

Reinforcement Learning Training 2025

Welding + RL

Journal Article • 10.1007/S00170-018-2864-2

1. An intelligent weld control strategy based on reinforcement learning approach

Zeshi Jin, Haichao Li, Hongming Gao

1 Feb 2019 - The International Journal of Advanced Manufacturing Technology

66 28



Request PDF



Podcast



Chat



66

The paper conducts simulation experiments using actor-critic reinforcement learning (ACRL) for welding process control, specifically focusing on gas tungsten arc welding (GTAW) and gas metal arc welding (GMAW) models, followed by open-loop and closed-loop control experiments to verify controller reliability.

Journal Article • 10.1007/S40747-021-00366-1

2. Collision-free path planning for welding manipulator via hybrid algorithm of deep reinforcement learning and inverse kinematics

Jie Zhong, Tao Wang, Lianglun Cheng

10 Apr 2021 - Complex & Intelligent Systems

66 36



PDF



Summary



Podcast



Chat



66

The paper conducted multiple path planning experiments for welding manipulators using a hybrid algorithm of deep reinforcement learning and inverse kinematics, demonstrating improved convergence performance, optimality, and robustness compared to traditional planning algorithms in various welding scenarios.

Journal Article • 10.1109/ACCESS.2020.2998052

3. Adaptive Laser Welding Control: A Reinforcement Learning Approach

Giulio Masinelli, Tri Le-Quang, Silvio Zanolì +2 more

27 May 2020 - IEEE Access

66 39



PDF

















Summary



Podcast

Reinforcement learning welding experiments involved collecting acoustic and optical signals from 15 welds at various laser powers (20-120 W). The system autonomously learned to optimize weld quality, achieving a reference weld with 80 W power and 150 μ m penetration depth.

