Introduction to Reinforcement Learning (RL)

Billy Okal
Apple Inc.

What is RL

Reinforcement learning is a paradigm for **learning** to make a **good sequence** of decisions

Reinforcement Learning in Context

Reinforcement Learning in Context

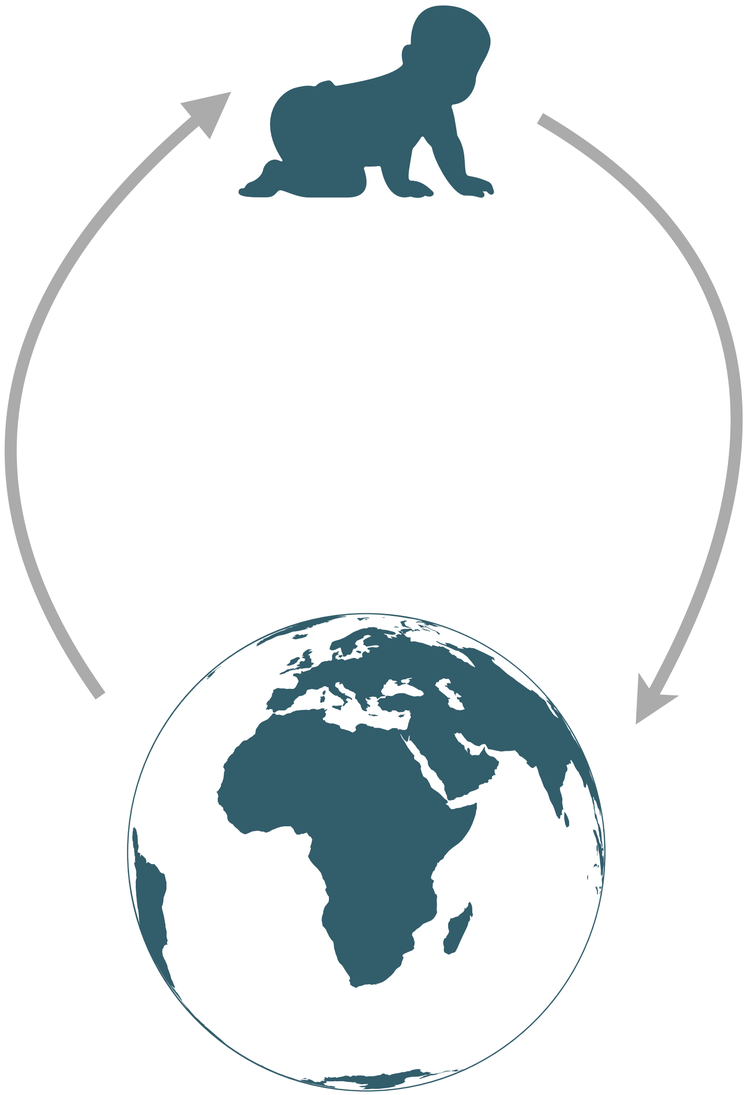
Supervised Learning	Reinforcement Learning	Unsupervised Learning
Data	Data	Data
Optimization	Optimization	Optimization
	Long-term consequences	
	Exploration	

