Reinforcement Learning for Language Model Training

Polina Tsvilodub

Reinforcement Learning

RL4 LMT

LMs: Recap

Transformers,Training & Inference

- $LM : X \mapsto \Delta(S)$
- transformers use self-attention to offer and retrieve relevant information
  - stacked transformer blocks and multi-head attention increase capacity
- LMs are trained to predict the next word using cross-entropy loss (via teacher-forcing)
- decoding schemes are used for inference given a trained LM
  - different stochastic sampling regimes
- SOTA models exhibit ‘in-context learning’
- advanced prompting techniques might improve LLMs’ generalization performance

Making LLMs useful

Adaptation

- training a task-specific head on top of a model
  - e.g., span prediction layer on top of BERT with frozen BERT
  - on a dataset of ground truth input-output pairs for a particular task
- fine-tuning the model
  - further training part or entire model for a shorter time
  - on a dataset of ground truth input-output pairs for a particular task
- practical problem
  - training with standard supervision is impractical (data collection)
  - and inefficient (restricting “ground truth” to finite set of answers for open-ended tasks)
- **RL is the solution:** learn to achieve goal based on feedback from environment rather than direct demonstration of correct behaviour

source

Reinforcement learning

Flavors of machine learning

- Unsupervised learning
  - e.g., clustering
- discover patterns in unlabeled data
  - ‘given my inductive bias, what is the likely structure of the data?’

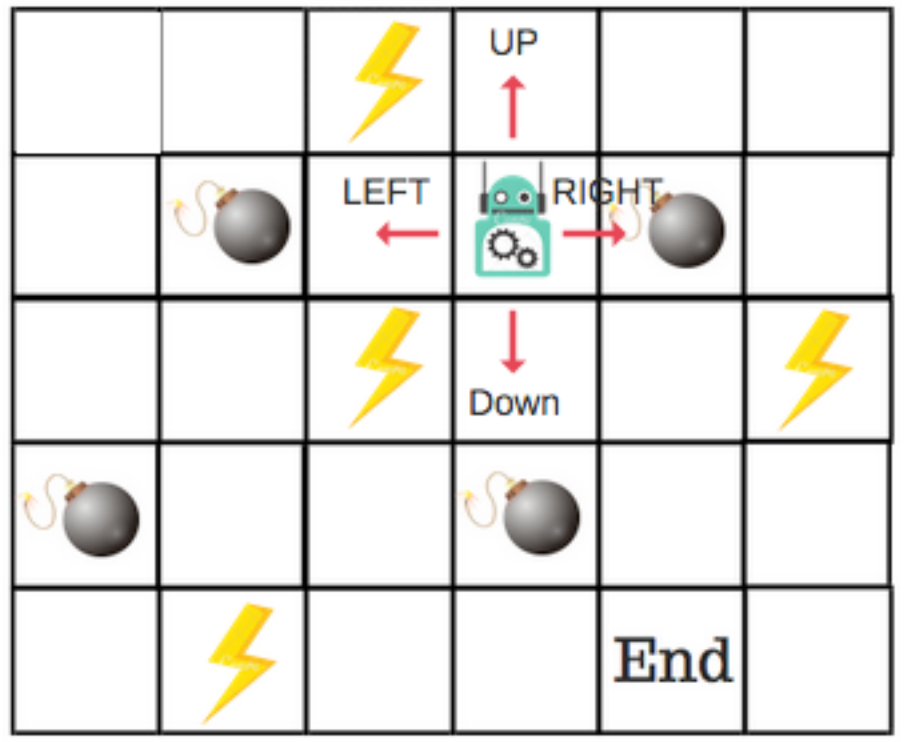
- Supervised learning
  - also self-supervised learning
  - aka behavioural cloning
- learn to output Y, given X, from labeled data
  - ‘do as I show you’
- learning from demonstration

- Reinforcement learning
  - trial-and-error learning
- learning from interaction / experience
  - ‘how do I optimally behave in order to maximize reward?’
  - or, ‘how do i optimally achieve my goal?’
    - most natural way of learning?
    - tightly connected to the way organisms behave (“pleasure maximizers”)

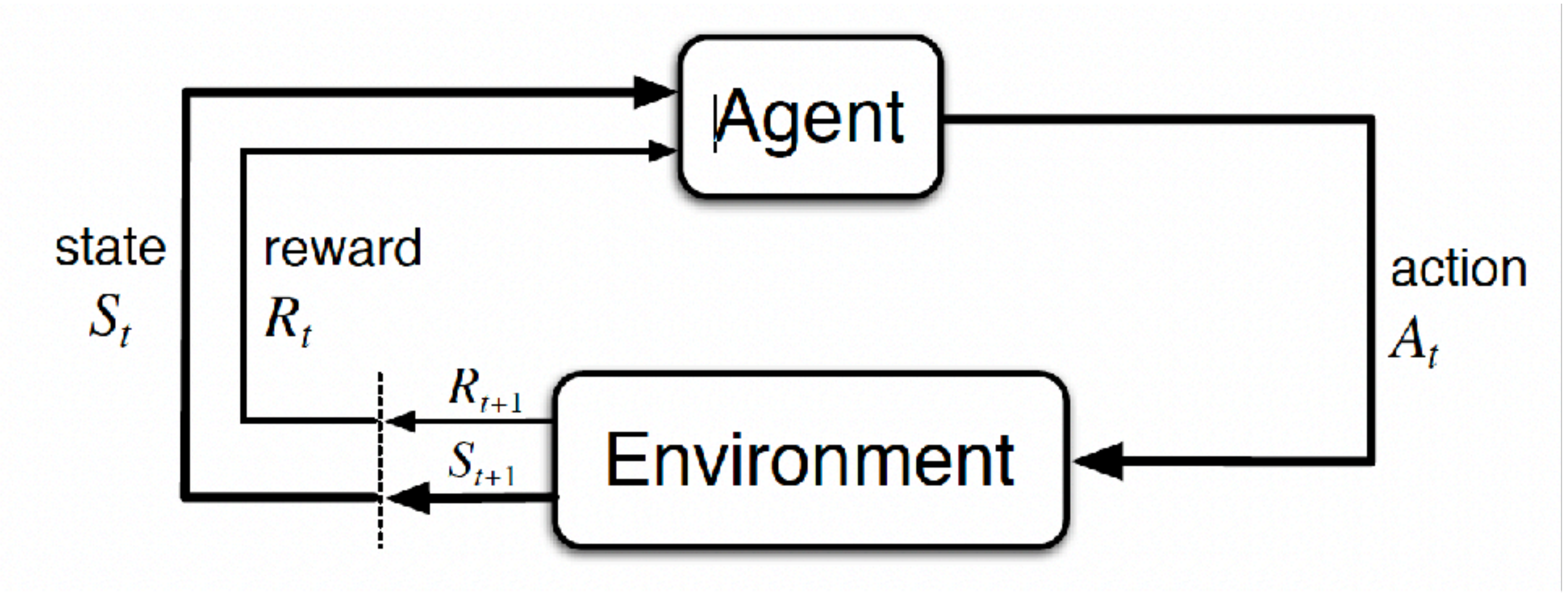
Christian (2020), Bishop (2009)

Reinforcement Learning: Overview
Introduction

- **Reinforcement Learning:** Computational formalisation of goal-directed learning and decision making under uncertainty
- **Goal:** Maximize rewards (by learning optimal behavior)
- **Basic building blocks:**
  - Agent
  - States
  - Actions
  - Transition function P
  - Reward
  - Policy



