ETH zürich

Multi-Agent Reinforcement Learning

Presented

by Thomas Asikis, asikist@ethz.ch

Self-organizing Multi-agent Systems

Multi agent systems

Multiple **interacting intelligent agents:**

- **Interacting:** communication, coordination & negotiation
- **Intelligent:** ability to understand, improvise, adapt & learn from environment and communication
- **Agents:** autonomous discrete entities

Self Organization

Local interactions between agents in a non-organized system lead to emergence of global order:

- Emergence of repeating and predictable patterns, usually from a random state.
- By design ***decentralized*** and ***distributed***.

*Haken, H. (2013). *Information and Self-Organization: A Macroscopic Approach to Complex Systems*. Springer Berlin Heidelberg. Retrieved from https://books.google.ch/books?id=XszyCAAAQBAJ

Global Patterns

- Aggregate values of individual agent metrics
- Global system metrics
- Distribution of metrics
- Signals (timeseries)
- Examples:
  - Total power consumption in Smart Grids
  - Stock market history of prices
  - Average traffic light wait time

Decentralized Architectures

- Centralized: **Central entity** manages actions and interactions between agents.
- Decentralized: All agents interact and act independently.

General Research Questions: What is interesting in the field

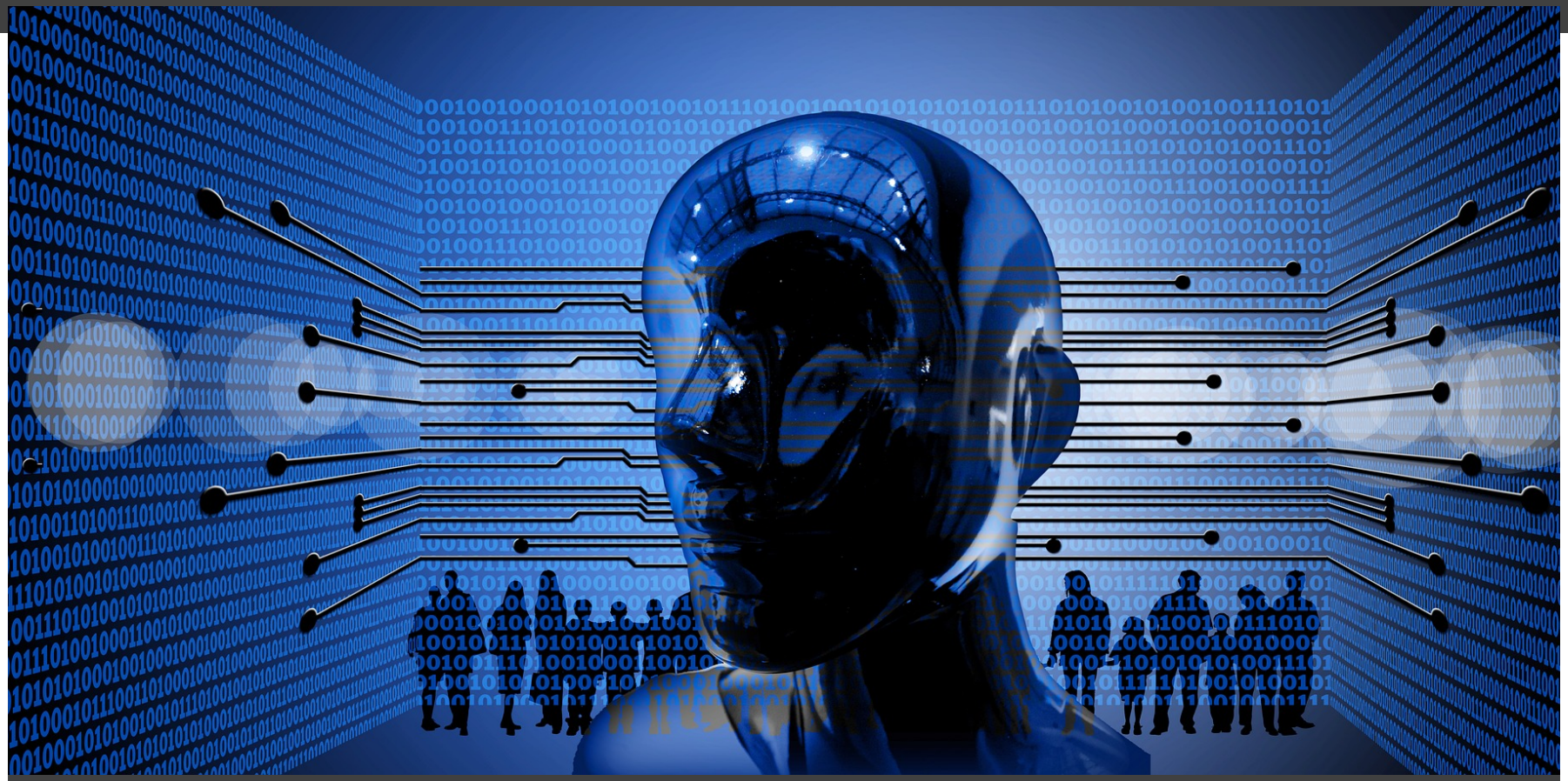
- **Define** & **evaluate** desired global patterns?
- What is the **relation** between agent **intelligence** and the emergence of **global patterns**?
- Can **desired global patterns** result from **specific intelligent agent designs/architectures**?

Autonomous Intelligent Agents

- Adaptability and interaction can lead to global behaviors that achieve optimally a global goal.
- Each agent → autonomous decision.
- Local goals may still be achievable
- All agents contribute to generate a specific "desired" global behavior or pattern

Learning

- Learn to understand: Analyze and process input from environment and other agents
- Learn to improvise: Make decisions and reason about their results
- Learn to adapt: Detect changes in the environment and other agent's behavior and decide for a new course of actions
- Learn to optimize: Given the environment and other agent's behavior, learn to make optimal decision regarding specific fitness functions
- Learn to interact: collaborate, compete and communicate



Reinforcement Learning

Designing Intelligent Agents

Reinforcement Learning

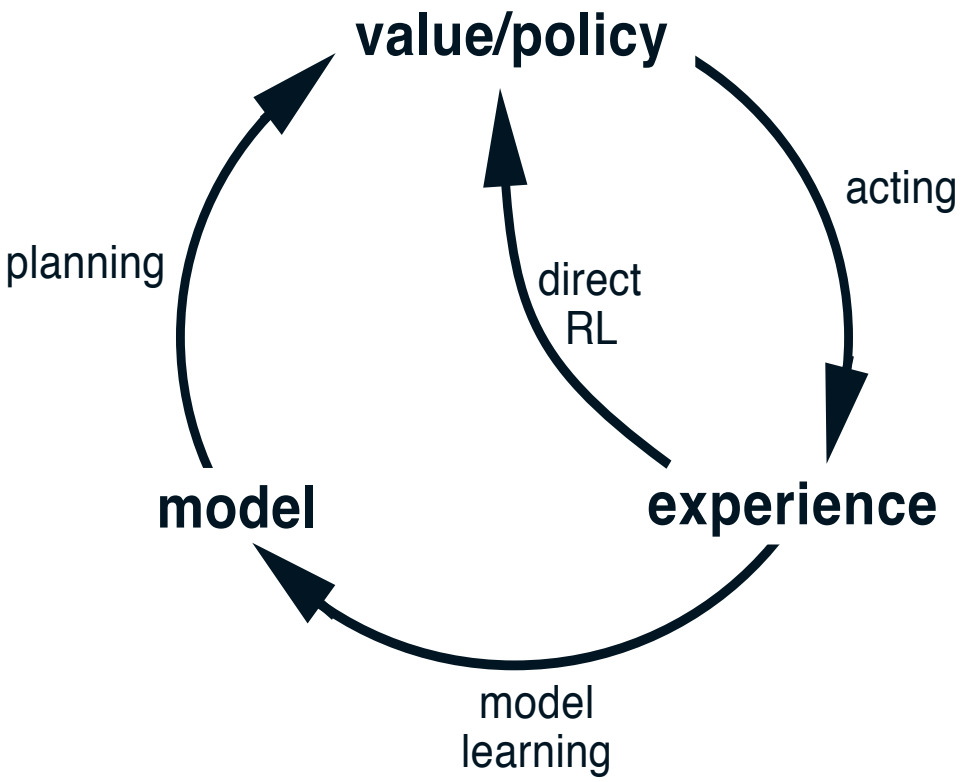
- Main elements: an *agent* and an *environment*
- *"A goal directed agent in an uncertain environment"*
- Learn a behavior by interacting with environment
  - ↳ Learn what actions to take given a situation
- Maximization of a reward (or minimization of cost)

