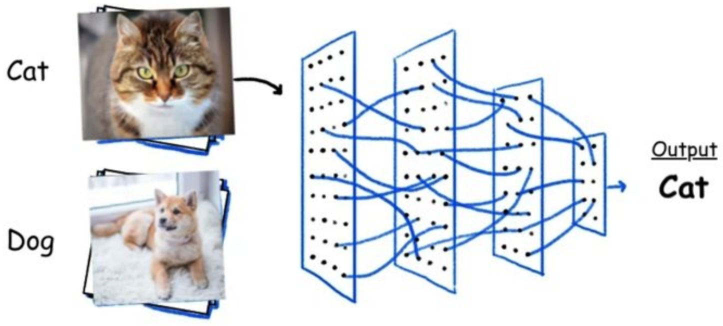
Mean-Variance Reinforcement Learning

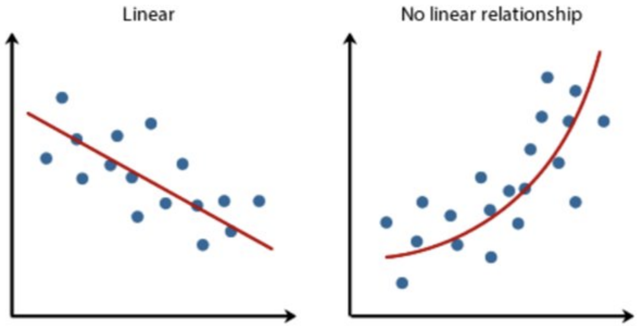
Presenter: Yudong Luo

Characteristics of Reinforcement Learning

- Machine Learning, e.g., supervised Learning



Classification



Regression

- $f_\theta(\vec{x}) \quad (\theta = \{\vec{\alpha}, b\})$, e.g, $\vec{\alpha}^\top \vec{x} + b$ Minimize $(\vec{\alpha}^\top \vec{x} + b - y)^2$
- Update parameters: take derivative, and gradient descent (Fit many $(\vec{x}, y)$ to update)
- Classification: apply softmax() function to output, turn value in $[0, 1]$
- Make it Non-linear: stack linear and non-linear layers
  - $f_{\theta_2}(\ \max\{0,\ f_{\theta_1}(\ \max\{0, f_{\theta_0}(\vec{x})\}\ )\ \}\ )$

Characteristics of Reinforcement Learning

What makes reinforcement learning different from other machine learning paradigms?

- No label (no supervisor), only a reward signal (given by the environment)
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives
- Feedback is delayed

Rewards

- Reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- Goal is to maximize cumulative reward $\mathbb{E}[R_1 + \gamma R_2 + \gamma^2 R_3 + ...]$ (discounted)

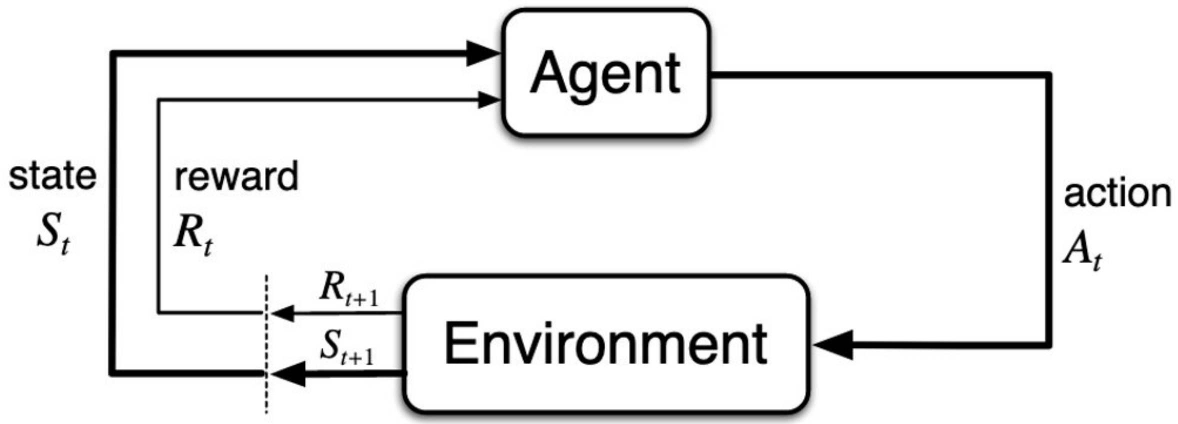
> Reward Hypothesis

All goals can be described by the maximization of expected cummulative reward

- Example: make a robot walk
  - positive reward for forward motion
  - negative reward for falling over