RL Protocol

1. An agent takes an action in an environment

2. The environment responds with a feedback signal

3. The agent uses the feedback to decide on future actions

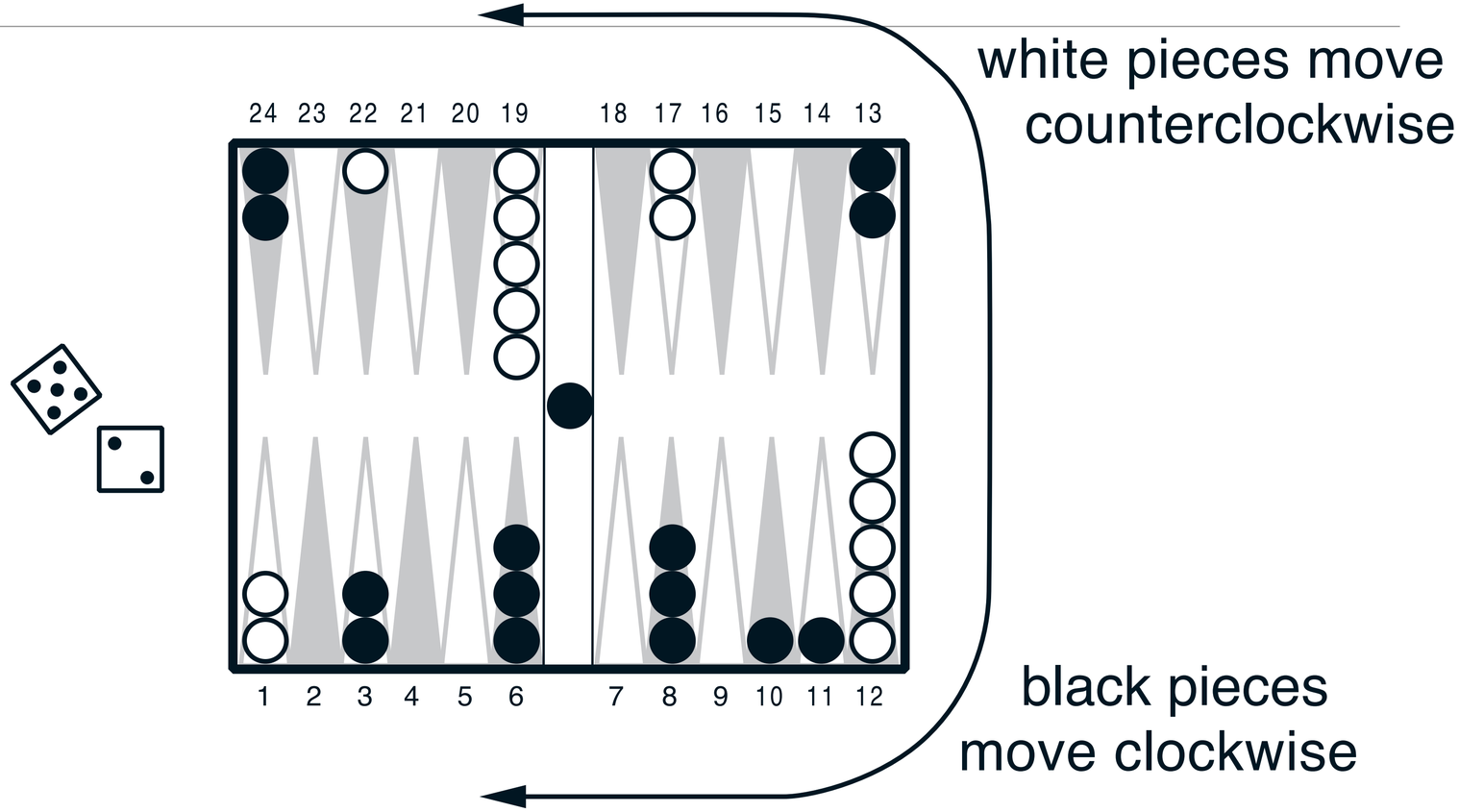


Example applications of RL

- Games — Go, video console games
- Robotics — drone, mobile robot navigation
- Medicine — administering trials
- Dialogue systems, e.g. chatbots
- Personalized web — internet ads, news feeds
- Finance — trading
- Process optimization — DRAM, elevator dispatch