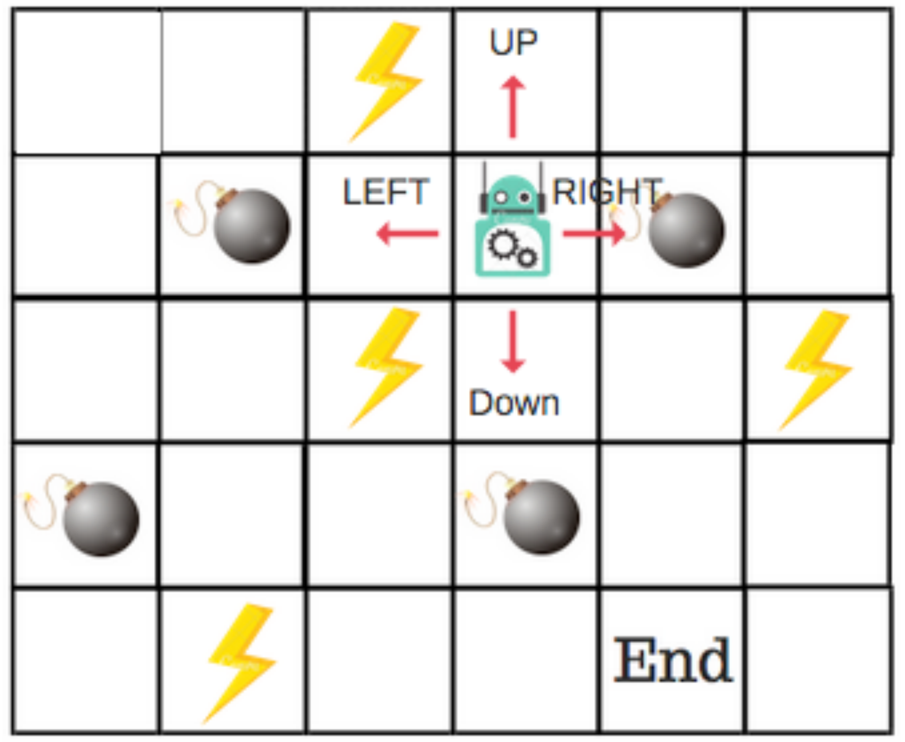
Associative RL



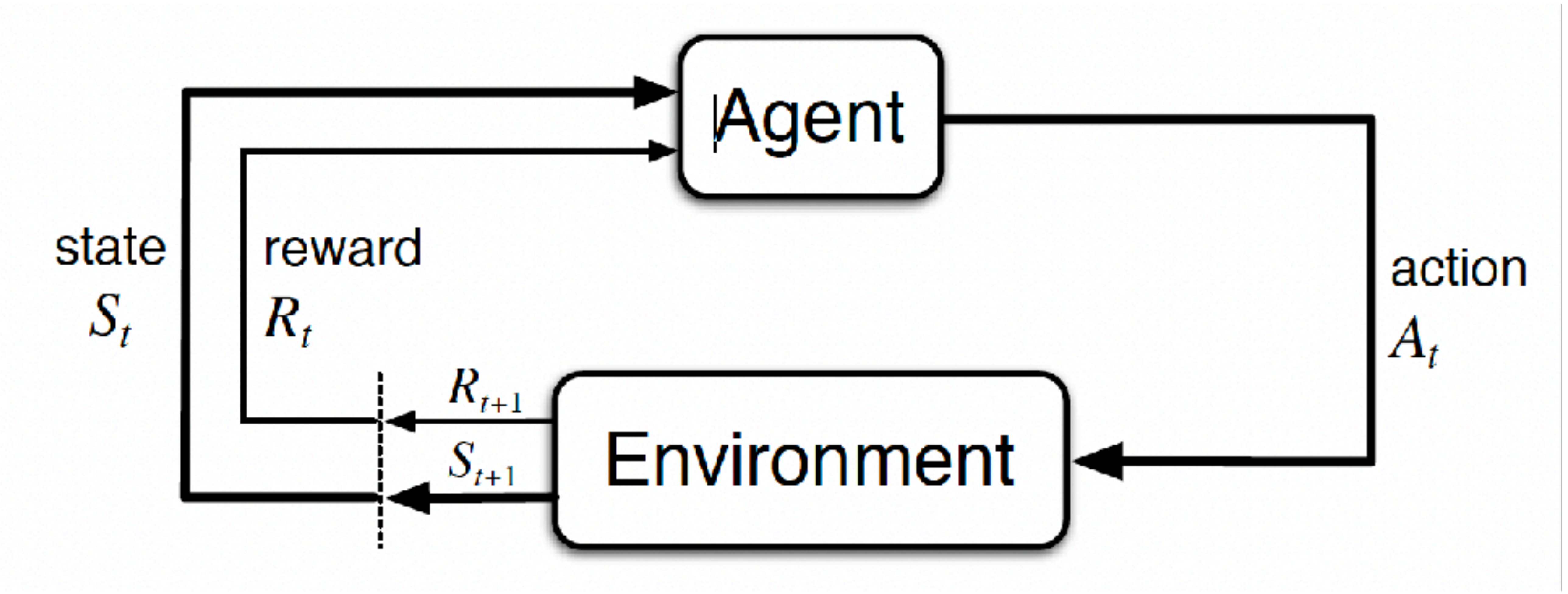
Sutton & Barto (2018, p. 48, Fig 3.1), image source

Reinforcement Learning: Overview

- **Reinforcement Learning:** Computational formalisation of goal-directed learning and decision making under uncertainty
- **Goal:** Maximize rewards (by learning optimal behavior)
- **Basic building blocks:**
  - Agent
  - States: $S_t \in \mathcal{S}$ for $t = 0, 1, 2, 3, \ldots$
  - Actions: $A_t \in \mathcal{A}(s)$
  - Transition function: $P(s' \mid s, a)$
  - Reward: $R_{t+1} \in \mathcal{R}$
  - Policy: $\pi(S_t) = P(A_t \mid S_t)$



Associative RL

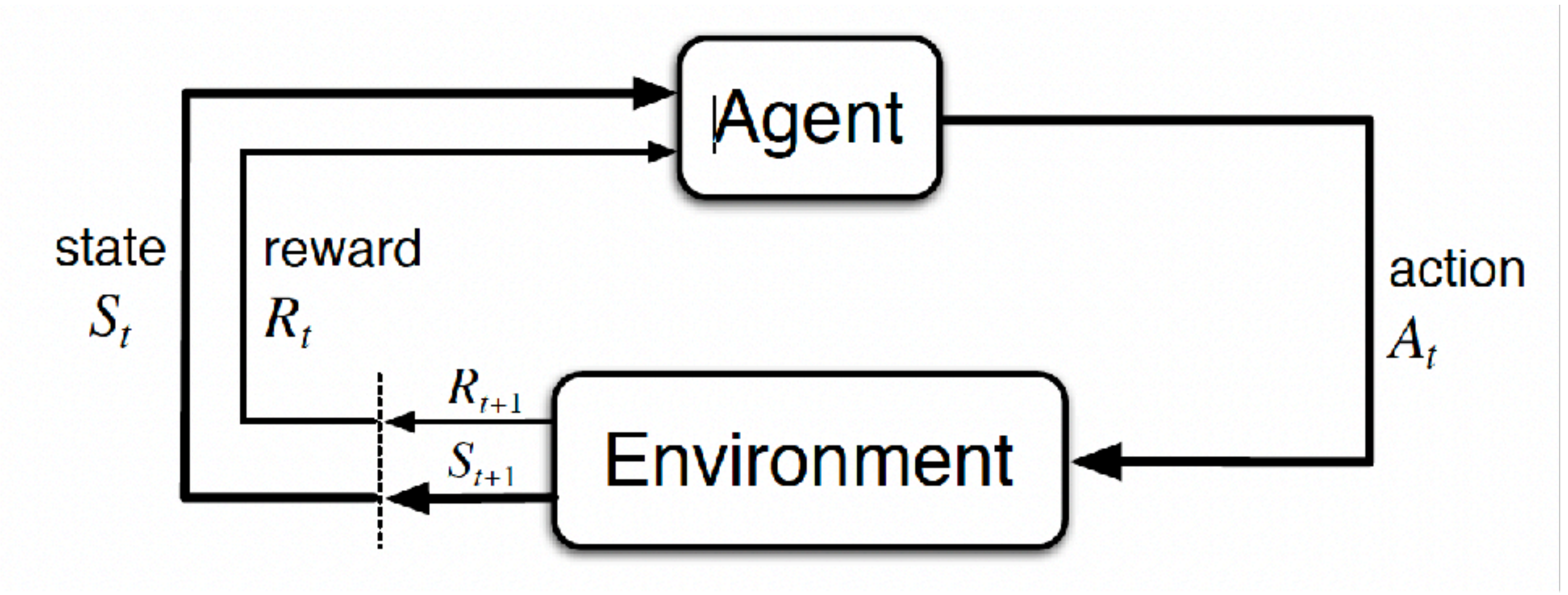


Sutton & Barto (2018, p. 48, Fig 3.1), image source

Markov Decision Processes
Formal definition

Finite MDPs: (S, A, T, R)