Design reasonable reward for your task

Markov Decision Process (MDP)



A Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}, \mathcal{A}$ finite set of states, actions
- $\mathcal{P}$ is state transition probability $\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$ $(P(s'|s, a))$
- $\mathcal{R}$ is reward function, $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$ $(r(s, a))$
- γ is a discount factor $\gamma \in [0, 1]$

Major Components of RL

Policy (usually denote by π):

- A policy is the agent's behavior
- It is a map from state to action $\pi : \mathcal{S} \to \mathcal{A}$
  - Maze. State: location. Action: {Forward, Backward, Left, Right}
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$ (good for exploration)
  - Get an action, sample from $\pi(\cdot|s)$

Major Components of RL

Value function

- Value function is a prediction of expected future return
- Evaluate the goodness/badness of states
- Therefore determine which action to choose

$$V^\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s]$$

$$Q^\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s, A_t = a]$$

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)}[Q^\pi(s, a)]$$

Bellman Equation

- Bellman Expectation Equation

$$V^{\pi}(s) = \mathbb{E}_{\pi}[R_{t+1} + \gamma V^{\pi}(S_{t+1})|S_t = s]$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[R_{t+1} + \gamma Q^{\pi}(S_{t+1}, A_{t+1})|S_t = s, A_t = a]$$

Consider a state-action pair (s, a), the reward $r(s, a)$ and all possible next $\{s', a'\}$

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

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

We are evaluating a policy π

Bellman Equation

- Bellman Optimality Equation

The optimal value function is the maximum value over all policies

$$V^*(s) = \max_\pi V^\pi(s) \quad Q^*(s,a) = \max_\pi Q^\pi(s,a)$$

$$V^*(s) = \max_a Q^*(s,a)$$

- Optimal Policy: every state achieves the optimal value (cannot do even better)

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

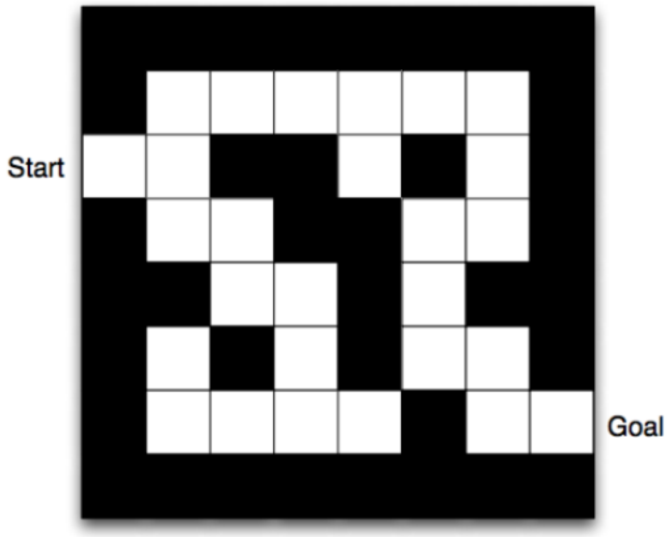
$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \max_{a' \in \mathcal{A}} Q^*(s',a')$$