Reinforcement Learning and Machine Learning

Machine Learning	Supervised	Learning from labeled set
		External supervisor
		Certainty: for given input always the same correct output
	Unsupervised	Finding structure/patterns in data.
		No goal
		Descriptive
	Reinforcement	Goal Driven agents
		Learn to optimize: Exploration vs Exploitation
		Uncertainty: same input does not always result to same output

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS. Ch.1Introduction, p.2

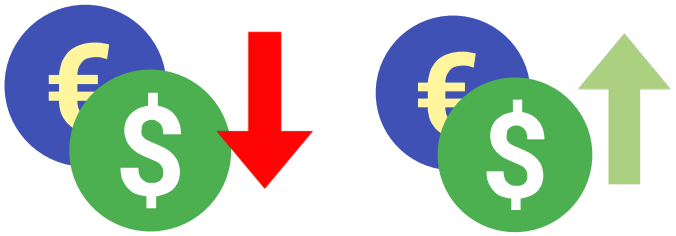
Reinforcement Learning and Planning



* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS. Ch.8 Planning and Learning with Tabular Methods, p.162-163

Learning to optimize

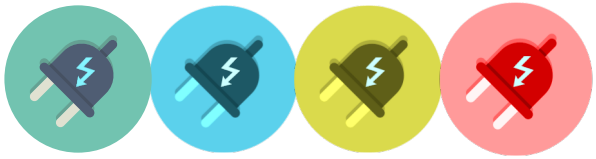
- Optimization Objective: $\min(O(a,s))$ V $\max(O(a,s)$
- Time t
- State $s_t \in \mathbb{S} \subset \mathbb{R}^{n \times m \times \cdots}$
- Actions $a_{t,} \in \mathbb{A} \subset \mathbb{R}^{n \times m \times \cdots}$
- Reward: $r_t = h(O(a,s))$
- State transition: $s_{t+1} = g(s_t, a_t)$



Agent Action

- Take an action and interact with the environment
- Possible actions may change

- Examples:
  - Choose a different power plan
  - Invest in a stock
  - Eat a burger



Agent State

s

- Perception of agent's (local) environment
- Changes by agent actions

- Examples:
  - Current power consumption
  - A person's preferences
  - Available budget

Value Function

Estimate how optimal:

- for an agent to be in a state $V(s)$ - state value function
- to take an action given a state $Q(s, a)$ - action-state value functions
- True value functions – objective truth/not estimates:

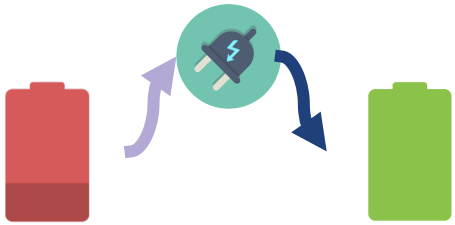
$$v(s), q(s, a)$$

E.g. How optimal is:

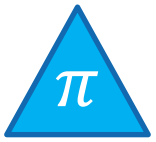
- my power consumption
- selecting a power plan "A" given my power consumption
- my current budget
- investing in stock "X" given my budget

State transition

- Deterministic
- Stochastic
  - ↳ Same action-state may lead to different states
- Affected by:
  - Current action and state
  - and past actions and states



Policy π



The strategy to select an action:

$$\pi(a_t|s_t)$$
$$\pi: \mathbb{S} \rightarrow \mathbb{A}$$

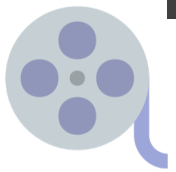
- Can be probabilistic or deterministic
- Infinite policies can be defined
- Often policy is affected by a value function
- Estimating a value function given a policy is called *prediction problem*
- E.g. How to select a:
  - new power plan
  - new investment

Reward

- Skirner, Behavioral Theory
- Positive is encouragement
- Negative is punishment
- Reward is not the Objective:
  e.g. objective difference in consecutive timesteps:

$$r_t = O_t - O_{t-1}$$

Episode



- A sequence of time-steps t
- End: a termination criterion is reached
- Goal is achieved or failed
- Continuous/Ongoing tasks: $T = \infty$
- Terminal state: s_T

Cumulative Reward & Return

- Reward in the long term
- Greedy selection of current maximum reward, or …
- Non-optimal selection now, long term optimization later
- Usually a discount factor is used (discounted reward):

$$R_t = \sum_{l=0} \gamma^l r_{t+l} \ , 0 < \gamma < 1$$

- The return is the cumulative reward at the end of an episode:

$$G_t = \sum_{l=0}^{T} \gamma^l r_{t+l}$$

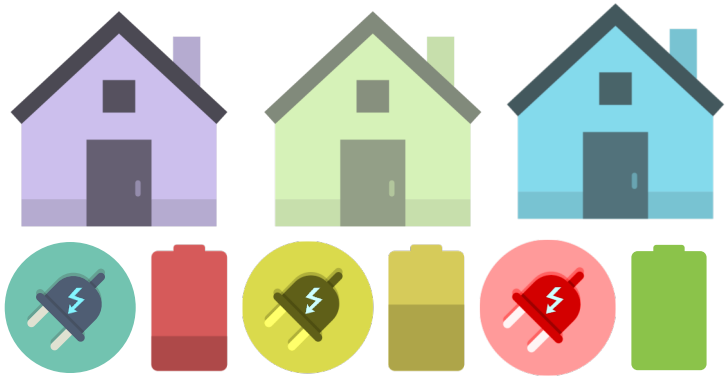
The control problem: Finding an optimal policy

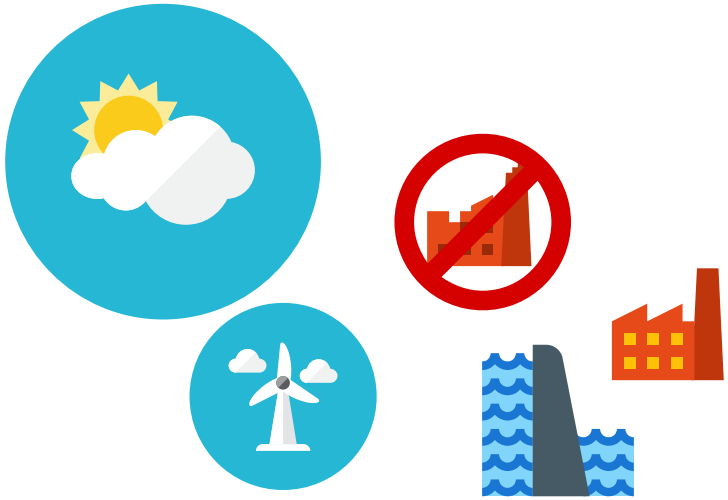
- Optimal policy: The policy that maximizes return by selecting optimal actions:

$$\pi^*(a_t|s_t, o_t)$$
$$s.t. G_t = G_t^*$$

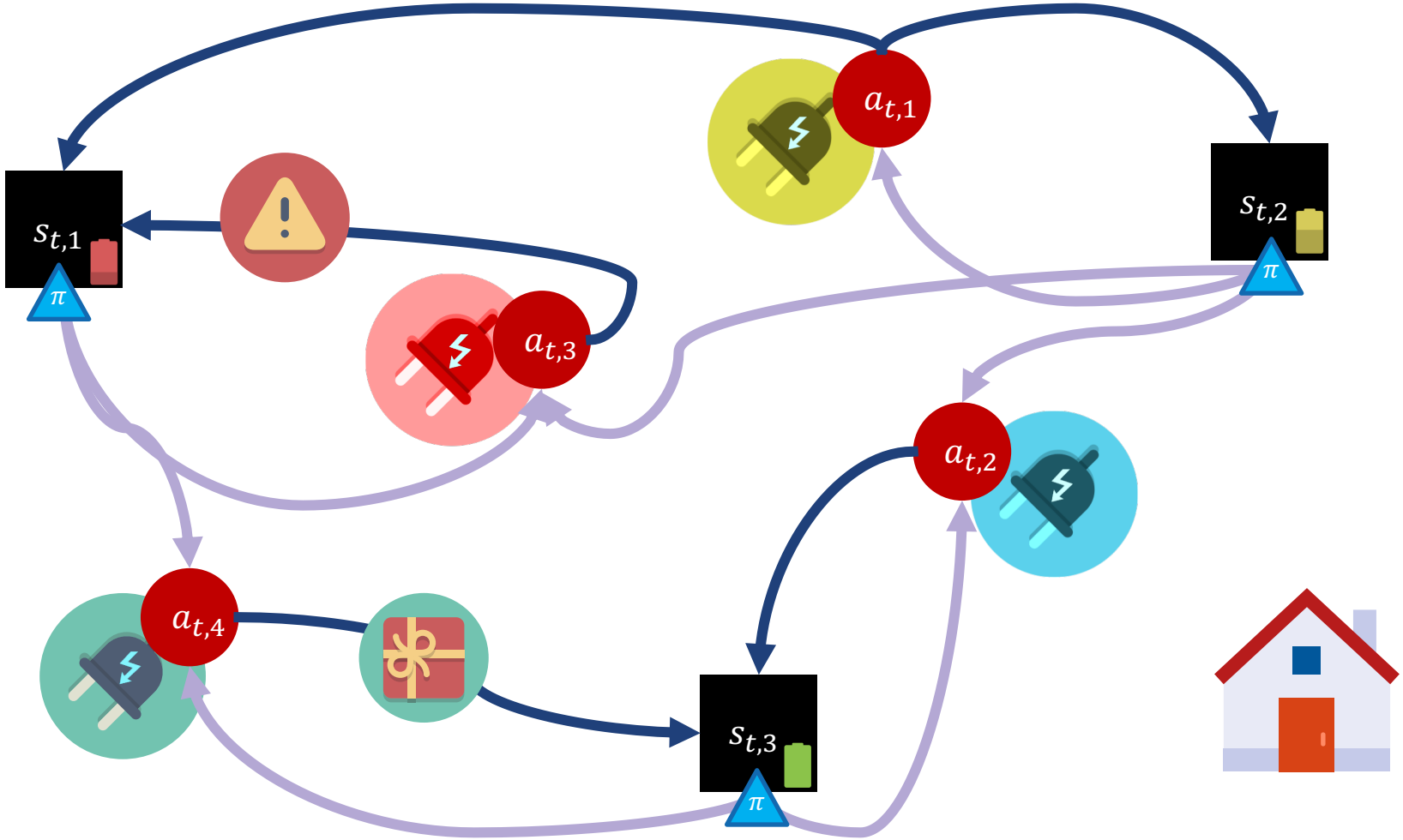
Environmental Observation o_t

- Information about general environment
- Can overlap with agent's state
- May contain information of the states of other agents
- Policy can be redefined as: $\pi(a_t|s_t, o_t)$



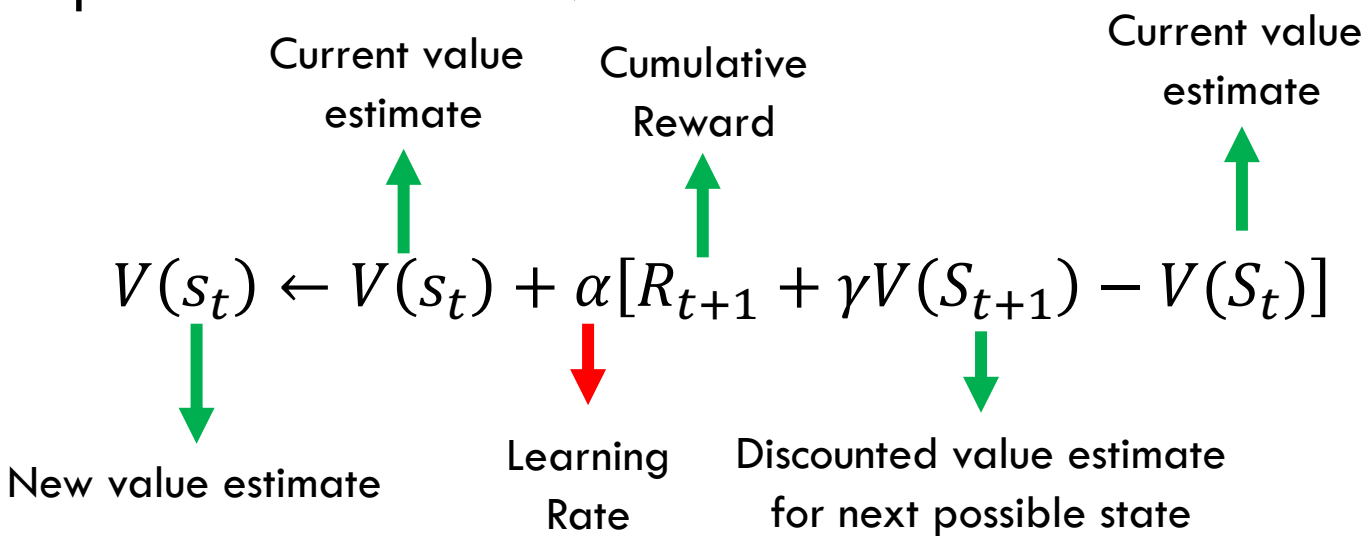


Markov Decision Process (MDP)



Temporal Difference

- Combination of Dynamic Programming and Monte-Carlo methods
  - ↳ Monte Carlo: learn from experience without knowing environment dynamics
  - ↳ Dynamic programming: update estimates based on other estimates, without waiting for an episode to finish
- A simple state value TD Method:



On-policy Reinforcement Learning

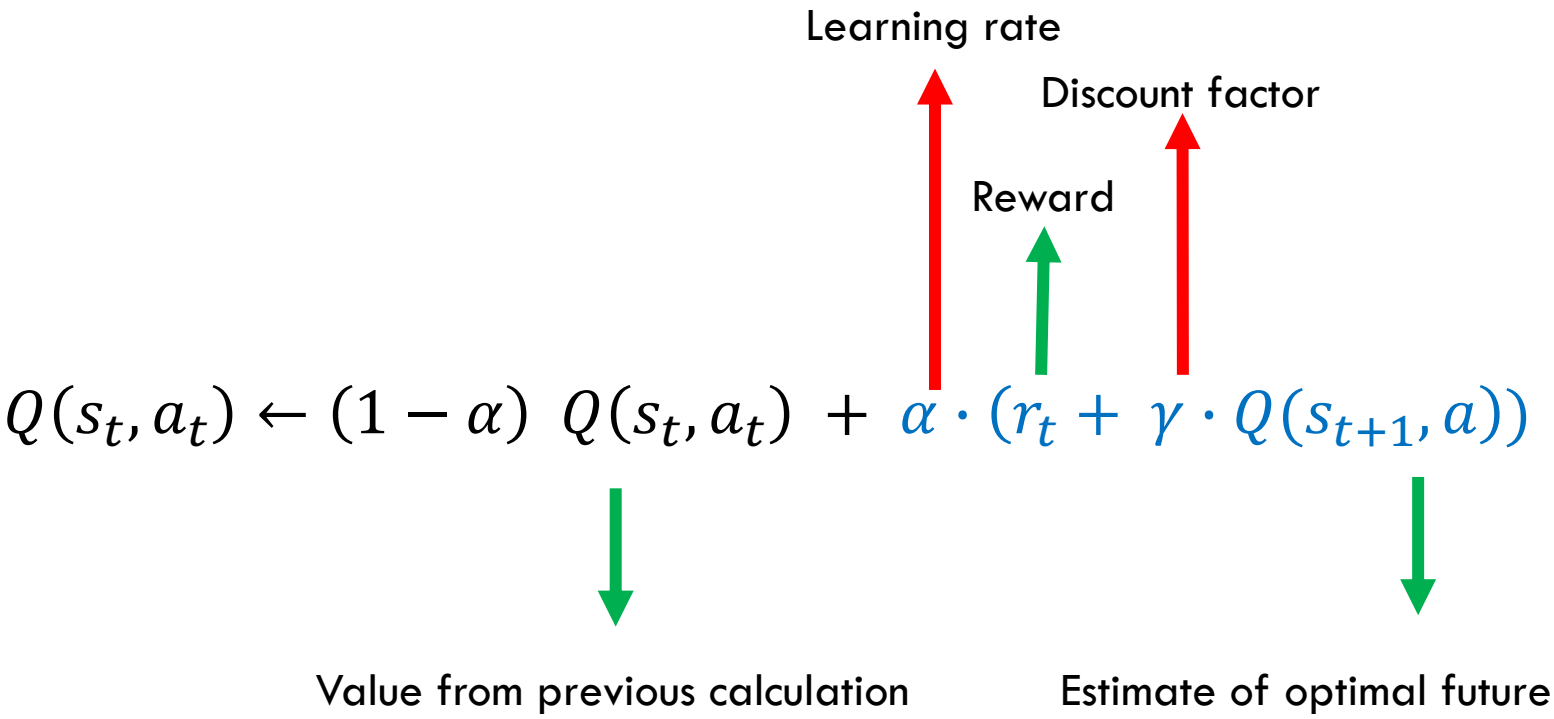
Update a policy based on actions taken by that policy

Example:

1. Chose action a based on π (e.g. ϵ-greedy*)
2. State transition
3. Update π based on outcome of a

nice discussion: https://datascience.stackexchange.com/questions/26471/is-my-understanding-of-on-policy-and-off-policy-td-algorithms-correct
*choose an action greedily or randomly based on a probability, e.g. $\epsilon = 0.1$ for random choice, and $1 - \epsilon = 0.9$ for deterministic

SARSA


Learning rate
Discount factor
Reward
$Q(s_t, a_t) \leftarrow (1-\alpha)\ Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot Q(s_{t+1}, a))$
Value from previous calculation
Estimate of optimal future value

Off-policy Reinforcement Learning

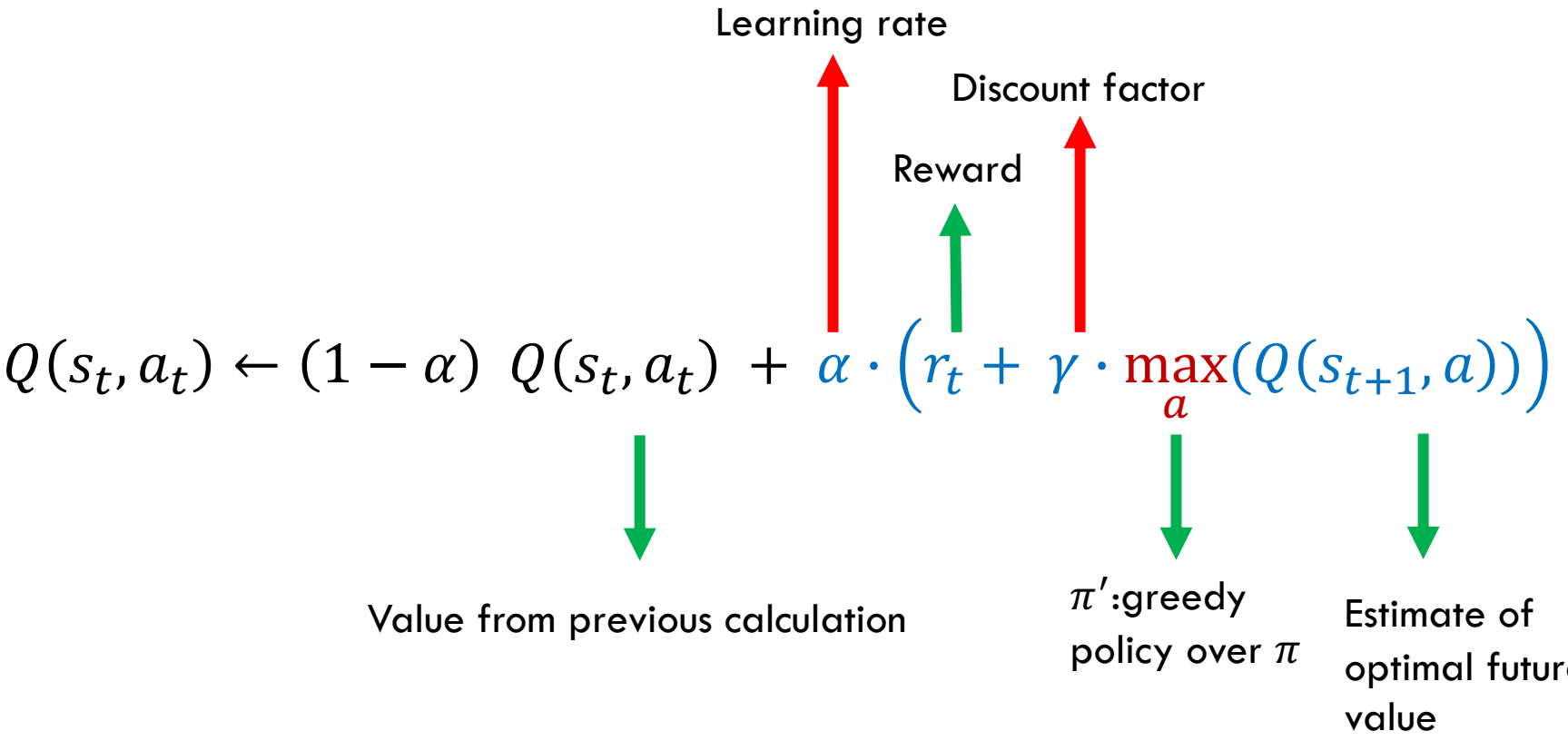
Update a policy based on action taken by other policies

Example:

