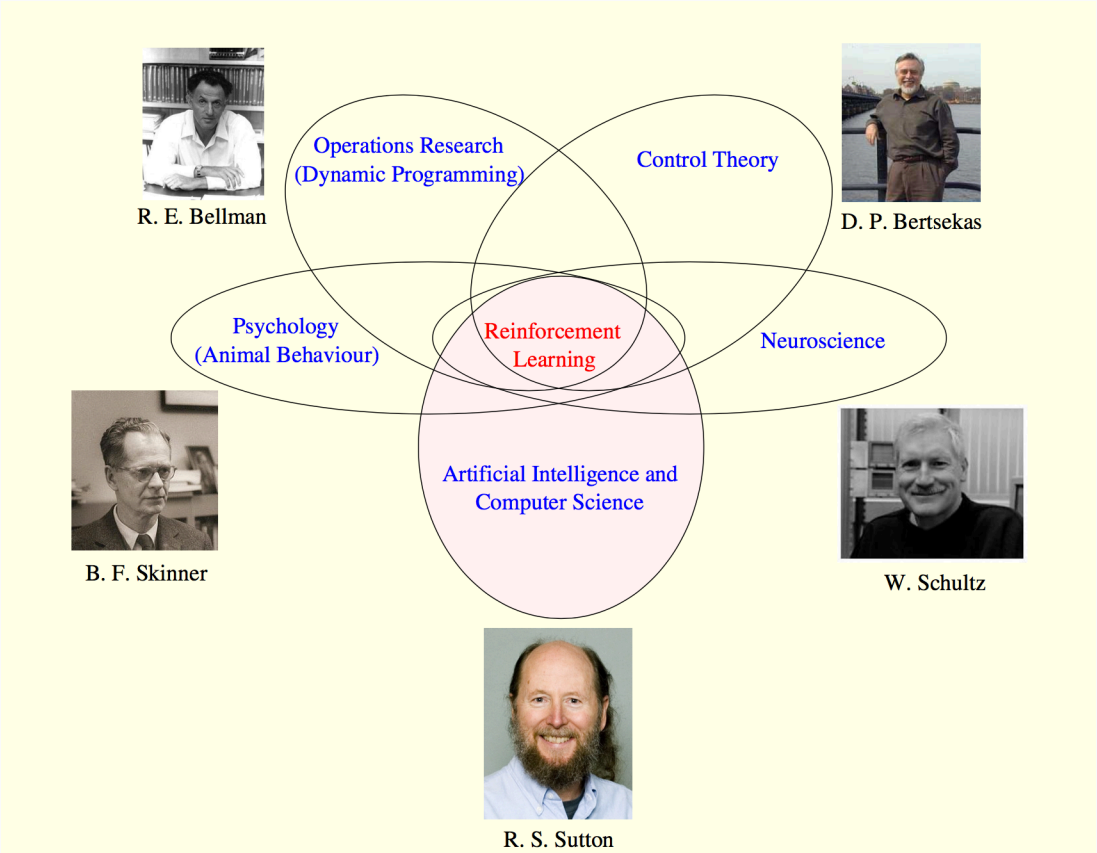
November 30, 2016

TOC

Reinforcement Learning Problem

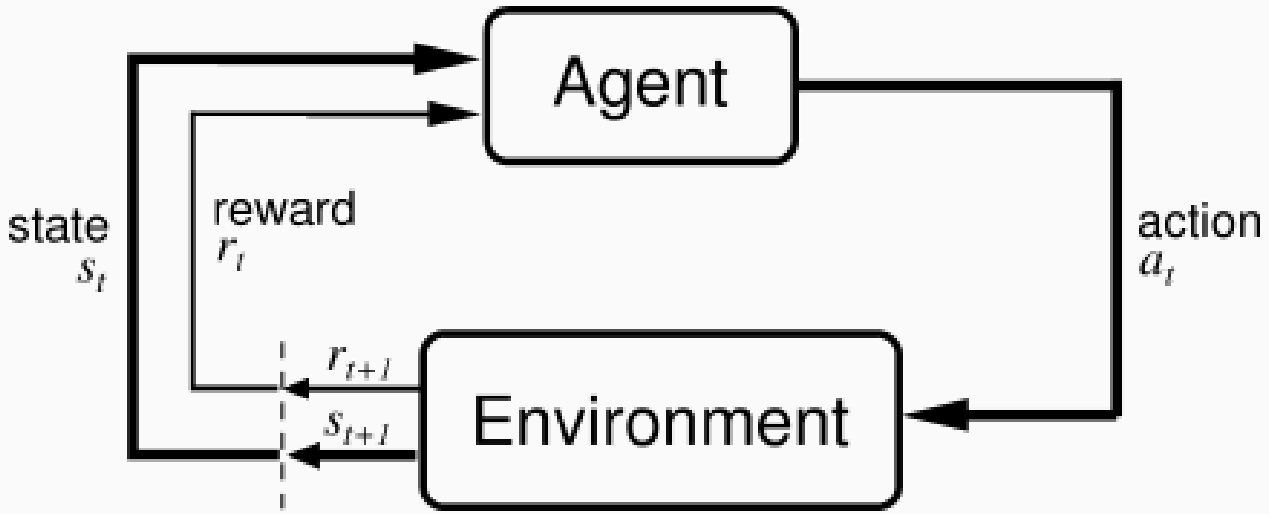
Reinforcement Learning

- Machine Learning
  - Supervised Learning
  - Unsupervised Learning
  - **Reinforcement Learning**[2]
- View from different fields



Characteristics of RL Problem

- There is no supervisor, only **reward**.
- Feedback is delay.
- Agent's action affects the environment.



Examples of RL Problems

- Games: Othello, Go, ...
- Self driven car
- Financial investment
- More successful stories: `http://umichrl.pbworks.com/Successes-of-Reinforcement-Learning/`

Reward and Goal

- The purpose or goal of the agent is formalized in terms of a special reward signal (r_t).
- The agent selects action to maximize expected future return (R_t).

Reward Hypothesis

All goals can be described by the maximisation of expected cumulative reward (R_t).

Finite horizon: $R_t = r_{t+1} + r_{t+2} + \ldots + r_T$

Infinite horizon:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Examples of Rewards

- Playing Othello
  - 0 for all non-terminal stages
  - +/-1 for winning/losing a game
- Financial investment
  - +/- reward for each $ in your account

State, Action and Policy

- $s_t \in S$: representation of the environment's state.
- $a_t \in \mathcal{A}(s_t)$: set of action available in state s_t.
- $\pi(s_t)$: rule of agent's behavior(map from state to action).
  - Deterministic policy: $a = \pi(s)$
  - Stochastic policy: $P(a|s) = \pi(s)$

Markov Property

$$P(s_{t+1}|s_t, s_{t-1}, \ldots, s_0) = P(s_{t+1}|s_t)$$

s_t contains all useful information from the history.

s_t is a sufficient statistic of the future.

Value Function

- Value function
  - Functions of states (or of state-action pairs) that estimate the expected future return.
  - It depends on the agent's policy.

State Value Function

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}$$

State-Action Value Function

$$Q^{\pi}(s, a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

Model-Based/Model-Free

- The **model** formulate how the environment work.
  - $P(s_{t+1}|s_t, a_t)$: prediction of the next state
  - $R(s_t, a_t)$: prediction of the immediate reward
- Model-Free
  - Policy and/or Value Function
  - No Model
- Model-Based
  - Policy and/or Value Function
  - Model

Value-Based/Policy-Based

- Value Based
  - No Policy (deduced from value function)
  - Value Function
- Policy Based
  - Policy
  - No Value Function
- Actor-Critic
  - Policy
  - Value Function