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

$\pi^*(a^*|s) = 1.0$

Find Optimal Policy: value based method

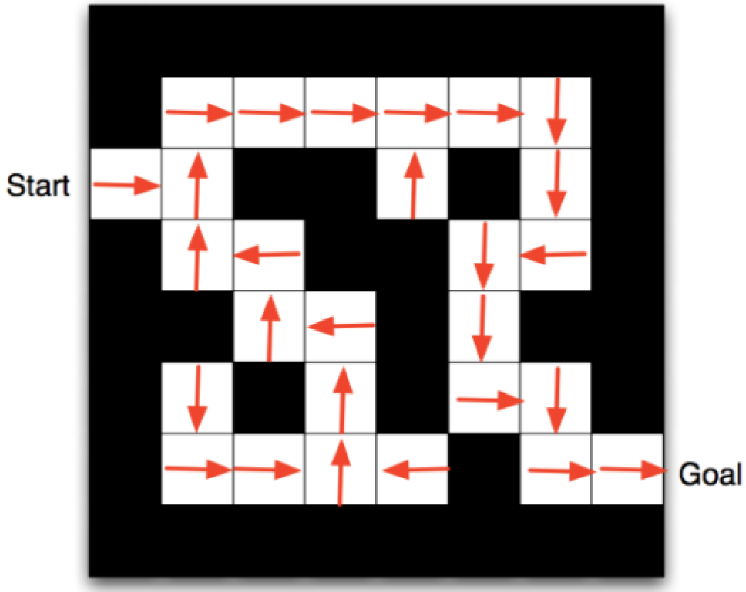
- There is always a deterministic optimal policy for any MDP
- If we know $Q^*(s, a)$, we immediately have the optimal policy

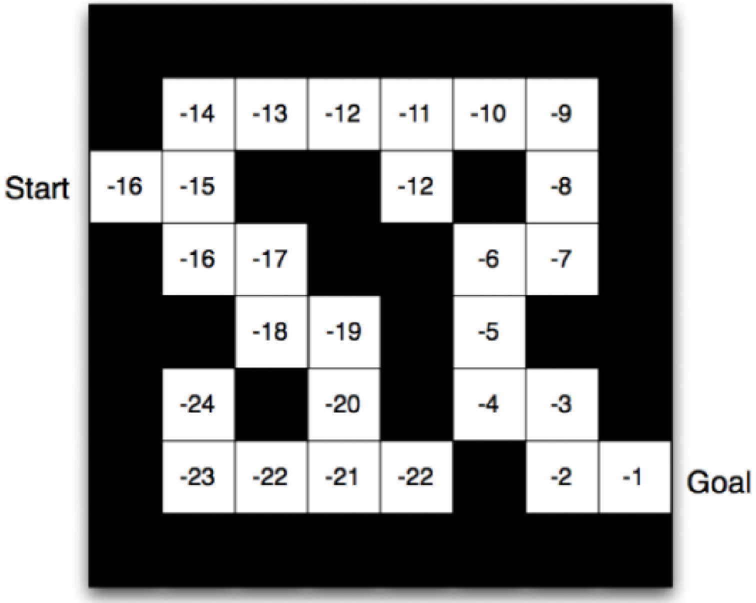


Rewards: -1 per step. Undiscounted ($\gamma = 1$).