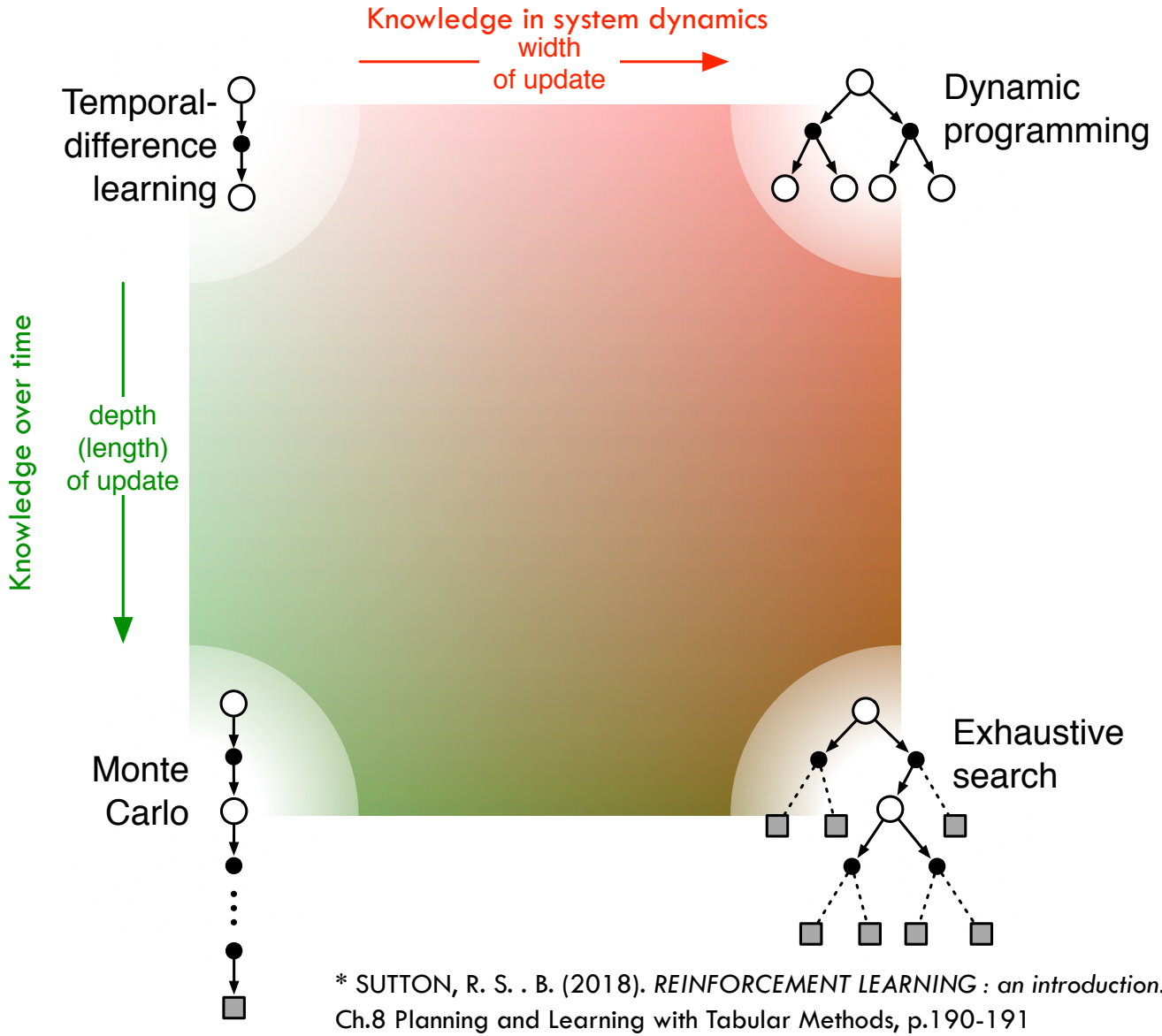
1. Chose action a based on $\pi' \neq \pi$
2. State transition
3. Update π based on outcome of a

* π': target policy that is evaluated or learned (e.g. $\max(Q_{t+1})$)
  π : behavior policy that affects choice of actions (e.g. ϵ-greedy)

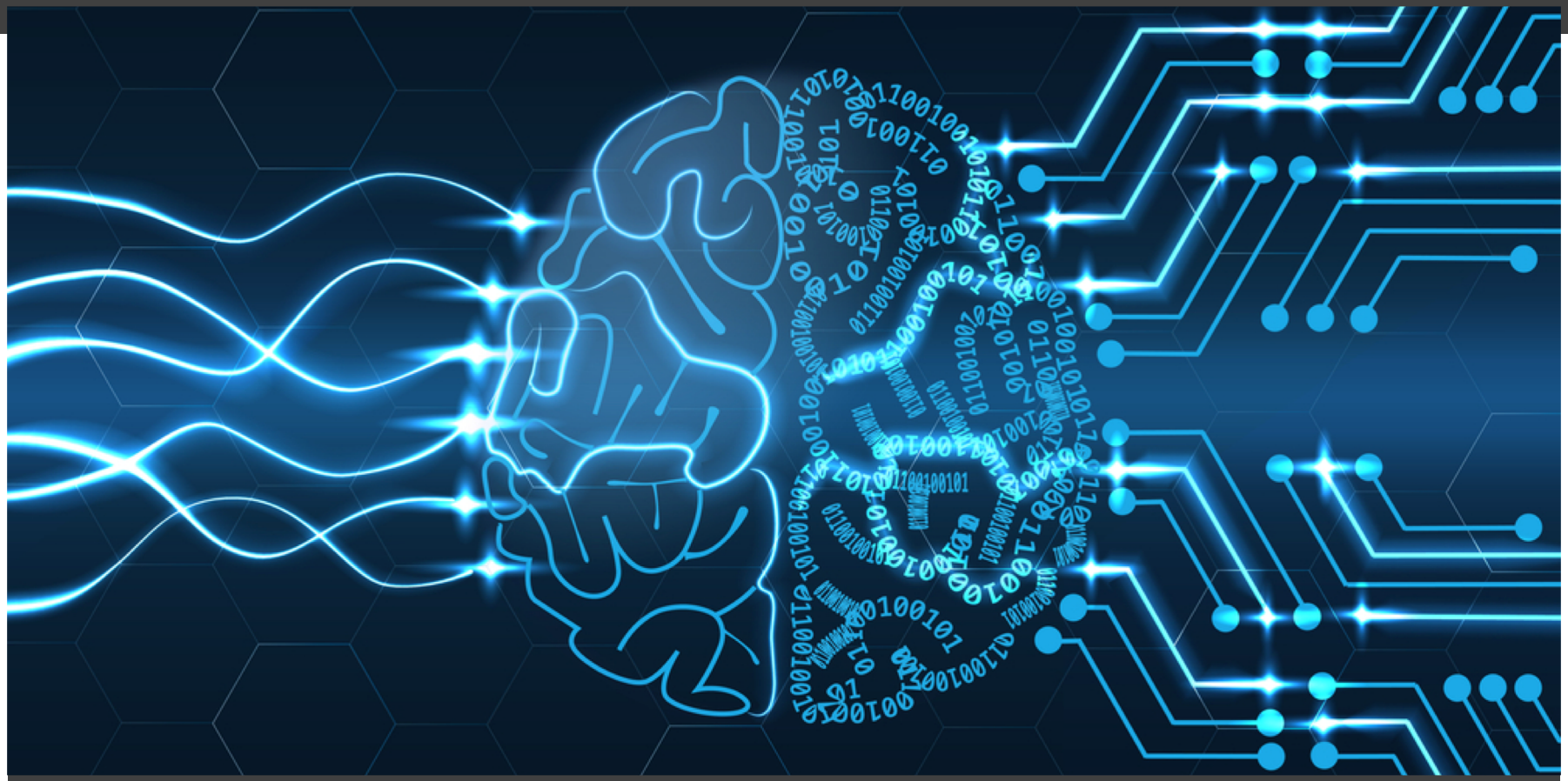
Q-learning


Learning rate
Discount factor
Reward
$Q(s_t, a_t) \leftarrow (1-\alpha)\ Q(s_t, a_t) + \alpha \cdot \left(r_t + \gamma \cdot \max_a (Q(s_{t+1}, a))\right)$
Value from previous calculation
π':greedy policy over π
Estimate of optimal future value

Dimensions of the Reinforcement Learning Problem



* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS. Ch.8 Planning and Learning with Tabular Methods, p.190-191



Deep Reinforcement Learning

Enhancing Predictive Capabilities

The Prediction Problem: Precise estimation of values

- Parametrize the value function: $V(s, \boldsymbol{w})$
- Given a policy π, update parameters w s.t.:

$$\min_{w} \left(\overline{VE(w)}\right)$$

Where $\overline{VE}$ is a value estimator:

$$\overline{VE(w)} = \sum_{s \in \mathbb{S}} \mu(s) \cdot [V_{\pi}(s) - V(s, \boldsymbol{w})]^2$$

Iteratively update the parameters with gradient update:

$$w' = w + \eta \nabla_{\mathrm{w}} \overline{VE(w)}$$

η: as in machine learning, it is the learning rate. Usually a constant in $(0,1)$.