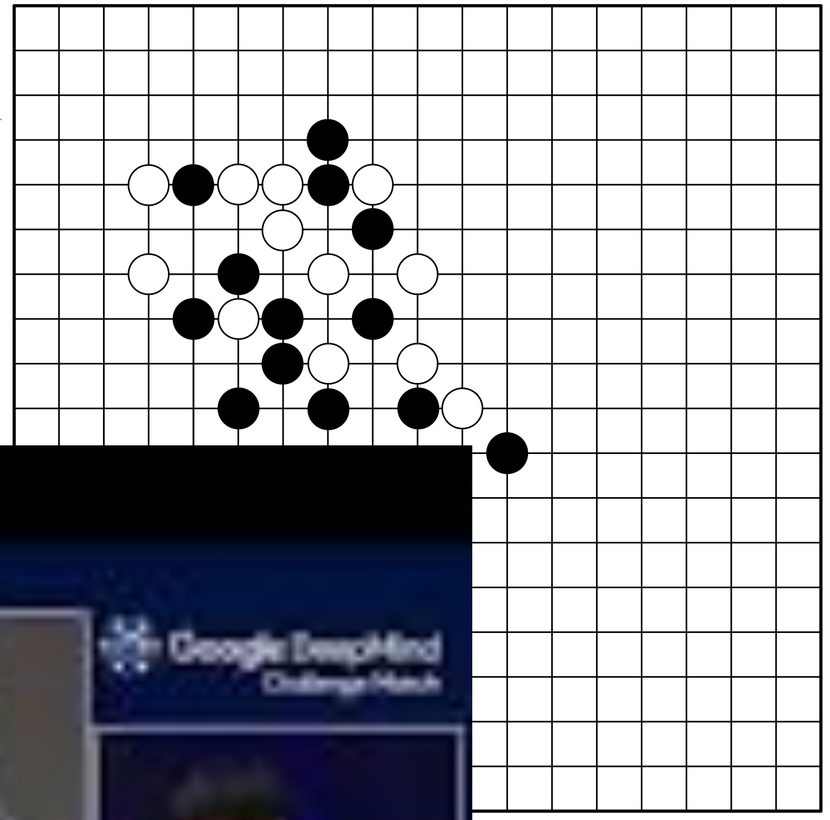
Example applications of RL

- TD-Gammon for the game backgammon
- Early 90s



Example applications of RL

- AlphaGo
- Very recent
- Combines RL with other learning approaches



Characteristics of RL tasks

- **Interaction!**
- When task requires making a **sequence** of decisions
- There is **feedback** resulting from the choice of state and/or actions
- Data is in the form of trajectories

Mathematical Formalism

Markov Decision Processes | Notation

- States: $\mathcal{S} = \{s, s'\}$ S_t, S_{t+1}
- Actions: $\mathcal{A} = \{a, a'\}$ A_t, A_{t+1}
- Transition/dynamics: $\Pr(S_{t+1} = s' \mid S_t = s, A_t = a)$ → `transition(state, action)`
- Feedback (reward, cost):

$$\Pr(R_t = r \mid S_t = s) \rightarrow \texttt{reward(state)}$$
$$\Pr(R_t = r \mid S_t = s, A_t = a) \rightarrow \texttt{reward(state, action)}$$
$$\Pr(R_t = r \mid S_t = s, A_t = a, S_{t+1} = s') \rightarrow \texttt{reward(state, action, next\_state)}$$

- **Note**: time is usually assumed to evolve in discrete steps

Markov Decision Processes | Building Blocks

- Return: the **sum** of rewards accumulated $G = R_{t+0} + R_{t+1} +, \ldots, + R_{t+H}$
- Discounting: future rewards are worth less than current ones

$$G = \gamma^0 R_{t+0} + \gamma^1 R_{t+1} +, \ldots, + \gamma^H R_{t+H} = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \qquad \gamma \in [0, 1]$$

- Policy : a prescription for actions $\pi(s) \qquad \pi : \mathcal{S} \times \mathcal{A} \longrightarrow [0, 1]$
- Value of a state: $V^{\pi}(s) = \mathbb{E}_{\pi}[G \mid S_t = s]$
- State-value function (Q-function): $Q^{\pi}(s, a) = \mathbb{E}_{\pi}[G \mid S_t = s, A_t = a]$