State: current location, Actions: four directions

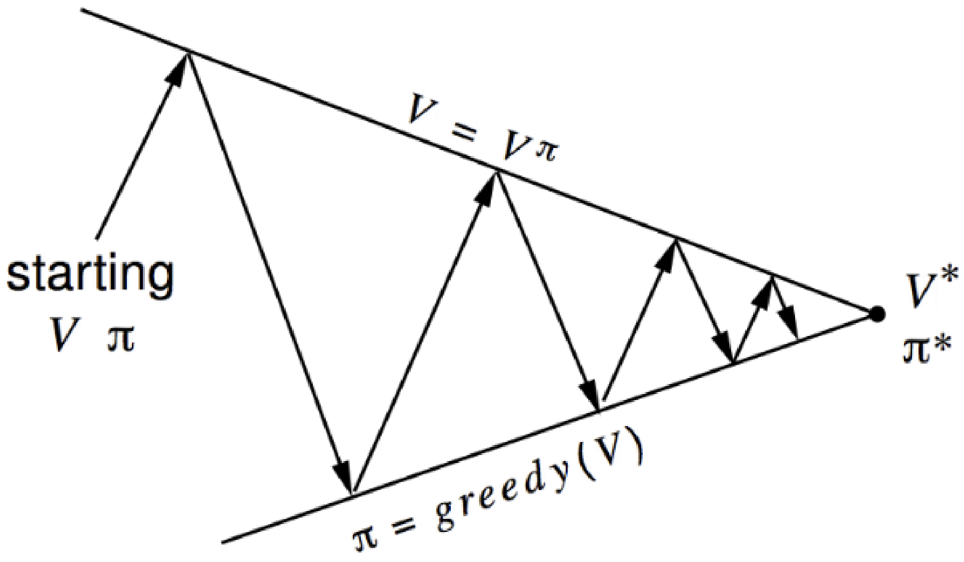
Find Optimal Policy: value based method





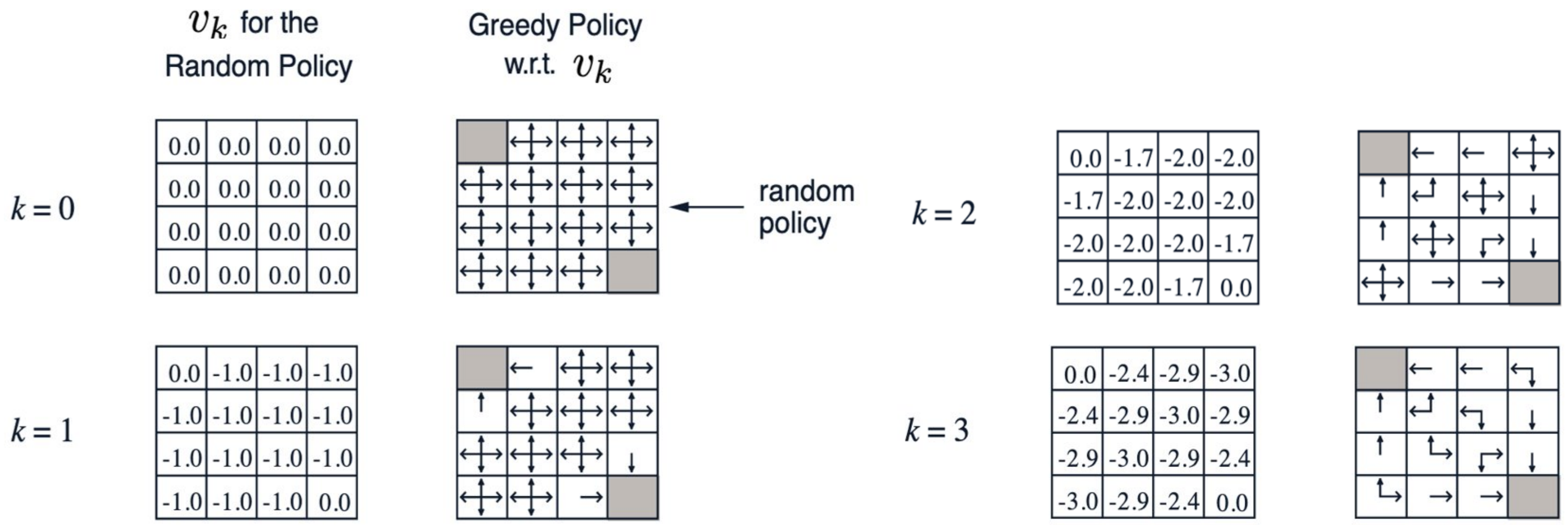
Find Optimal Policy: value based method

- There is always a deterministic optimal policy for any MDP
- If we know $Q^*(s, a)$, we immediately have the optimal policy
- How to find Q^*?
  - Policy Iteration / Value Iteration



Find Optimal Policy: value based method

Policy Iteration / Value Iteration



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

$$\pi(s) = \arg\max_{a} Q(s,a)$$

Find Optimal Policy: value based method

Q-Learning

1. Initialize Q table, learning rate α
2. At state s, choose action $a = \arg\max_a Q(s,a)$ (may add noise for exploration)
3. Observe reward r and next state s'
4. $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big]$