On-policy Distribution

$\mu(s)$:

- The number of timesteps spend in the state.
- Also called on-policy distribution.
- Can also be parametrized or estimated as well

Policy Gradient

- A policy can be parametrized $\pi(a|s, o, \boldsymbol{\theta})$
- Approximate optimal policy by learning parameters

$$\pi(\boldsymbol{\theta}) \rightarrow \pi^*$$

- Update $\theta' = \theta + \eta \nabla_\theta J(\theta)$, where $J = f(\mu, Q, \pi)$ is a learning performance metric (policy estimator).
- Policy gradient theorem: The learning performance gradient is proportional to $\mu, Q, \nabla\pi$

$$\nabla_\theta J(\theta) \propto \sum_{s \in \mathbb{S}} \mu(s) \sum_{a \in \mathbb{A}} Q_\pi(s, a) \nabla_\theta \pi(a|s, \boldsymbol{\theta})$$

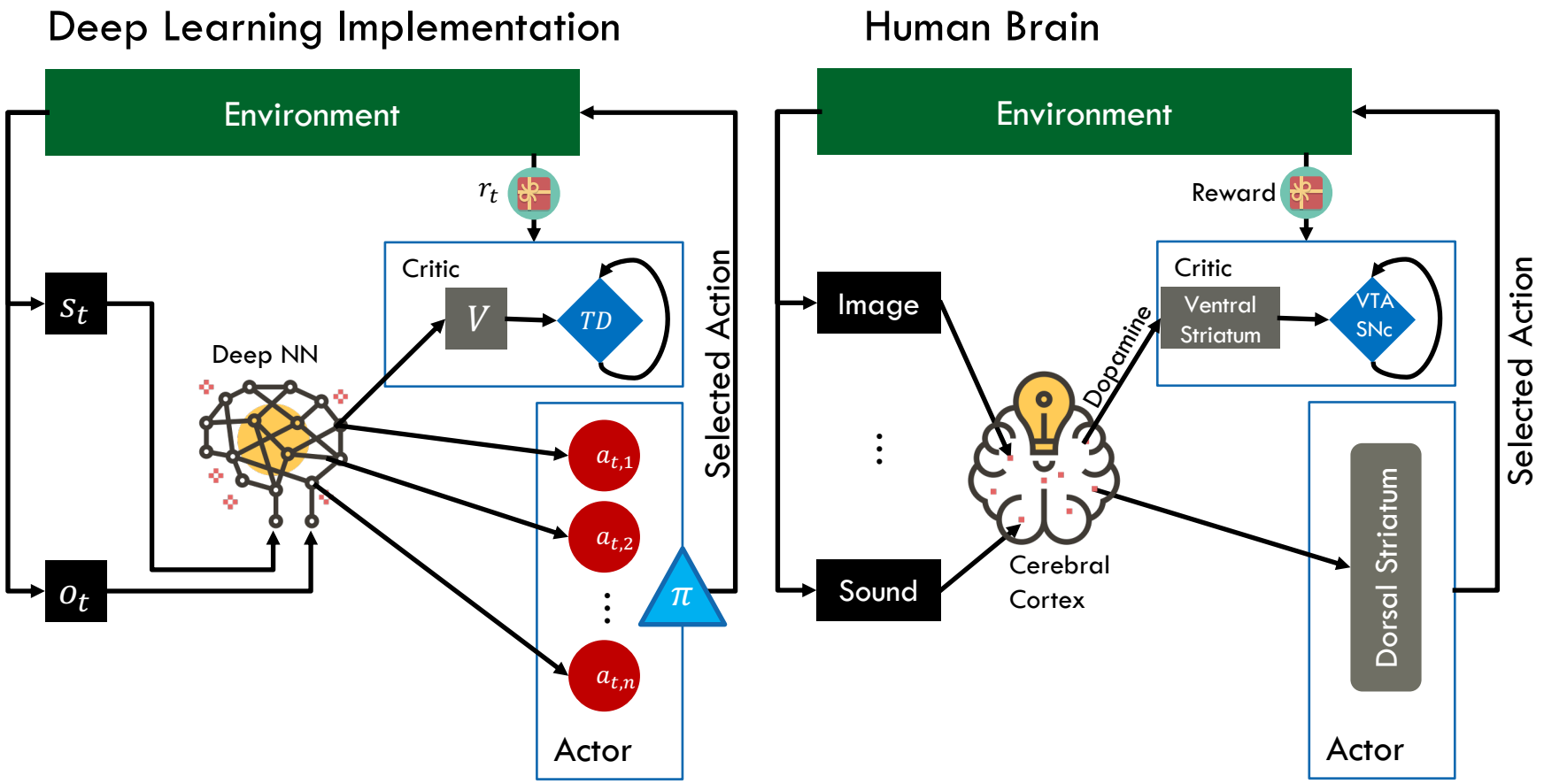
Actor Critic

- Combine policy gradient with value estimation
- Use one set of parameters $\boldsymbol{w}$ to calculate accurate estimates with V
- Another set of parameters $\boldsymbol{\theta}$ to estimate π
- Update both using gradient update with different learning rates

Deep Learning

- High performance in learning tasks
- Requires high data volumes ~ easily acquired in combinatorial problems
- Efficient function approximation using non-linearity
- A variety of learning loss functions to be used as value $\overline{VE}$ or policy J estimators

Neural Actor-Critic



Multi-agent Reinforcement Learning

Enabling Collaboration and Competition

Collective Learning

- Klopf's Hedonistic neuron → all neurons of dorsal and ventral striatum of brain can be treated as cooperating RL agents
- Reward signal is usually shared by all agents
- The reward signal is not affected by a single agent actions

Global vs Local Goals

- Optimize a local reward
- Optimize global reward
- May be contradicting or orthogonal

Examples of global rewards and local rewards:

- Power plans that increase user comfort vs energy saving power plans
- Buying sustainable products vs buying products that satisfy the user

Learn to Communicate

- Messages are exchanged: rewards, actions, states
- Messages can be used as an observation input o_t
- Agent reward functions can include aggregates of other agent rewards or objectives
- Message exchange is done via communication protocols, like gossiping

Learning to Compete

- Agents may need to compete with each other
- Rewards are usually local objective functions
- Rewards are designed to be opposite or orthogonal between agents:

$$r_i \uparrow\downarrow r_k \vee r_i \perp \mathrm{r}_k$$

- Fake communication $\sim$ adversarial models, usually for competition or malicious behavior