Markov Decision Processes | Prediction

- How to estimate the value of a policy:
  - **Monte Carlo** — average the values of states as the agent visits them.
    - For infinite samples (state visitation) converges to the true value
  - **Dynamic programming** — exploits the recursive relation between successive state values, $V^\pi(s) \quad V^\pi(s')$
    - Backup values from future states to the present state.

$$V^\pi(s) = \mathbb{E}_\pi[R_t + \gamma V^\pi(s') \mid S_t = s, A_t = a, S_{t+1} = s']$$

Markov Decision Processes | Assessment/Evaluation

- Given two policies, π_1, π_2
- Policy 1 is **better** than policy 2 iff $V^{\pi_1}(s) > V^{\pi_2}(s) \qquad \forall s \in \mathcal{S}$
- What is the **best/optimal** policy?
  - Has corresponding to the optimal value and Q functions,

$$V^*(s) = \max_{\pi \in \Pi} V^{\pi}(s) \qquad Q^*(s,a) = \max_{\pi \in \Pi} Q^{\pi}(s,a)$$

- Extract policy from value or Q-function $\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a)$

Additional RL Terminologies

- Episodic vs continuous
- On-policy vs off-policy
- Greedy — take the action providing maximum gain

Markov Decision Processes | Model-based vs Model-free

- **Model-based**: Collect data and learn reasonable approximations of T and R then apply value iteration to find the optimal value function, from which the policy to use is then extracted.

- **Model-free**: Directly estimate the value function or the optimal policy.

Planning & Learning

Markov Decision Processes | Planning (1)

- Given a full MDP model, find the optimal policy (or value function)
- Value iteration algorithm

$$
\begin{aligned}
&\text{Repeat} \\
&\quad \Delta \leftarrow 0 \\
&\quad \text{For each } s \in \mathcal{S}: \\
&\qquad v \leftarrow V(s) \\
&\qquad V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big] \\
&\qquad \Delta \leftarrow \max(\Delta, |v - V(s)|) \\
&\text{until } \Delta < \theta \text{ (a small positive number)}
\end{aligned}
$$

Markov Decision Processes | Planning (2)

- Q-learning: A model free way to find the optimal policy

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ϵ-greedy)
        Take action A, observe R, S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
        $S \leftarrow S'$
    until S is terminal

Markov Decision Processes | Exploration-Exploitation

- Actions determine which states the agent ends up.
  - Trying actions already known to lead to ‘good’ states — exploitation
  - Occasionally trying new actions to potentially end up in new states — exploration
- Exploration gathers new data while exploitation maximizes earnings
- When to stop exploring?

Markov Decision Processes | Reward Learning

- Inverse reinforcement learning
  - Recover an expert agent's reward function from traces of its behavior (state, action trajectories)
- Imitation learning
  - Assume that an expert agent is acting optimally, replicate its behavior.

Large scale

Markov Decision Processes | Large Models

- In large state and/or action spaces e.g. continuous domain, we use f**unction approximation** to represent the value or Q-functions.
  - Choice of function approximation method is crucial
  - RBFs, PVFs, Neural networks, etc
- Deep Q networks — use a neural network to approximate Q-function, plus a few more tricks (experience replay, etc)

Going further

Reinforcement Learning | Beyond MDPs

- POMDPs
  - Models for **uncertainty** in observation (states) and control (actions)
- SMDPs
  - Additional **abstraction** over states and actions
  - Hierarchy
- Bandits — essentially an MDP with a single state.

What next

RL Software | Getting Started

- Environment,
  - Examples: Gym https://gym.openai.com
- Algorithms
  - VI, PI, TD, Monte Carlo — common implementations found on github.

References

1. General go to books: **https://mitpress.mit.edu/books/reinforcement-learning**
2. Emma's RL course at Stanford: http://web.stanford.edu/class/cs234/syllabus.html