Time difference (TD) error: $r + \gamma \max_{a'} Q(s',a') - Q(s,a)$

Find Optimal Policy: value based method

Deep Q-Learning

- When $\mathcal{S}$ and $\mathcal{A}$ spaces are large, it is impossible to maintain a Q table.
- Use function approximation to represent $Q(s, a)$, e.g., a deep neural network with parameter θ.
- Update θ to minimize the TD error.
    - Take derivative for θ and gradient descent

Find Optimal Policy: policy gradient method

- When action space is continuous, it is unable to $\arg\max_a Q(s,a)$
  - PG still works when action space is discrete
- Goal is to find optimal policy, can we directly parametrize the policy and optimize?

$$\pi_\theta(a|s) = \mathbb{P}[a|s,\theta]$$

- Start from some initial state, execute policy up to some time horizon T.

$$\tau = (s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T, s_T) \quad R(\tau) = \sum_t \gamma^t r_{t+1}$$

- Maximize $\mathbb{E}_\tau[R(\tau)]$
- Need to take derivative for θ and do gradient ascent

Find Optimal Policy: policy gradient method

Policy Gradient

$$\nabla_\theta \mathbb{E}_\tau[R(\tau)] = \mathbb{E}_\tau[\nabla_\theta \log P(\tau|\theta) R(\tau)] \quad \text{apply} \quad \nabla_\theta \log(x) = \frac{1}{x}\nabla_\theta x$$

$$P(\tau|\theta) = \mu(s_0) \cdot \prod_{t=0}^{T-1} [\pi_\theta(a_t|s_t) \cdot P(s_{t+1}|s_t, a_t)]$$

$$\log P(\tau|\theta) = \log \mu(s_0) + \sum_{t=0}^{T-1} \log[\pi_\theta(a_t|s_t) + P(s_{t+1}|s_t, a_t)]$$

$$\nabla_\theta \log P(\tau|\theta) = \nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)$$

$$\nabla_\theta \mathbb{E}_\tau[R(\tau)] = \mathbb{E}_\tau[R(\tau) \nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)]$$

Find Optimal Policy: policy gradient method

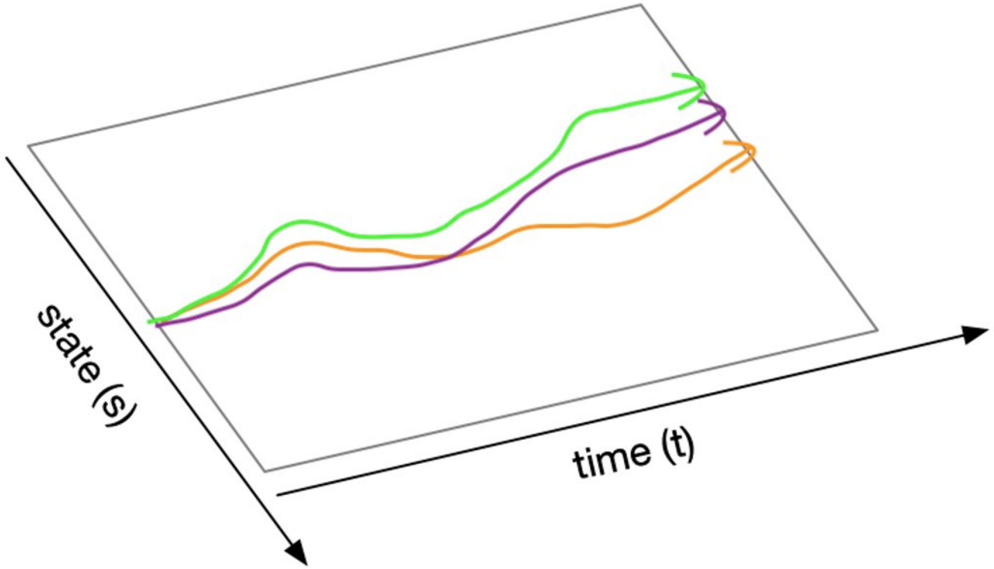
Vanila Policy Gradient

- Initialize a policy π_θ, learning rate α
- sample $\tau = (s_0, a_0, r_1, s_1, \dots)$ by executing π_θ
- $\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}_\tau [R(\tau)]$