1. $S_t \in S$ for $t = 0, 1, 2, 3, \dots$
2. $A_t \in A(s)$
3. $R_{t+1} \in R$
4. $T(s' \mid s, a) = \sum_{r' \in R} P(s', r' \mid s, a)$
5. Horizon H, discount factor $0 \leq \gamma \leq 1$



Expected rewards for state s and action a: $r(s, a) = \mathbb{E}(R_t \mid S_{t-1} = s, A_{t-1} = a) = \sum_{r \in R} r \sum_{s' \in S} P(s', r \mid s, a)$

Markov property: S_{t-1}, A_{t-1} include all information about past agent-environment interactions that are relevant for S_t, R_t

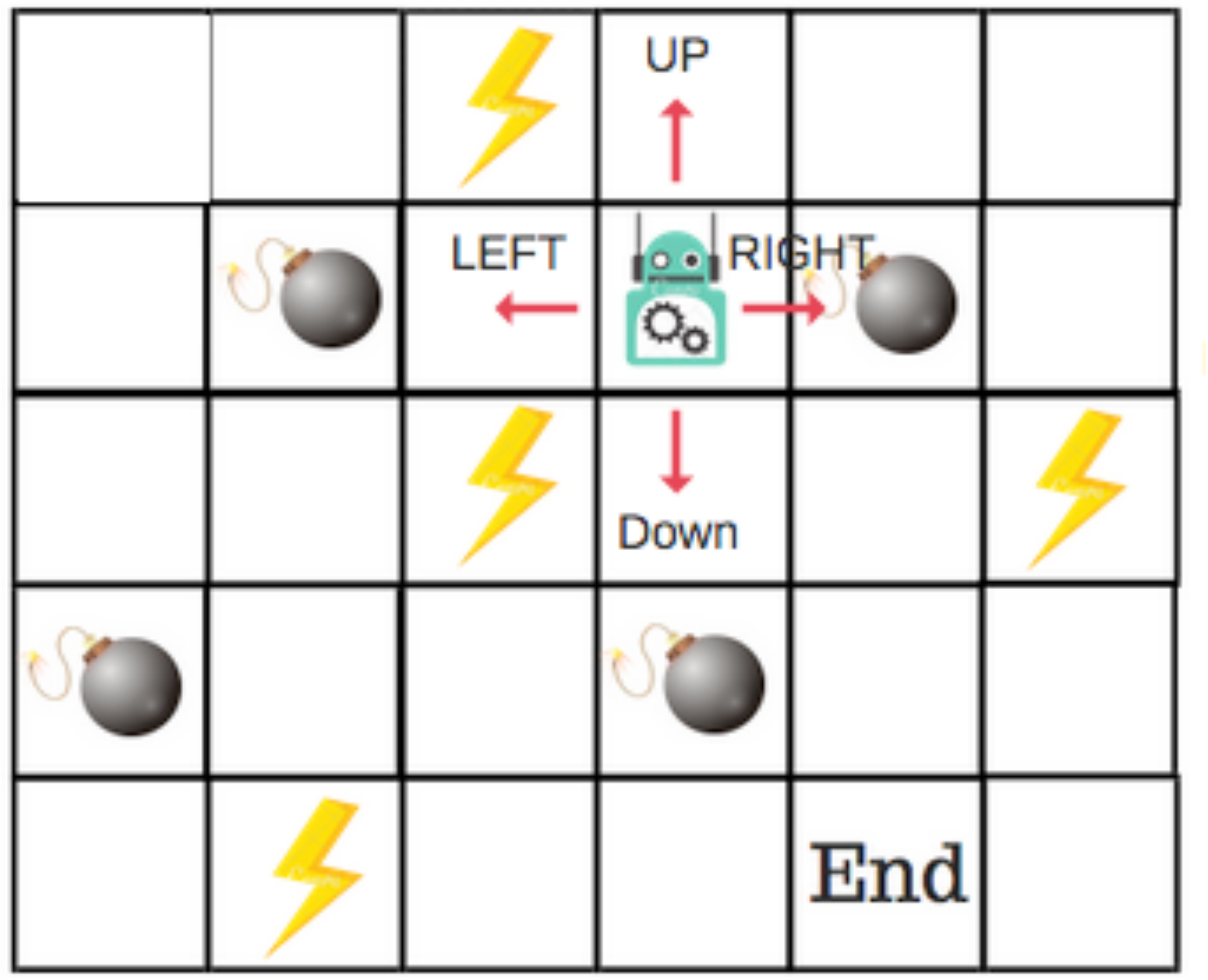
Sutton & Barto (2018, p. 48, Fig 3.1)

Markov Decision Processes

Formal definition

Finite MDPs: (S, A, T, R)

1. $S_t \in S$ for t $= 0, 1, 2, 3, \ldots$
2. $A_t \in A(s)$
3. $R_{t+1} \in R$
4. $T(s' \mid s, a) = \sum_{r' \in R} P(s', r' \mid s, a)$



Goal: maximize **returns** until goal achieved

$G_t = R_{t+1} + R_{t_2} + \ldots + R_T$

Formally: maximize expected **discounted** rewards over **episode**

$$G_t = R_{t+1} + \gamma R_{t_2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

Sutton & Barto (2018, p. 48, Fig 3.1)

Markov Decision Processes
Formal definition

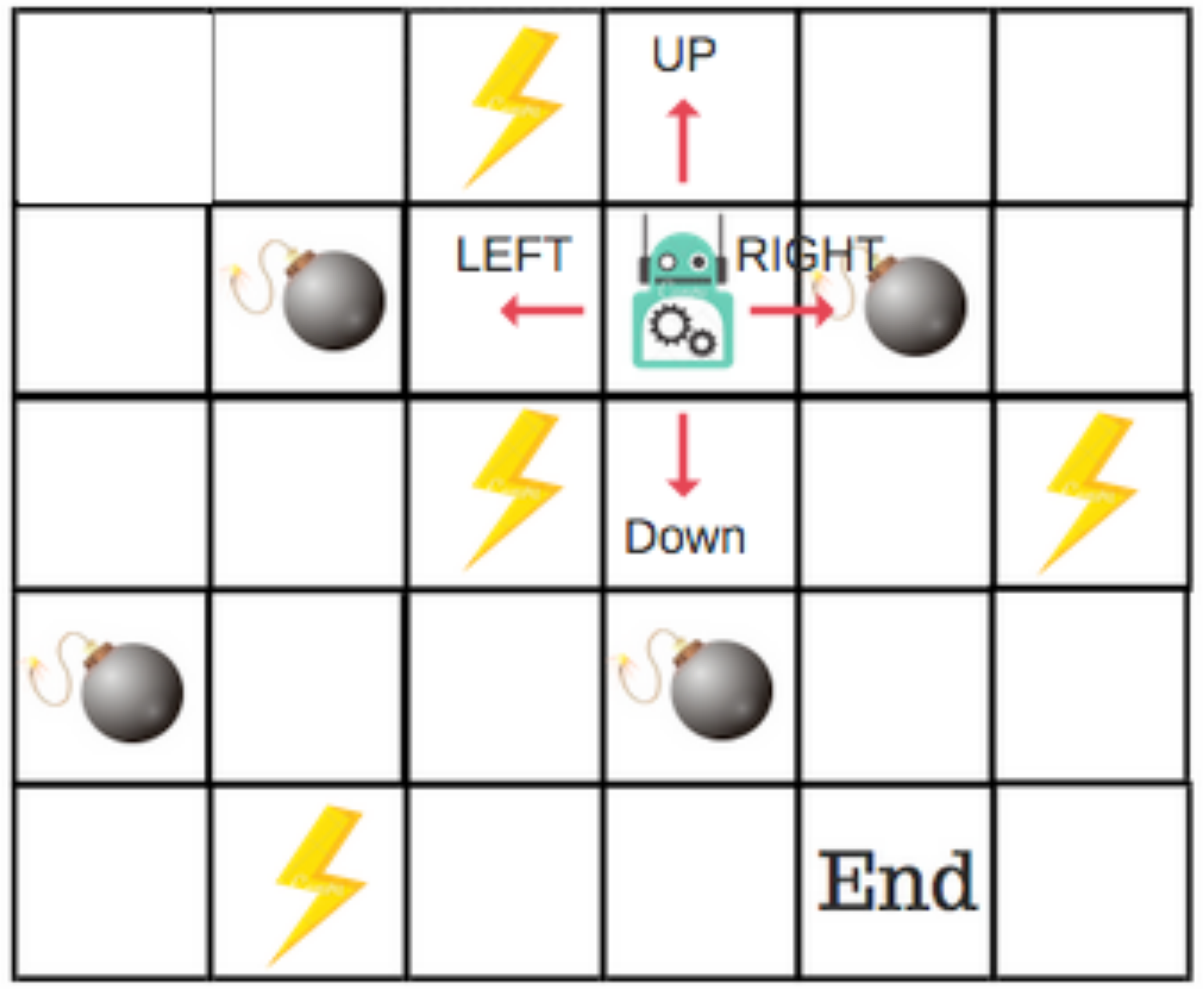
Finite MDPs: (S, A, T, R)