Learn to Collaborate

- Objective functions can be local and global
- Communication or observation of other agent actions
- Usually rewards combine both global and local objectives
- Rewards between agents are most of the times co-monotonous:

$$r_i \uparrow\uparrow r_k$$

Heterogeneous Agents

- Agents with different state & action spaces
- Reward functions and objectives might be different for each agent
- Agent interactions can be modeled via reward and message exchanges
- Heterogeneous objective functions can be combined in agent rewards, but the renormalization might be needed

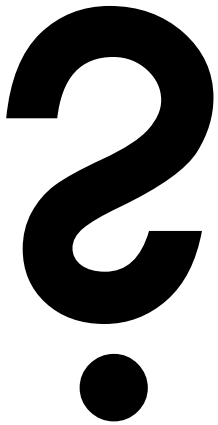
Applications of Multi-agent RL

- Smart Grids
- Autonomous Vehicles
- Traffic Control
- Games

Summary

- Reinforcement Learning can be used for uncertain and dynamic environments
- Exploitation vs Exploration
- Control Problem: optimal policy, Prediction Problem: optimal estimation
- Deep learning can be used to tackle the prediction problem
- Actor-Critic modelling tackles both Prediction and control problems
- Multi-agent reinforcement learning is possible
- Collaboration and Competition can be modeled via reward functions
- Communication is possible via protocols and exchange of agent states
- Heterogeneous agents can be combined

Questions

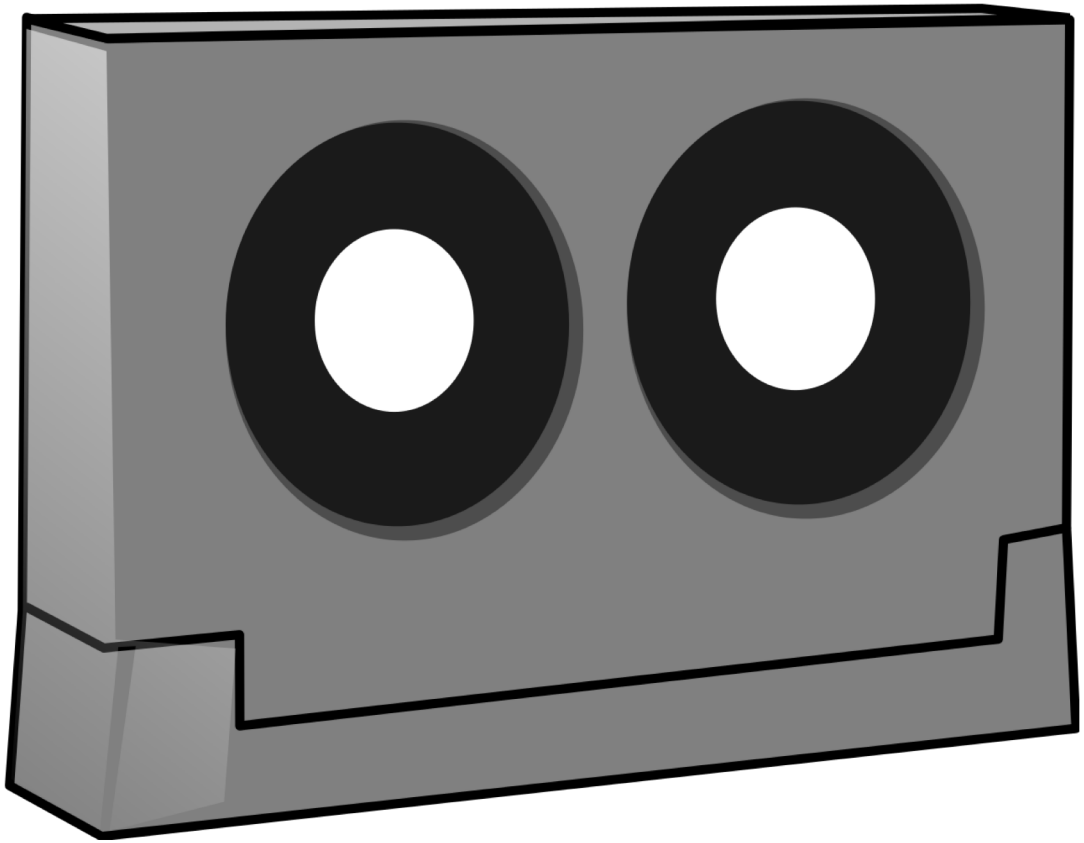


Selected References

- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning : an introduction*. MIT Press. Retrieved from https://drive.google.com/file/d/1opPSz5AZ_kVa1uWOdOiveNiBFiEOHjkG/view

  Interesting Chapters:

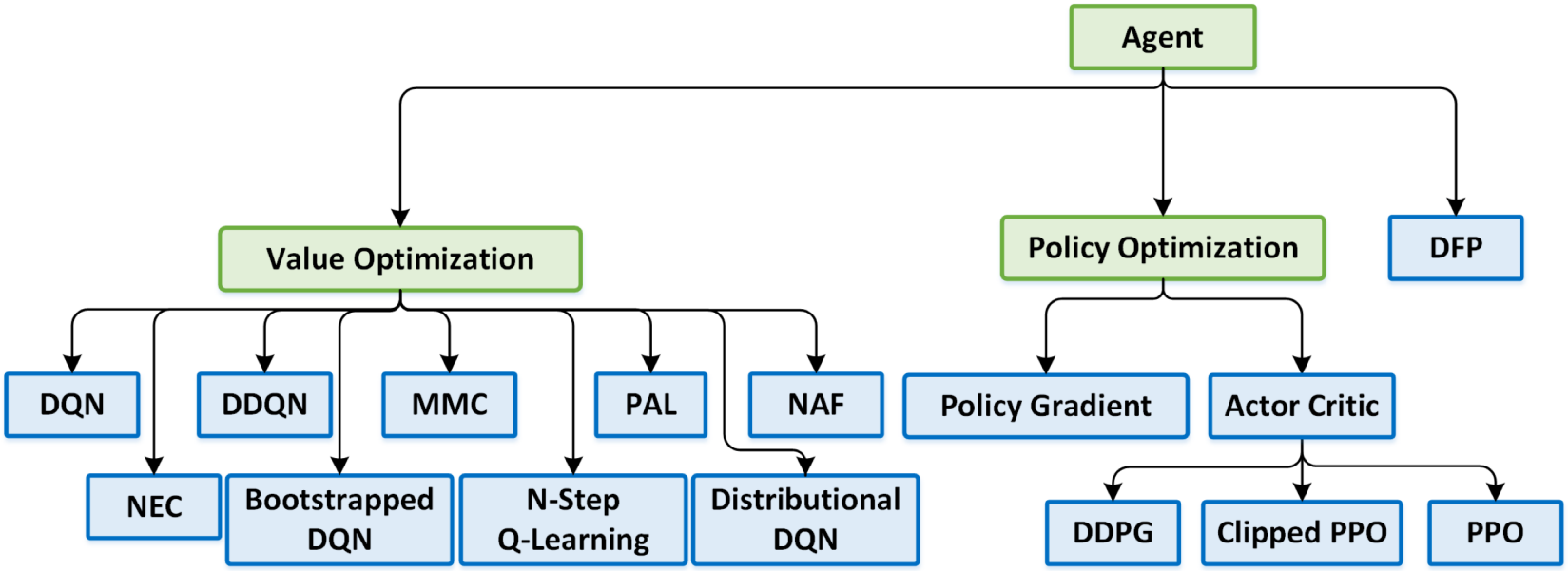
  - 1 Introduction: 1.1 Reinforcement Learning, 1.2 Examples, 1.3 Elements of Reinforcement Learning
  - 3 Finite Markov Processes: 3.1 The Agent-Environment Interface, 3.2 Goals and Rewards, 3.3 Returns and Episodes
  - 6 Temporal-Difference Learning: 6.1 TD Prediction, 6.2 Advantages of TD Prediction Methods, 6.4 Sarsa: On-policy TD Control, 6.5 Q-learning Off-policy TD Control
  - 8 Planning and Learning with Tabular Methods: 8.1 Models and Planning, 8.2 Dyna: Integrated Planning, Acting, and Learning, 8.13 Summary of Part I: Dimensions
  - 9 On-policy Prediction with Approximation: 9.1 Value-function Approximation, 9.2 The Prediction Objective $(\overline{VE})$, 9.3 Stochastic-gradient and Semi-gradient Methods
  - 13 Policy Gradient Methods: 13.1 Policy Approximation and its Advantages, 13.2 The Policy Gradient Theorem, 13.5 Actor-Critic Methods
  - 14 Psychology: 14.1 Prediction and Control
  - 15 Neuroscience: 15.4 Dopamine, 15.7 Neural Actor-Critic, 15.9 Hedonistic Neurons, 15.10 Collective Reinforcement Learning
  - 17 Frontiers: 17.3 Observations and State
- Zhang, K., Yang, Z., Liu, H., Zhang, T., & Basar, T. (2018, July 3). Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents. Retrieved from http://proceedings.mlr.press/v80/zhang18n.html
- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., … Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359. https://doi.org/10.1038/nature24270