This gradient has high variance

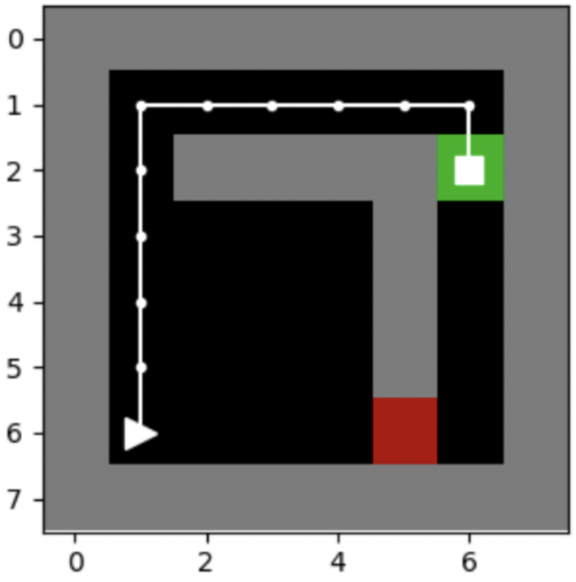
Risk-neutral vs Risk-averse RL

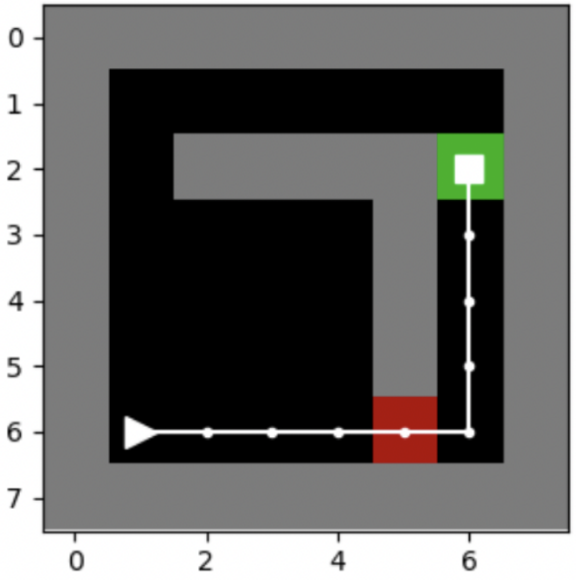
The cumulative return, i.e., $R_1 + \gamma R_2 + \gamma^2 R_3 \ldots$, is a random variable.



- Risk-neutral RL: only maximize the expectation (mean)
- Risk-averse RL: maximize mean while minimize some risk term, e.g. variance

Mean-Variance RL: example





- -1 per step, except
- red state: $\{-10, -1, 8\}$ with prob $\{0.4, 0.2, 0.4\}$.
    - $(-10) \times 0.4 + (-1) \times 0.2 + 8 \times 0.4 = -1$

Mean-Variance RL

$G = R_1 + \gamma R_2 + \gamma^2 R_3 \ldots$

$$\max_{\pi} \mathbb{E}[G] - \lambda \mathbb{V}[G]$$

$\mathbb{E}[G]$: time consistent, Bellman equation, TD learning

$\mathbb{V}[G]$: time inconsistent, minimizing variance at each step is not minimizing variance of the total return

Unable to use valued based method, consider policy gradient?