1. $S_t \in S$ for $t = 0, 1, 2, 3, \ldots$
2. $A_t \in A(s)$
3. $R_{t+1} \in R$
4. $T(s' \mid s, a) = \sum_{r' \in R} P(s', r' \mid s, a)$

Goal: maximize discounted **returns**

$$G_t = R_{t+1} + \gamma R_{t_2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

$$= R_{t+1} + \gamma G_{t+1}$$



- We can identify optimal way to behave if we know what good particular states and/or actions are:

Optimal **state-value function:**

$$V^*_\pi(s) = \max_\pi \mathbb{E}[G_t \mid S_t = s] = \max_\pi \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \max_a \sum_{s',r} P(s', r \mid s, a)[r + \gamma G_{t+1} \mid S_t = s] \text{ for all } s$$

- Optimization problem: (computationally) find **optimal policy** $\pi^*(S_t) = P(A_t \mid S_t)$

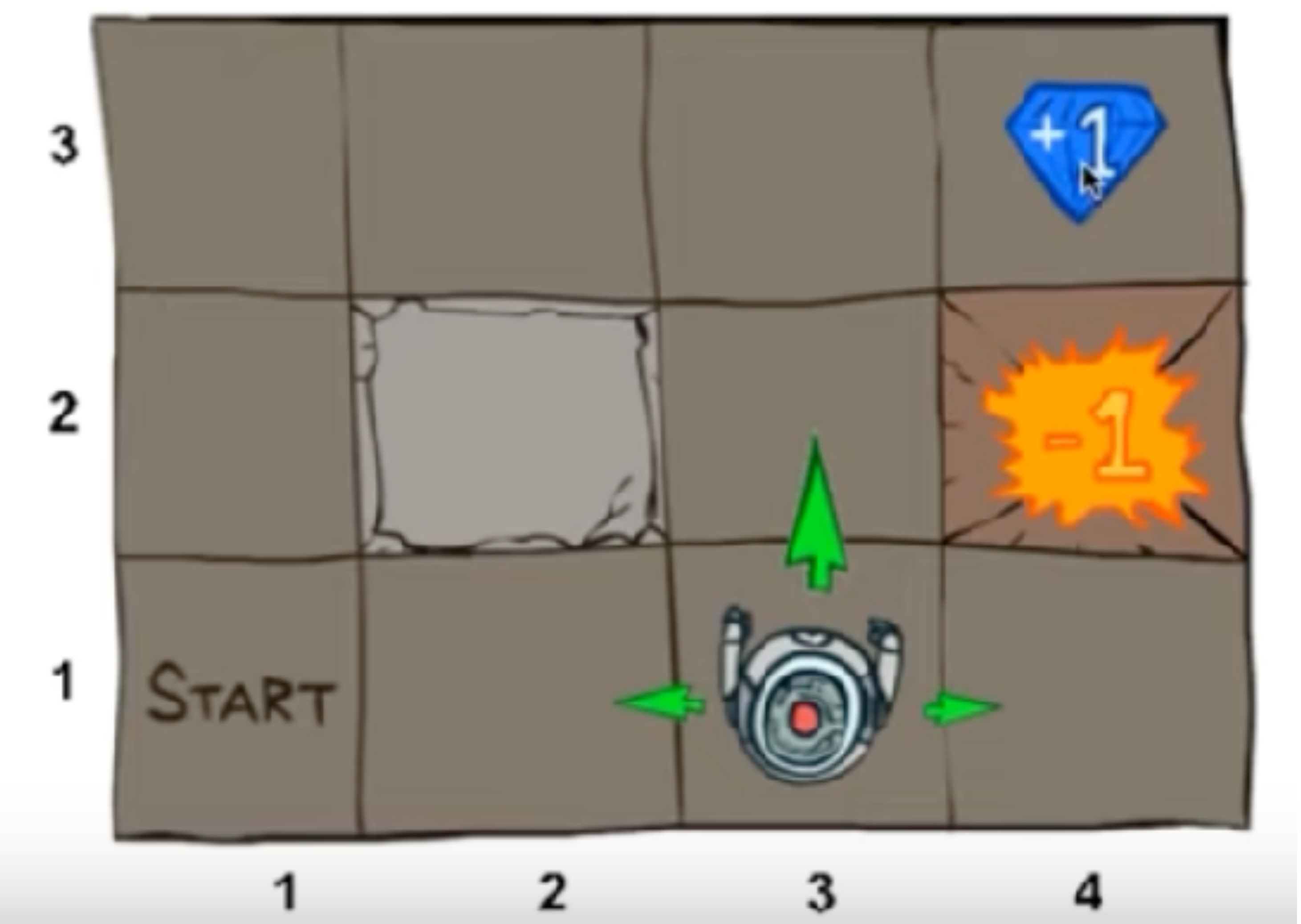
Sutton & Barto (2018, p. 48, Fig 3.1)

Markov Decision Processes

Formal definition

Finite MDPs: (S, A, T, R)

1. $S_t \in S$ for $t = 0, 1, 2, 3, \ldots$
2. $A_t \in A(s)$
3. $R_{t+1} \in R$
4. $T(s' | s, a) = \sum_{r' \in R} P(s', r' | s, a)$