Backup Slides

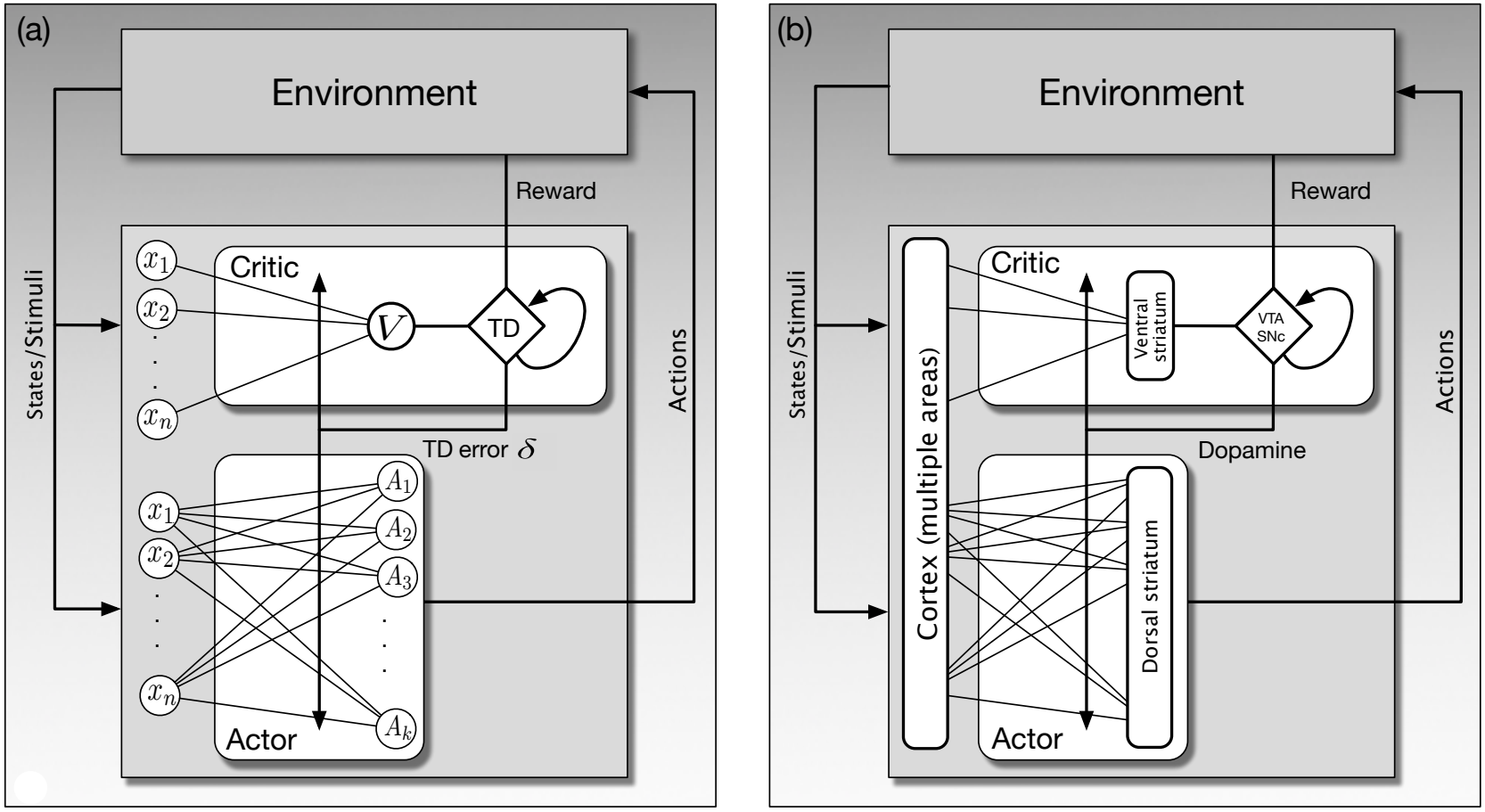
Just in case

An RL Taxonomy



https://ai.intel.com/reinforcement-learning-coach-intel/

Neural Actor Critic



Notation Table I

Symbol	Explanation
i, j	Agent indeces
$O(x)$	An objective function that operates on input x
t	A timestep
$a_{t,i}$	An action taken by agent i at time t
$s_{t,i}$	The agent state of agent i at time t
$r_{t,i}$	The reward received by an agent i at time t
$g(s_{t,i}a_{t,i})$	The state transition that happens from time t to $t+1$ given agent i state and selected action
$V(s_{t,i})$	The value function, that provides the agent i estimates about how optimal is its state at time t
$Q(s_{t,i}, a_{t,i})$	The action-value function, that provides the agent estimates about how optimal is its state $s_{t,i}$ and the action it selected $a_{t,i}$ at time t

Notation Table II

Symbol	Explanation
$v(s_{t,i})$ $q(s_{t,i}, a_{t,i})$	The true state and action-state value functions that provided the actual value of how optimal the state $s_{t,i}$ and selected action $a_{t,i}$ are for agent i at time t. Usually, they are not known.
$\pi(a_{t,i}\|s_{t,i})$	The policy that selects the action $a_{t,i}$ given the state $s_{t,i}$ for the agent i at time t.
γ	The discount factor, usually $0 \leq \gamma \leq 1$, which discounts future rewards and values
$R_{t,i}$	The cumulative reward from time t and on, for agent i
$G_{t,i}$	The return, which is the cumulative reward from time t until the end of an episode (e.g.. when a goal is met or failed for an agent i).
$\pi^*(a_{t,i}\|s_{t,i})$	The optimal policy that maximizes cumulative reward and return
$o_{t,i}$	The environmental observation of an agent i at time t. Usually modelled along with state.

Notation Table III

Symbol	Explanation
α	The learning rate in temporal difference models, usually $0 \leq \alpha \leq 1$
$\max_x f(x)$	The maximization of a function $f(x)$ in regards to x
$\min_x f(x)$	The minimization of a function $f(x)$ in regards to x
w	A tensor of learnable parameters for value estimation
$\overline{VE(w)}$	Value estimator of via parameters w
η	The learning rate for value and policy estimation via parameter learning
$\nabla_w f(w)$	The gradient of function $f(w)$ in regards to elements of w
$\mu(s)$	On-policy distribution. The amount of timesteps spent (or expected to be spent) on the state s
θ	A tensor of learnable parameters for policy estimation
$J(\theta)$	Policy estimator via parameter learning