Mean-Variance RL Policy Gradient

(Switch among π, θ, π_θ)

$\mathbb{V}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$

Define $J(\theta) = \mathbb{E}_\pi[G]$, $M(\theta) = \mathbb{E}_\pi\left[(\sum_{t=0}^{\infty} \gamma^t R_{t+1})^2\right]$

$$J_\lambda(\theta) = J(\theta) - \lambda\big(M(\theta) - J^2(\theta)\big)$$

$$\nabla_\theta J_\lambda(\theta) = \nabla_\theta J(\theta) - \lambda\big(\nabla_\theta M(\theta) - 2J(\theta)\nabla_\theta J(\theta)\big)$$

- $\nabla_\theta J(\theta)$: $\mathbb{E}_\tau[R(\tau)\omega(\theta)]$ $\qquad \omega(\theta) = \nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)$
- $\nabla_\theta M(\theta)$: $\mathbb{E}_\tau[R^2(\tau)\omega(\theta)]$
- $J(\theta)\nabla_\theta J(\theta)$: need double sampling

Mean-Variance RL Policy Gradient

Tamar et al. (2012): two time scale algorithm

- A faster learning rate for estimating $\mathbb{E}[G]$ and $\mathbb{V}[G]$.
- A slower learning rate for updating θ.
  - $J(\theta)\nabla J(\theta)$: $J(\theta)$ uses $\mathbb{E}[G]$, only compute $\nabla_\theta J(\theta)$

Mean-Variance RL Policy Gradient

Xie et al. (2018): introduce Fenchel duality: $x^2 = \max_y(2xy - y^2)$.

Define $J(\theta) = \mathbb{E}_\pi[G]$, $M(\theta) = \mathbb{E}_\pi\big[(\sum_{t=0}^\infty \gamma^t R_{t+1})^2\big]$

$$\max_\theta J_\lambda(\theta) = J(\theta) - \lambda\big(M(\theta) - J^2(\theta)\big)$$

$$\begin{aligned} F_\lambda(\theta) &= (J(\theta) + \frac{1}{2\lambda})^2 - M(\theta) = \frac{J_\lambda(\theta)}{\lambda} + \frac{1}{4\lambda^2} \\ &= \max_y \left(2y(J(\theta) + \frac{1}{2\lambda}) - y^2\right) - M(\theta) \end{aligned}$$

No $J(\theta)\nabla_\theta J(\theta)$ term!

Upper bound of total return variance

Regard per-step reward as a random variable R (Bisi et al., 2020)

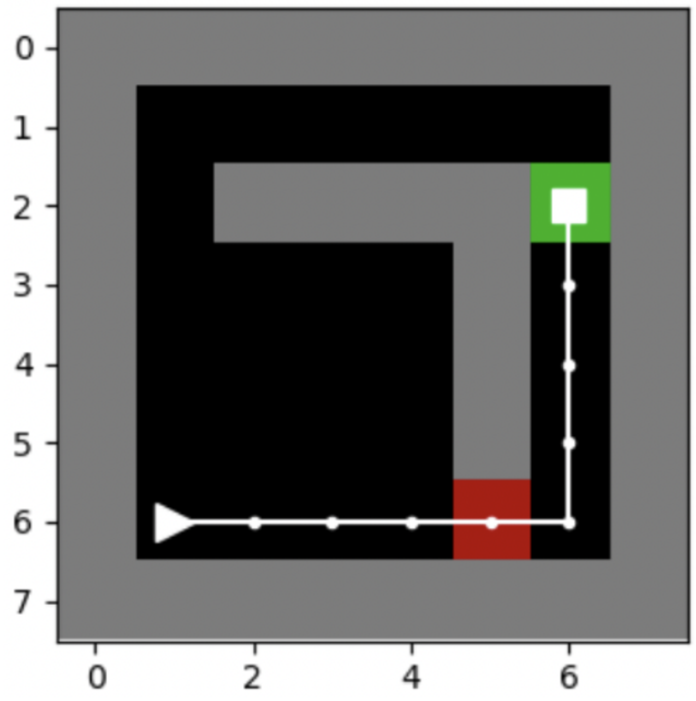
$$\mathbb{V}[G] \leq \frac{\mathbb{V}[R]}{(1-\gamma)^2}$$