Optimal state-value function:

$$V_\pi^*(s) = \max_\pi \mathbb{E}[G_t | S_t = s] = \max_\pi \mathbb{E}[\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k | S_t = s]$$

deterministic optimal policy $\gamma = 1$	deterministic optimal policy $\gamma = 0.9$
$V^*(4,3) = 1$	$V^*(4,3) = 1$
$V^*(3,3) = 1$	$V^*(3,3) = 0.9$
$V^*(2,3) = 1$	$V^*(2,3) = 0.81$
$V^*(1,1) = 1$	$V^*(1,1) = 0.9^5 = 0.59$
$V^*(4,2) = -1$	$V^*(4,2) = -1$

source

Markov Decision Processes

Formal definition

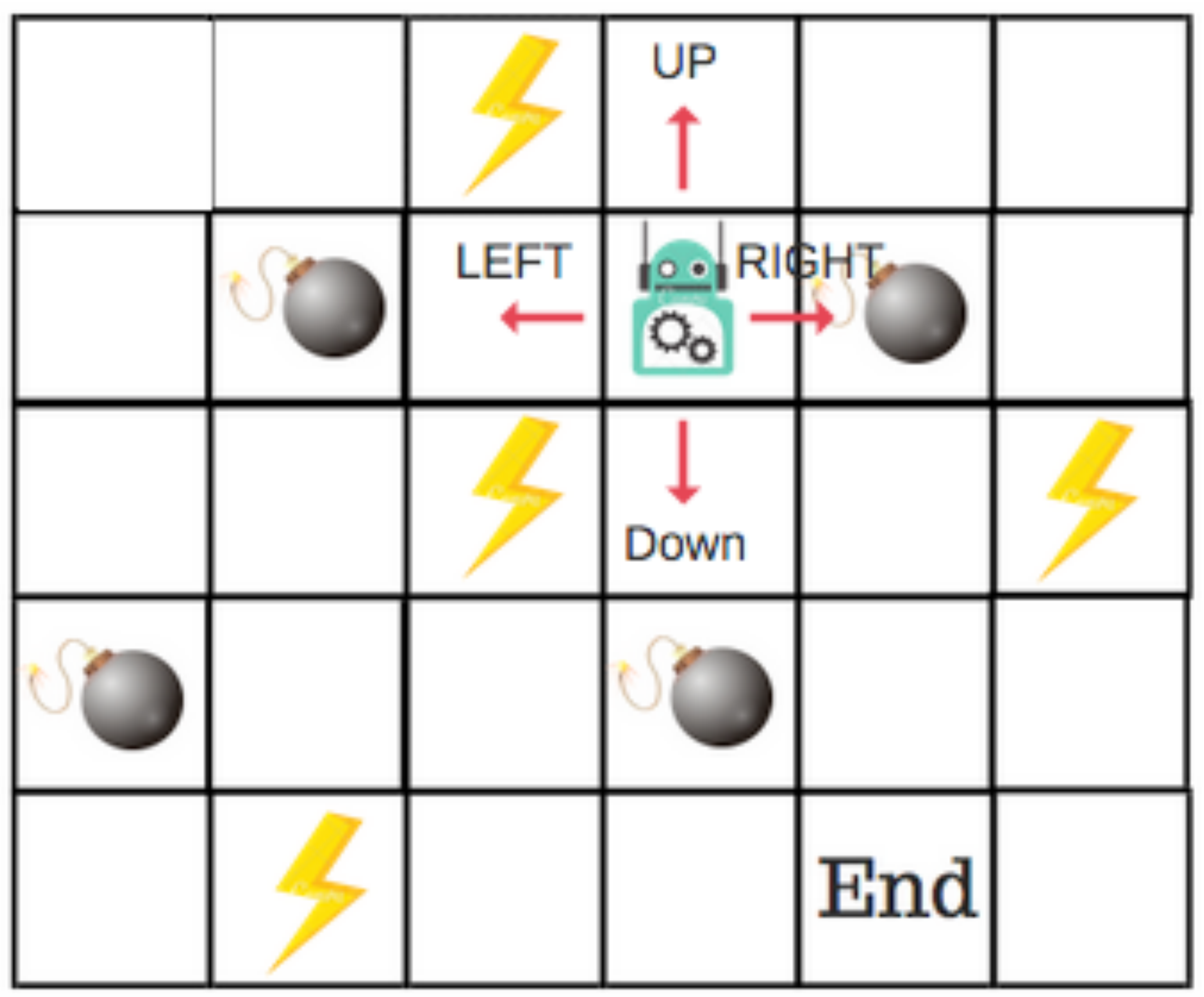
Finite MDPs: (S, A, T, R)

1. $S_t \in S$ for $t = 0, 1, 2, 3, \ldots$
2. $A_t \in A(s)$
3. $R_{t+1} \in R$
4. $T(s' \mid s, a) = \sum_{r' \in R} P(s', r' \mid s, a)$

Goal: maximize discounted **returns**

$$G_t = R_{t+1} + \gamma R_{t_2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

$$= R_{t+1} + \gamma G_{t+1}$$



- We can identify optimal way to behave if we know what good particular states and/or actions are:

Optimal **action-value function:**

$$Q^*_\pi(s, a) = \max_\pi \mathbb{E}[G_t \mid S_t = s, A_t = a] = \max_\pi \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a]$$

$$= \sum_{s', r} P(s', r \mid s, a)[r + \gamma \max_{a'} Q^*(s', a') \mid S_t = s, A_t = a] \text{ for all } s, a$$

Sutton & Barto (2018, p. 48, Fig 3.1)

Markov Decision Processes

Formal definition

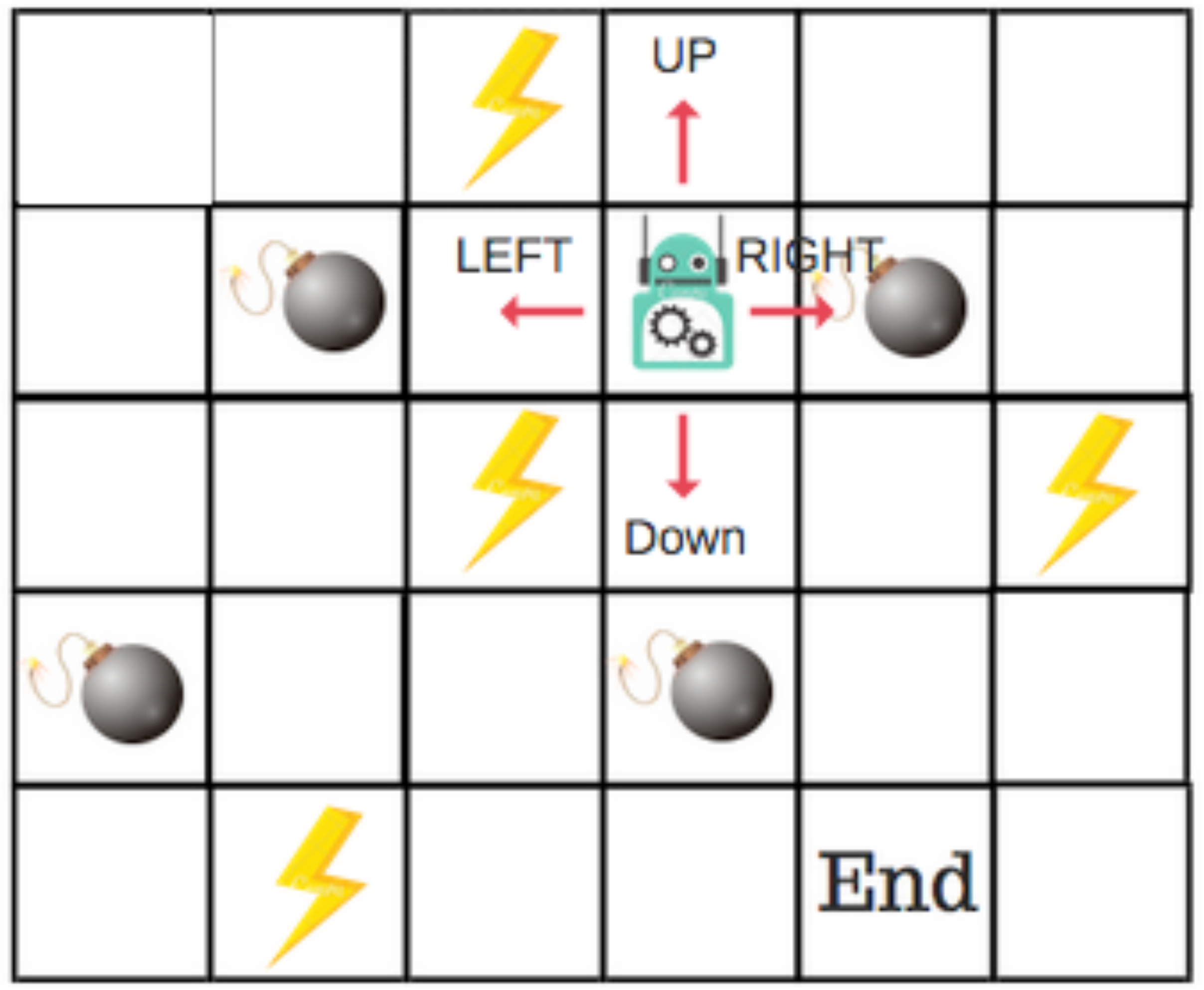
Finite MDPs: (S, A, T, R)