Value Based Method

Bellman's Optimality Equation

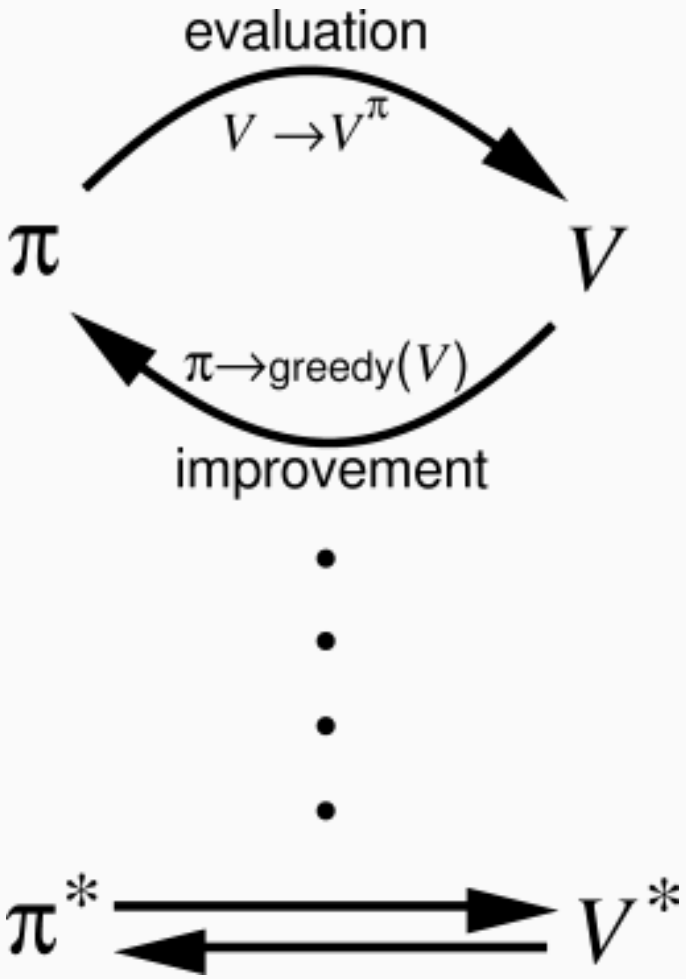
- Bellman's Optimality Equation

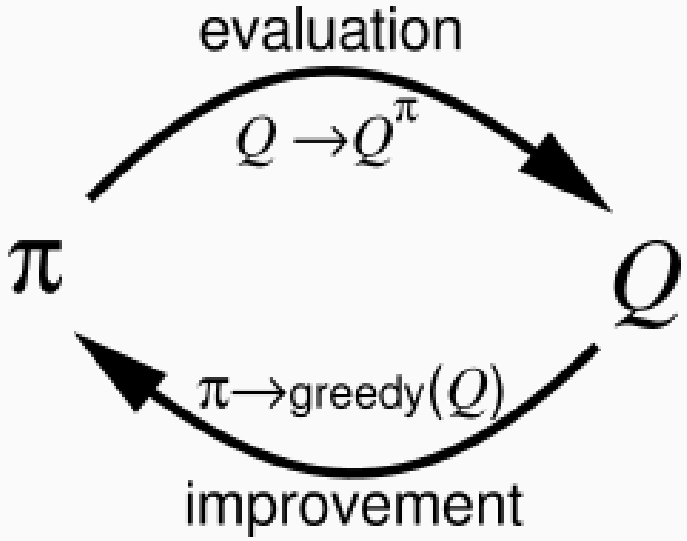
$$\exists \pi^* \in \Pi, \forall \pi \in \Pi, \forall s \in S, V^{\pi^*}(s) \geq V^{\pi}(s)$$

$$V^{\pi^*}(s) = \max_a \sum_{s'} P(s'|s,a)(r + \gamma V^{\pi^*}(s'))$$

Hard to solve Bellman's Optimality Equation directly.