$\gamma = 0.99$

-1 per step, 10 for goal, red state $\{-10, -1, 8\}$

$\{-1, \ldots, -10, \ldots, 10\}, p = 0.4, g \approx -7.23$
$\{-1, \ldots, -1, \ldots, 10\}, p = 0.2, g \approx 1.50$
$\{-1, \ldots, 8, \ldots, 10\}, p = 0.4, g \approx 10.23$

$\mathbb{V}[G] \approx 61.01, \mathbb{V}[R]/(1-0.99)^2 \approx 19.15/(0.01)^2$



Upper bound of total return variance

Benefit?

$$\hat{J}_\lambda(\pi) = \mathbb{E}[R] - \lambda\mathbb{V}[R] = \mathbb{E}[R] - \lambda(\mathbb{E}[R^2] - (\mathbb{E}[R])^2)$$

Fenchel duality (Zhang et al., 2021)

$$\hat{J}_\lambda(\pi) = \mathbb{E}[R] - \lambda\mathbb{V}[R] = \mathbb{E}[R] - \lambda\mathbb{E}[R^2] + \lambda \max_y(2\mathbb{E}[R]y - y^2)$$

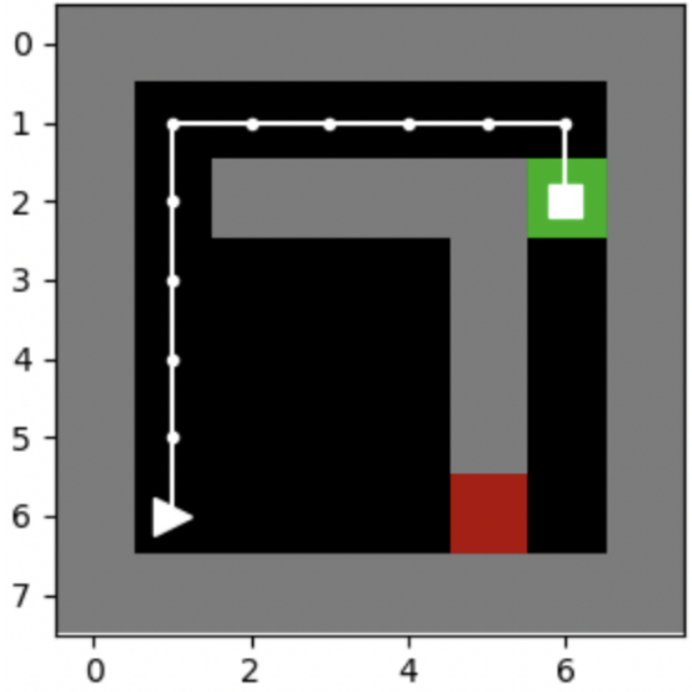
- close form solution for y (quadratic): $y^* = \mathbb{E}_{s,a\sim d^\pi}[R(s,a)]$
- New reward $\hat{r}(s,a) = r(s,a) - \lambda r(s,a)^2 + 2\lambda r(s,a)y$ to optimize for π.

Why? (occupancy measure)

$$\mathbb{E}_\pi[G] = \mathbb{E}_{s,a\sim d^\pi(s,a)}[R(s,a)]$$

Limitations

- Directly optimize mean-variance: double sampling issue
- Use $\mathbb{V}[R]$:
  - Different term, e.g., $\mathbb{V}[R] \neq 0$ while $\mathbb{V}[G] = 0$
  - Sensitive to reward magnitude



References

Tamar, A., Di Castro, D., and Manor, S. Policy Gradients with Variance related Risk criteria. International Conference on Machine Learning, 2012.

Xie, T., Liu, B., Xu, Y., Ghavamzadeh, M., Chow, Y., Lyu, D., and Yoon, D. A block coordinate ascent algorithm for mean-variance optimization. Advances in Neural Information Processing Systems, 31, 2018.

Bisi, L., Sabbioni, L., Vittori, E., Papini, M., and Restelli, M. Risk-averse trust region optimization for reward-volatility reduction. International Joint Conference on Artificial Intelligence, 2020.

Zhang, S., Liu, B., and Whiteson, S. Mean-variance pol- icy iteration for risk-averse reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, 2021.