1. $S_t \in S$ for $t = 0, 1, 2, 3, \ldots$
2. $A_t \in A(s)$
3. $R_{t+1} \in R$
4. $T(s' \mid s, a) = \sum_{r' \in R} P(s', r' \mid s, a)$

Goal: maximize discounted **returns**

$$G_t = R_{t+1} + \gamma R_{t_2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

$$= R_{t+1} + \gamma G_{t+1}$$



- Optimization problem: (computationally) find **optimal policy** $\pi^*(S_t) = P(A_t \mid S_t)$
- Optimal policy $\pi^* : \pi \geq \pi' \Leftrightarrow v^*_{\pi^*}(s) \geq v_{\pi'}(s)$ for all s and $q^*_{\pi^*}(s, a) = \max_{\pi'} q_{\pi'}(s, a)$
- **Can be estimated from experience!**
  - e.g., value iteration methods for episode tasks

Sutton & Barto (2018, p. 48, Fig 3.1)

Multi-Armed Bandits

Finding the best restaurant

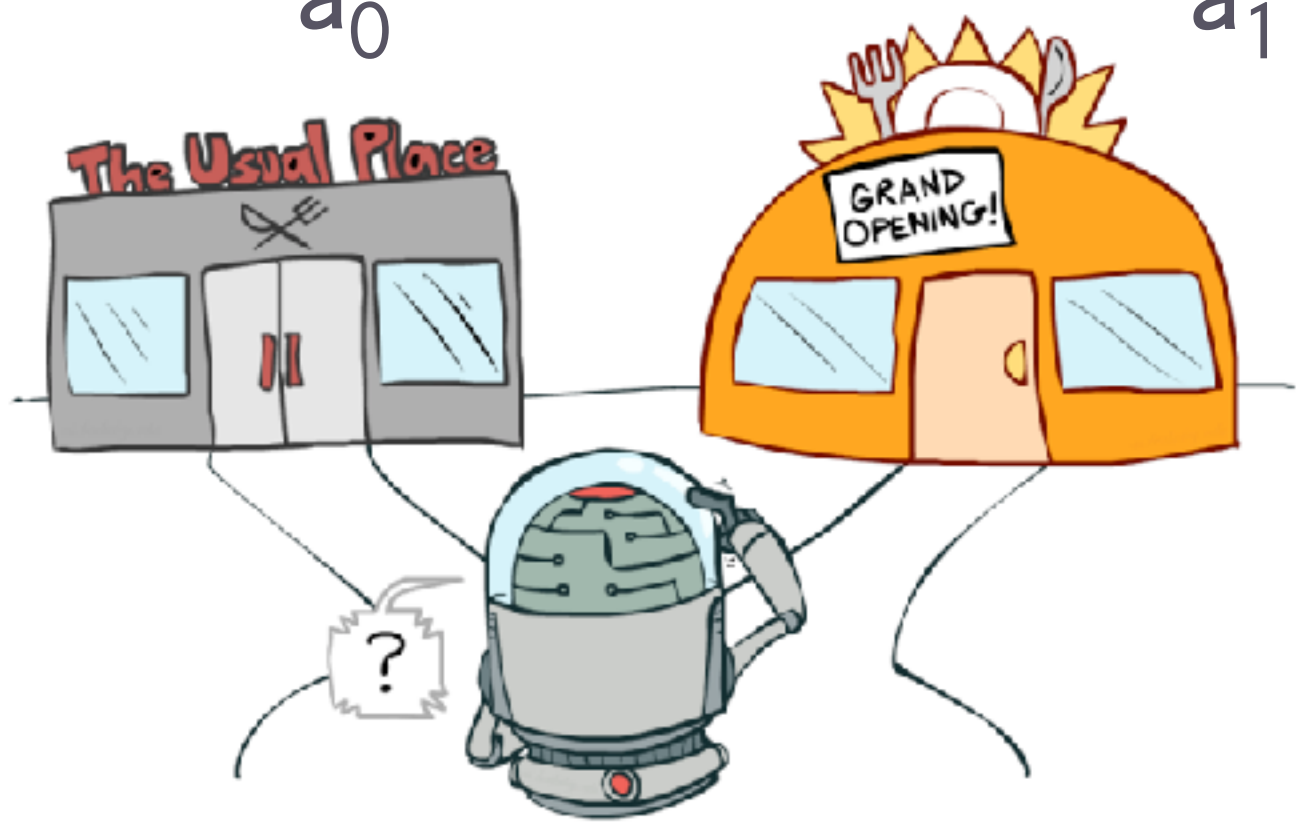
Finite MDPs: (S, A, T, R)

1. $S_t \in S$ where $S = \{s\}$

t is iteration over repeated choice now

2. $A(s) = \{a_0, a_1, \dots\}$
3. $R_t \in R$
4. $T(s' \mid s, a) = \sum_{r' \in R} P(s', r' \mid s, a)$