Generalized Policy Iteration





Generalized Policy Iteration

- Policy Evaluation
  - Evaluate

$$V^{\pi}(s)$$

$$Q^{\pi}(s, a)$$

- Policy Improvement
  - Greedy Policy

$$\begin{aligned}\pi'(s) &= \arg\max_a Q^{\pi}(s, a) \\ &= \arg\max_a \sum_{s'} P(s'|s, a)(r + \gamma V^{\pi}(s'))\end{aligned}$$

Dynamic Programming

Dynamic Programming is a model based method.

DP is a bootstrapping method.

DP is based on Bellman's Equation.

Input π, the policy to be evaluated
Initialize $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx V^\pi$

Monte Carlo

Monte Carlo Method is a model free method.

Initialize:

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ an arbitrary state-value function

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

(a) Generate an episode using π

(b) For each state s appearing in the episode:

$R \leftarrow$ return following the first occurrence of s

Append R to $Returns(s)$

$V(s) \leftarrow \text{average}(Returns(s))$

Incremental Mean

$$
\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}(\sum_{j=1}^{k-1} x_j + x_k) \\
&= \frac{1}{k}((k-1)\mu_{k-1} + x_k) \\
&= \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})
\end{aligned}
$$

Temporal Difference

- Monte Carlo

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \frac{1}{k}(R_t(s_t) - V^{\pi}(s_t))$$

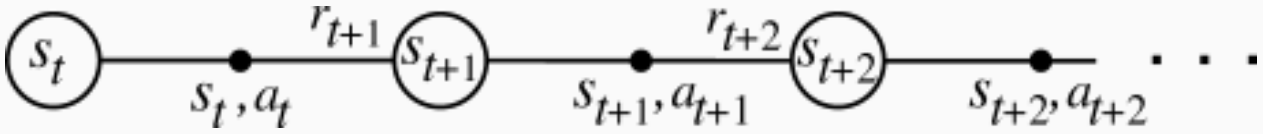
- TD(0)
  - Tracking a non-stationary problems, use running mean

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha(R_t(s_t) - V^{\pi}(s_t))$$

  - estimate $R_t(s_t)$ by $R_t^{(1)} = r_t + V^{\pi}(s_{t+1})$

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha(r_t + V^{\pi}(s_{t+1}) - V^{\pi}(s_t))$$

Temporal Difference



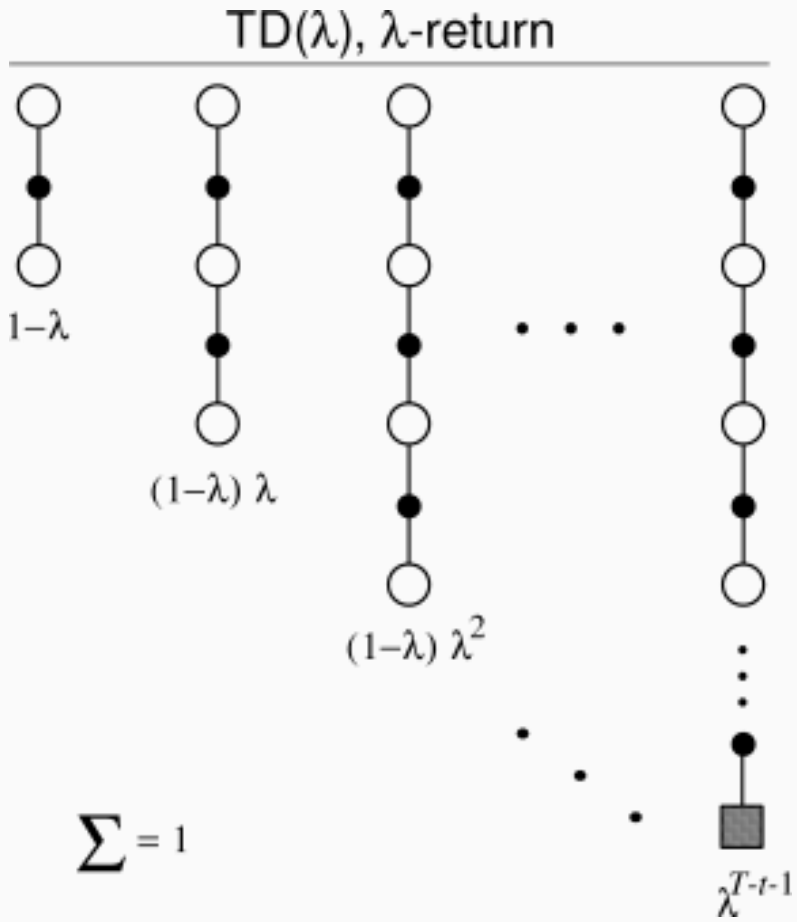
TD methods have an advantage over DP methods in that they do not require a model of the environment, of its reward and next-state probability distributions.