$A_t = a_i$ a_0 a_1



s

Goal: maximize reward over repeated action selection

Optimal **action-value function:**

$$Q^*_\pi(s, a) = \max_\pi \mathbb{E}[G_t \mid S_t = s, A_t = a]$$

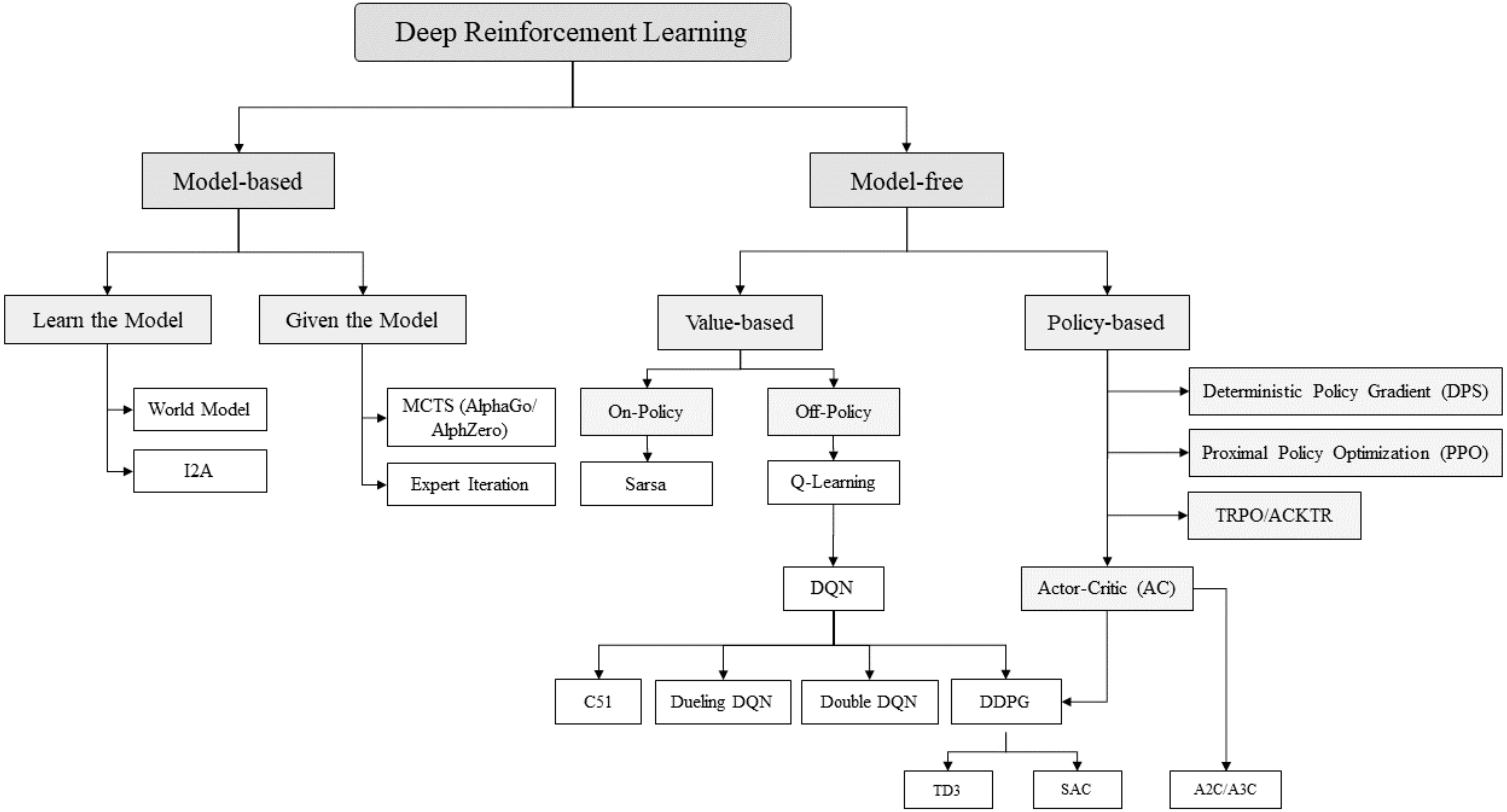
$$\downarrow$$

$$Q^*_\pi(a) = \max_\pi \mathbb{E}[R_t \mid A_t = a]$$

- Optimization problem: (computationally) find optimal policy $\pi^*(S_t) = P(A_t \mid S_t)$
- **Can be estimated from experience over repeated decision-making!**
  - e.g., with action-value methods like the sample-average method
- central problem: **exploration vs. exploitation**

image source

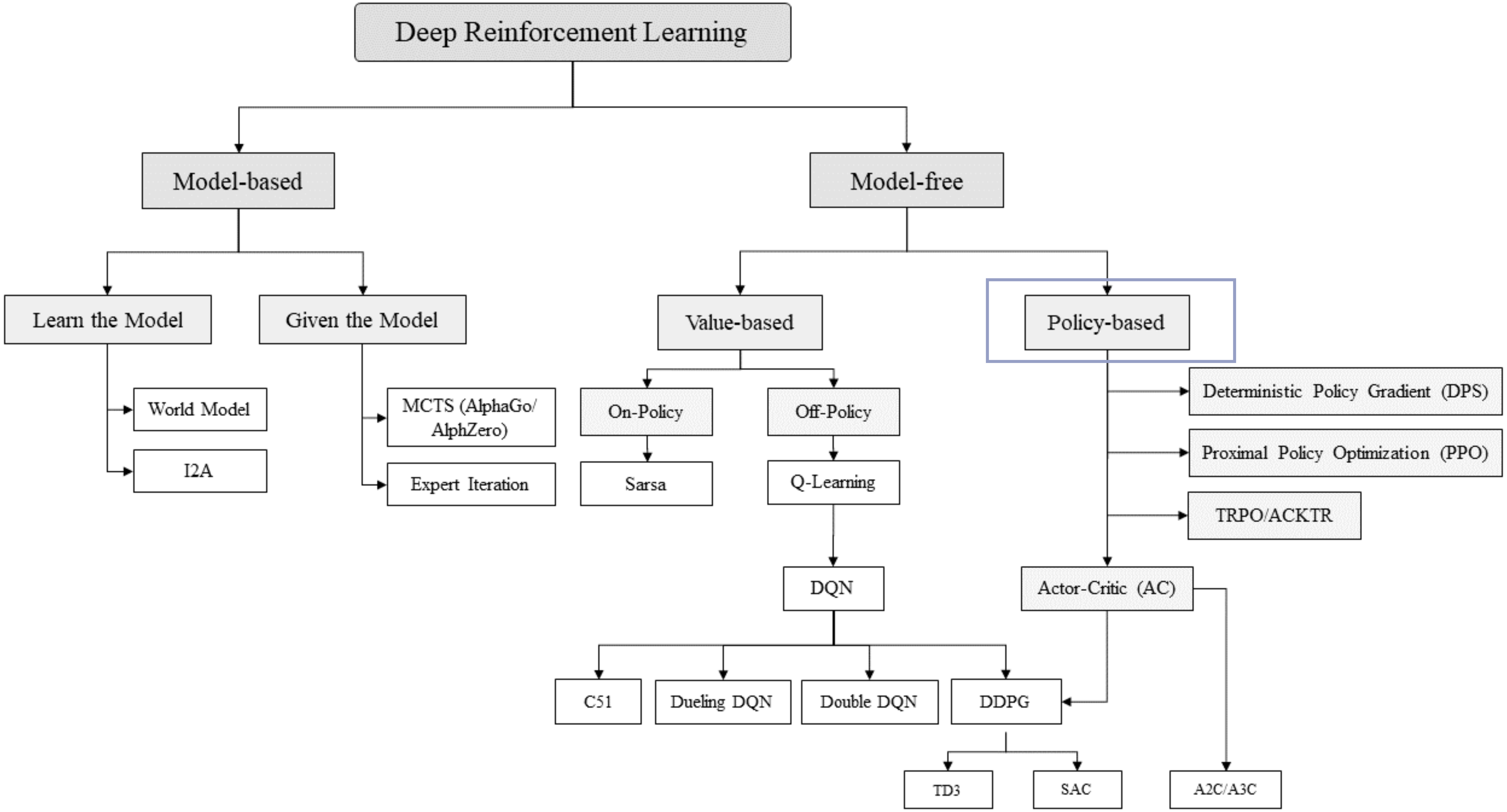
RL Algorithms
Approximating Optimal Policy



[3]

RL Algorithms

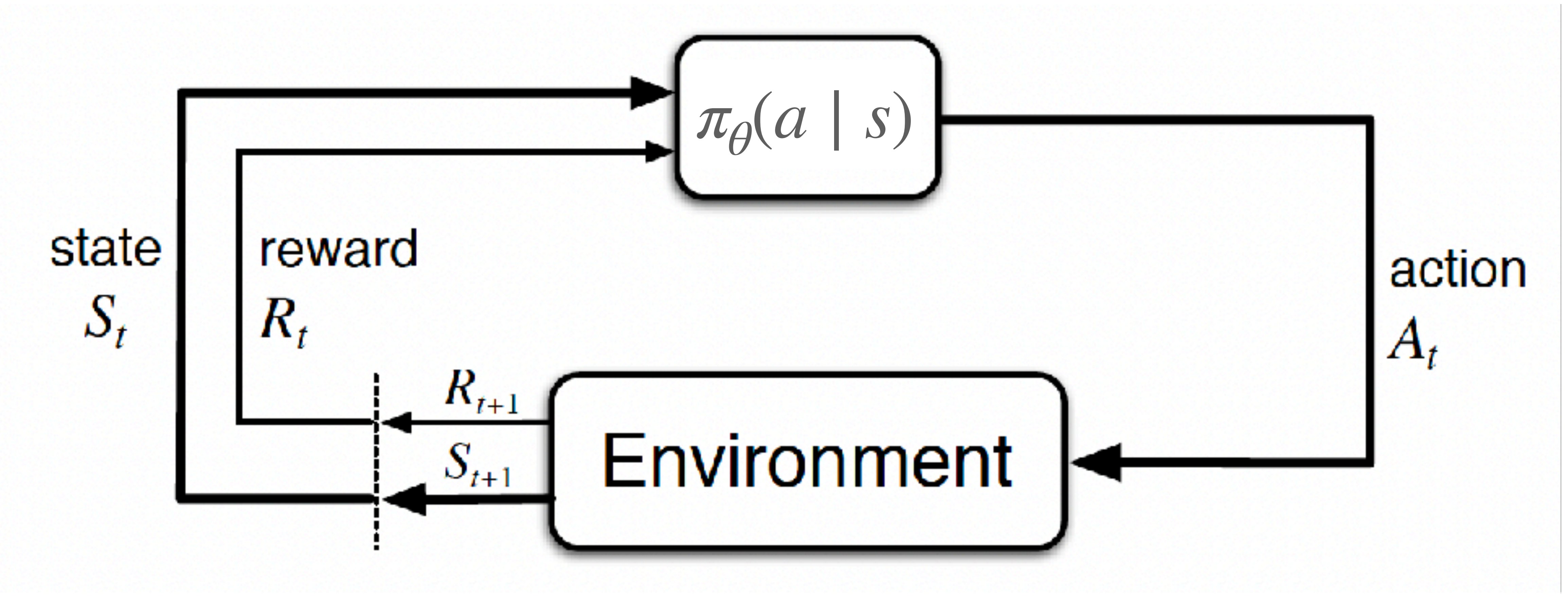
Approximating Optimal Policy



[3]