TD methods over Monte Carlo methods is that they are naturally implemented in an on-line, fully incremental fashion.

Bias/Variance Trade-Off

- $R_t(s_t)$ is an unbiased estimate of $V^\pi(s_t)$, but has higher variance. (Depends on many random actions, transitions, rewards).
- $R_t^{(1)}(s_t)$ is a biased estimate of $V^\pi(s_t)$, but has lower variance. (Depends on one random action, transition, reward).

TD(λ)



Estimate $R_t(s_t)$ by $R_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}$

Large Scale RL

State space S and action space A can be huge.

- Go: 10^{170} states

Curse of dimensionality

Lookup table methods: intractable

Value Function Approximation

Estimate value function with function approximation

$$\hat{V}(s, w) \approx V^{\pi}(s)$$
$$\hat{Q}(s, a, w) \approx Q^{\pi}(s, a)$$

Supervised models are used.

- **Linear model**
- **Neural network**
- Regression Tree
- ...

Explore and Exploit

Greedy Policy

$$\pi'(s) = \arg\max_a Q^\pi(s, a)$$

Problem

- Greedy policy may lock into a sub-optimal action forever.
- Very small part of the state space will be explored.

Explore and Exploit

- ϵ-greedy action selection
  - With probability $(1 - \epsilon)$, select $a = \arg\max_a Q^\pi(s, a)$.
  - With probability ϵ, select action randomly in $\mathcal{A}(s)$.
- Softmax action selection
  - Select action a with probability $\frac{e^{Q^\pi(s,a)/\tau}}{\sum_{b \in \mathcal{A}(s)} e^{Q^\pi(s,b)/\tau}}$

Policy Based Method

Policy Based Method

Directly parameterise the policy

$$\pi_\theta(s, a) = P(a|s)$$

Pros:

- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies

Cons:

- Typically converge to a local rather than global optimum
- Evaluating a policy is typically inefficient and high variance

Policy Gradient

Policy objective

$$J(\theta) = E\{V^{\pi_\theta}\}$$

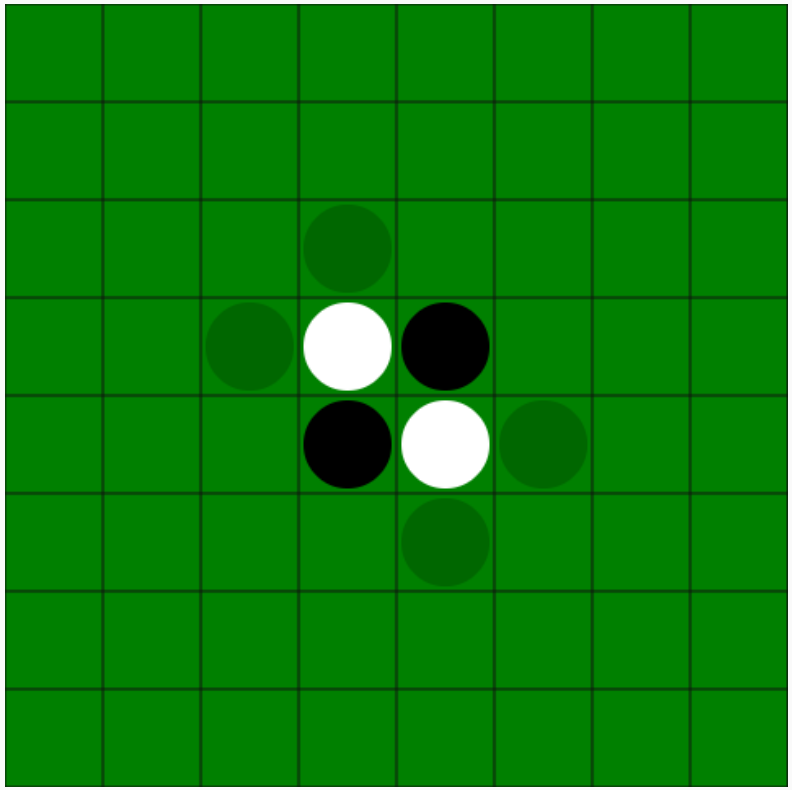
Policy gradient

$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$

Case Study

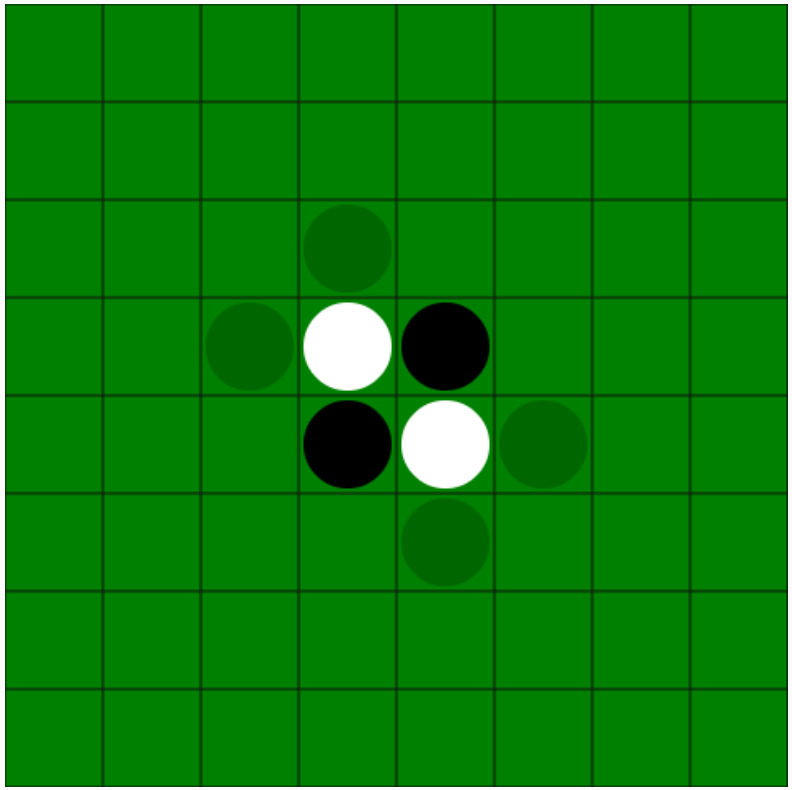
Othello Game

- `https://en.wikipedia.org/wiki/Reversi`
- Rule: "Dark must place a piece with the dark side up on the board, in such a position that there exists at least one straight (horizontal, vertical, or diagonal) occupied line between the new piece and another dark piece, with one or more contiguous light pieces between them."