Policy-Gradient Methods
Introduction

- so far: deriving optimal policy from estimated value function
  - coming up with value functions might be difficult
  - state-value function doesn't prescribe actions
  - action-value functions require argmax
- idea: **optimize policy directly, such that expected reward is maximized**
  - think: optimize model with respect to objective function
- goal: find optimal θ
  - $\max_{\theta} \mathbb{E}_{\pi_\theta}[G_t]$
- recall LM optimization: tweak θ so as to minimize loss
  - Gradient descent: $\theta_{new} = \theta_{old} - \alpha \nabla L_\theta$
  - Now: gradient ascent: $\theta_{new} = \theta_{old} + \alpha \nabla L_\theta$



source

Policy-Gradient Methods

Policy-gradient theorem

- goal: find optimal θ
  - Now: gradient ascent: $\theta_{new} = \theta_{old} + \alpha \nabla L_\theta$
- we write τ for a sequence of states, actions, rewards and $R(\tau)$ for (discounted) return
  - $L(\theta) = \sum_\tau P(\tau, \theta)\ R(\tau)$
- sample-based policy gradient estimation

$$
\begin{aligned}
\nabla L(\theta) &= \nabla \sum_\tau P(\tau, \theta)\ R(\tau) = \sum_\tau \nabla_\theta P(\tau, \theta)\ R(\tau) \\
&= \sum_\tau \frac{P(\tau, \theta)}{P(\tau, \theta)} \nabla_\theta P(\tau, \theta) R(\tau) \\
&= \sum_\tau P(\tau, \theta) \frac{\nabla_\theta P(\tau, \theta)}{P(\tau, \theta)} R(\tau) = \sum_\tau P(\tau, \theta) \nabla_\theta \log P(\tau, \theta) R(\tau) \\
&\approx \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau^i, \theta) R(\tau^i)
\end{aligned}
$$

$\nabla \log(f(x)) = \nabla f(x)/f(x)$

Policy-Gradient Methods
REINFORCE

Policy gradient estimation: $\nabla L(\theta) = \sum_{\tau} P(\tau, \theta)\, \nabla_\theta \log P(\tau, \theta) R(\tau) \;\approx \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau^i, \theta) R(\tau^i)$

REINFORCE update rule:

for each episode:
  generate $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$ following $\pi(\,.\,|\,.\,,\theta)$
  for t in {0, 1, ….. T-1}:

$$G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

$$\theta_{t+1} \leftarrow \theta_t + \alpha\, G \nabla log\; \pi(A_t \,|\, S_t, \theta_t)$$

Sutton & Barto (2018)

Actor-Critic Algorithms
REINFORCE

(Advantage) Actor-Critic (A2C)

$$\theta_{t+1} = \theta_t + \alpha(G_t - \hat{v}(s, \mathbf{w}))\ \nabla log\ \pi(A\,|\,s, \theta_t))$$

- Actor-critic methods evaluate the value of the action taken in a state:

$$\theta_{t+1} = \theta_t + \alpha(\underbrace{\boxed{G_{t:t+1}}}_{\text{One-step return}} - \underbrace{\boxed{\hat{v}(s, \mathbf{w}))}}_{\text{Critic}}\ \nabla log\ \underbrace{\boxed{\pi(A\,|\,s, \theta_t))}}_{\text{Actor}}$$

-

Sutton & Barto (2018)

Policy-Gradient Methods

TRPO & PPO

$$\nabla L(\theta_t) \propto \mathbb{E}_\pi[G_t \nabla log\ \pi(A \mid s, \theta)]$$

- Rewriting policy-gradient theorem:

  $L_\theta = \mathbb{E}_t[\hat{A}_t\ log\ \pi_\theta(a_t \mid s_t)]$, where **advantage** $\hat{A}_t = G_t - b(S_t)$

- Improve PG with Trust Region Policy Optimization (TRPO):

  Surrogate objective

  $\boxed{L^{CPI}(\theta) = \mathbb{E}_t[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta\_old}(a_t \mid s_t)} \hat{A}_t]}$ such that $\hat{\mathbb{E}}_t[KL[\pi_{\theta\_old}(\,.\mid s_t)\pi_\theta(\,.\mid s_t)]] \leq \delta$

  Justification actually suggests penalty instead of constraint: $\mathbb{E}_t[r_t(\theta)A_t - \beta KL[\pi_{\theta\_old}(\,.\mid s_t)\pi_\theta(\,.\mid s_t)]]$

Schulman et al. (2015), Schulman et al. (2017)

Policy-Gradient Methods

TRPO & PPO

$$\nabla L(\theta_t) \propto \mathbb{E}_\pi[G_t \nabla log\ \pi(A \mid s, \theta)]$$

- Rewriting policy-gradient theorem:

  $L_\theta = \mathbb{E}_t[\hat{A}_t\ log\ \pi_\theta(a_t \mid s_t)]$, where **advantage** $\hat{A}_t = G_t - b(S_t)$

- Improve PG with Trust Region Policy Optimization (TRPO):

  Surrogate objective

  $\boxed{L^{CPI}(\theta) = \mathbb{E}_t[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta\_old}(a_t \mid s_t)}\hat{A}_t]}$ such that $\hat{\mathbb{E}}_t[KL[\pi_{\theta\_old}(\,.\mid s_t)\pi_\theta(\,.\mid s_t)]] \leq \delta$

  Justification actually suggests penalty instead of constraint: $\mathbb{E}_t[r_t(\theta)\hat{A}_t - \beta KL[\pi_{\theta\_old}(\,.\mid s_t)\pi_\theta(\,.\mid s_t)]]$

- Clip updates with PPO:

  Expectation over batch

  Boundaries

  $$L^{CLIP}(\theta) = \boxed{\mathbb{E}_t}[\boxed{min}(\boxed{r_t(\theta)\hat{A}_t}, clip(r_t(\theta), \boxed{1-\epsilon, 1+\epsilon})\hat{A}_t)]$$

  Lower bound on unclipped objective

  Standard probability ratio

Schulman et al. (2015), Schulman et al. (2017)

Policy-Gradient Methods

Language models as policies

Policy gradient estimation: $\nabla L(\theta) = \sum_{\tau} P(\tau, \theta)\, \nabla_\theta \log P(\tau, \theta) R(\tau) \approx \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau^i, \theta) R(\tau^i)$

- policy P: language model
- trajectories τ: generations from language model
- $\log P(\tau^i, \theta)$: log probability of a generation τ^i under the language model
- **$R(\tau^i)$: reward for generation τ^i**

Sutton & Barto (2018)

Summary

Reinforcement learning

- the central framework for formalizing RL problems are Markov Decision Processes (MDPs)
- task of RL is to solve MDP such that the expected return is maximized
  - and to find the optimal policy
- classical solution methods for MDPs include estimation of optimal state- and action-value functions
- policy gradient methods directly optimize the policy such that the expected return is maximized
  - can be applied to LMs!

Announcements

Solutions to exercises will be on Moodle on November 15th!

Next class (November 15th) online only!