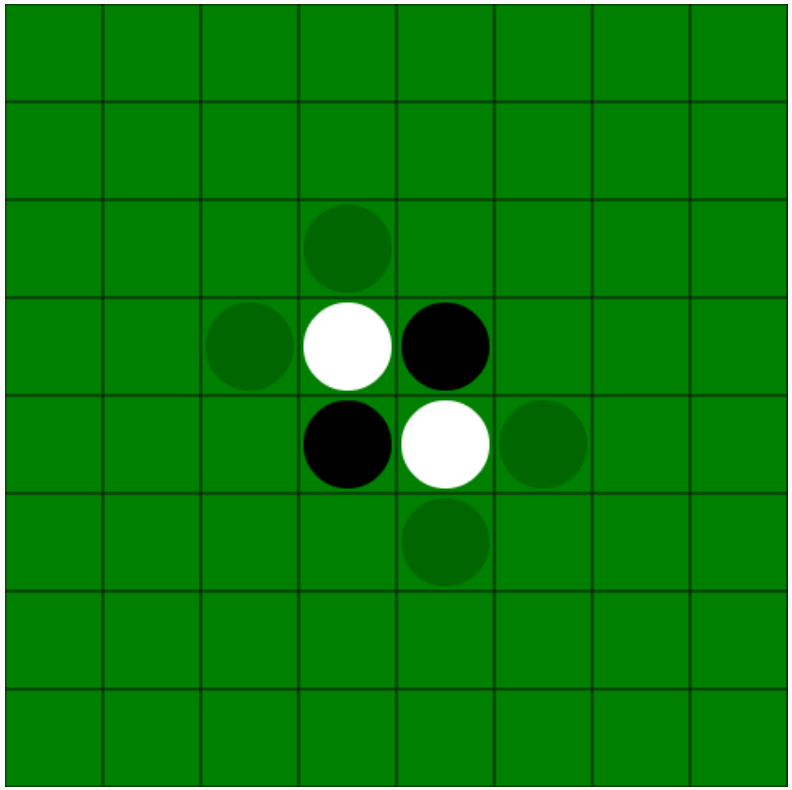
Othello Common Strategy

- Corners
- Edges
- Mobility
- ...



Othello RL Formulation

- State s_t: board status
- Action $\mathcal{A}_t$: $\mathcal{A}_t(s_t)$ is the feasible positions' set



Othello RL Formulation (Cont'd)

- Reward r_t
  - 0 for all non-terminal stages
  - $C_{black} - C_{white}$ for terminal stage

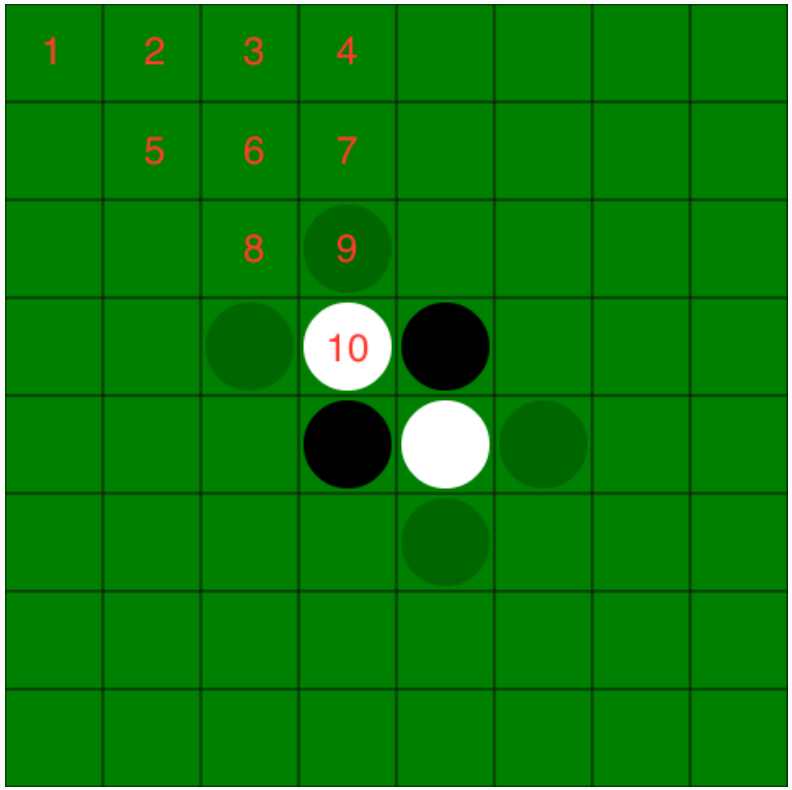
Othello RL Formulation (Cont'd)

- Value Function Approximation $V^{\pi}(s_t)$
  - state-space complexity[1]: 10^{28}
  - symmetries of board
  - n-tuple feature[1]
  - linear value function: $V^{\pi}(s_t) = w^T f(s_t)$

[1] https://en.wikipedia.org/wiki/Game_complexity

Othello RL Formulation (Cont'd)

- simplest 1-tuple form:
  $10 \times 3 = 30$-dim feature



Othello RL Formulation (Cont'd)

- Policy
  - Dark
    - Choose $a_t \in \mathcal{A}_t$, such that $V^{\pi} s_{t+1}$ is maximized.
  - Bright
    - Choose $a_t \in \mathcal{A}_t$, such that $V^{\pi} s_{t+1}$ is minimized.

Policy Evaluation And Improvement

- Self-Play
- TD(0)
  - non-terminal stage

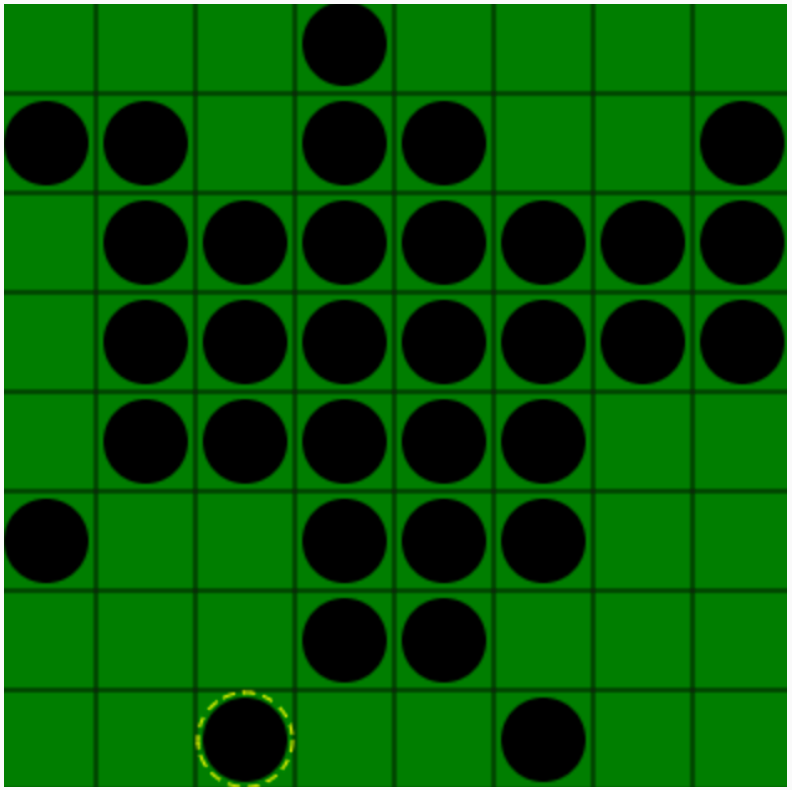
$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha(R_t(s_t) - V^{\pi}(s_t))$$
$$= V^{\pi}(s_t) + \alpha(V^{\pi}(s_{t+1}) - V^{\pi}(s_t))$$

  - terminal stage

$$V^{\pi}(s_t) \leftarrow C_{black} - C_{white}$$

Explore and Exploit

- State space is not well explored.
- ϵ-greedy action selection.
  - With probability $(1 - \epsilon)$, select $a_t = \arg\max_{a \in \mathcal{A}_t} V^{\pi}(s_{t+1})$.
  - With probability ϵ, select action randomly in $\mathcal{A}_t(s_t)$.



Reference

Reference

Wojciech Jaśkowski.
Systematic n-tuple networks for othello position evaluation.
ICGA Journal, 37(2):85–96, 2014.

Richard S Sutton and Andrew G Barto.
***Reinforcement learning: An introduction*, volume 1.**
MIT press Cambridge, 1998.

Q&A