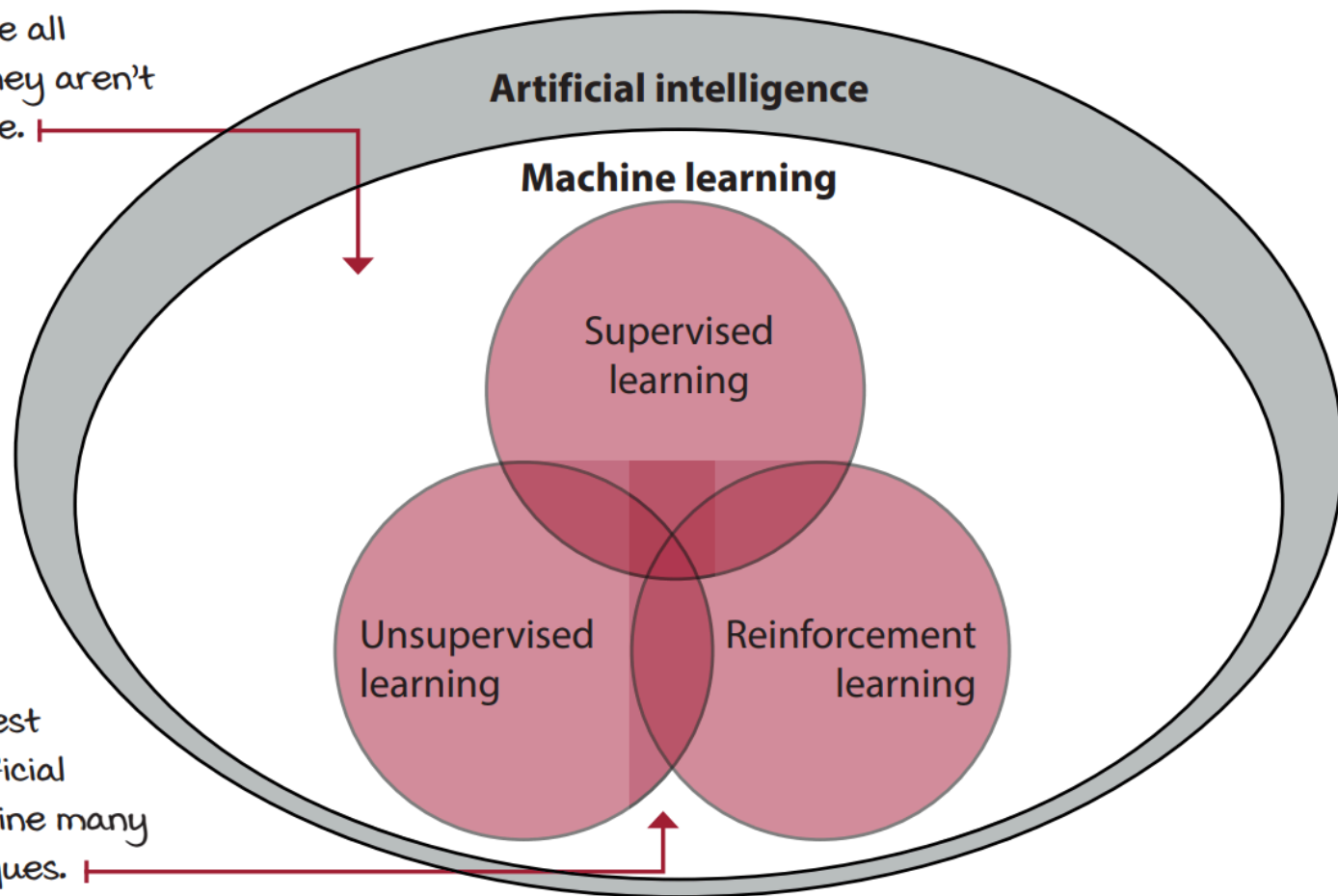
Welding + Optimization

<div><div> Journal Article • 10.1109/access.2023.3277953</div><div><div><input type="checkbox"/></div><div>1. PHH: Policy-Based Hyper-Heuristic With Reinforcement Learning</div></div><div>1 Jan 2023 • IEEE Access</div><div><div>66 3</div><div> PDF</div><div> Summary</div><div> Podcast</div><div> Chat</div><div> 66</div></div></div>	<p>The paper evaluates a policy-based hyper-heuristic framework using reinforcement learning to solve the vehicle routing problem, demonstrating superior performance compared to traditional meta-heuristic and hyper-heuristic methods, particularly in handling unforeseen problem instances and optimizing workflow scheduling.</p>
<div><div>Journal Article • 10.1109/tnnls.2024.3371781</div><div><div><input type="checkbox"/></div><div>2. Deep Reinforcement Learning for Solving Vehicle Routing Problems With Backhauls.</div></div><div>Conghui Wang, Zhiguang Cao, Yaoxin Wu +2 more</div><div>29 Mar 2024 • IEEE transactions on neural networks and learning systems</div><div><div>66 1</div><div> Request PDF</div><div> Podcast</div><div> Chat</div><div> 66</div></div></div>	<p>The paper proposes a neural heuristic using deep reinforcement learning to solve vehicle routing problems with backhauls (VRPBs). It employs an encoder-decoder structured policy network to construct routes, demonstrating effective performance against conventional and neural heuristic baselines.</p>
<div><div>Journal Article • 10.1016/j.eswa.2022.118812</div><div><div><input type="checkbox"/></div><div>3. A reinforcement learning-Variable neighborhood search method for the capacitated Vehicle Routing Problem</div></div><div>Panagiotis Kalatzantonakis, Angelo Sifaleras, Nikolaos Samaras</div><div>1 Sep 2022 • Expert systems with applications</div><div><div>66 37</div><div> Request PDF</div><div> Podcast</div><div> Chat</div><div> 66</div></div></div>	<p>The paper introduces Bandit VNS, a reinforcement learning hyperheuristic for the Capacitated Vehicle Routing Problem, enhancing General Variable Neighborhood Search with adaptive neighborhood selection strategies. It demonstrates over 25% improvement in solution quality compared to conventional methods on benchmark instances.</p>

Where is RL in ML?

Main branches of machine learning

(1) These types of machine learning tasks are all important, and they aren't mutually exclusive.



(2) In fact, the best examples of artificial intelligence combine many different techniques.

Supervised Learning

- We know *all* the right answers (label)
- We teach machine.

Unsupervised Learning

- We don't know the answer.
- We let machine find structure in the data.

Reinforcement Learning

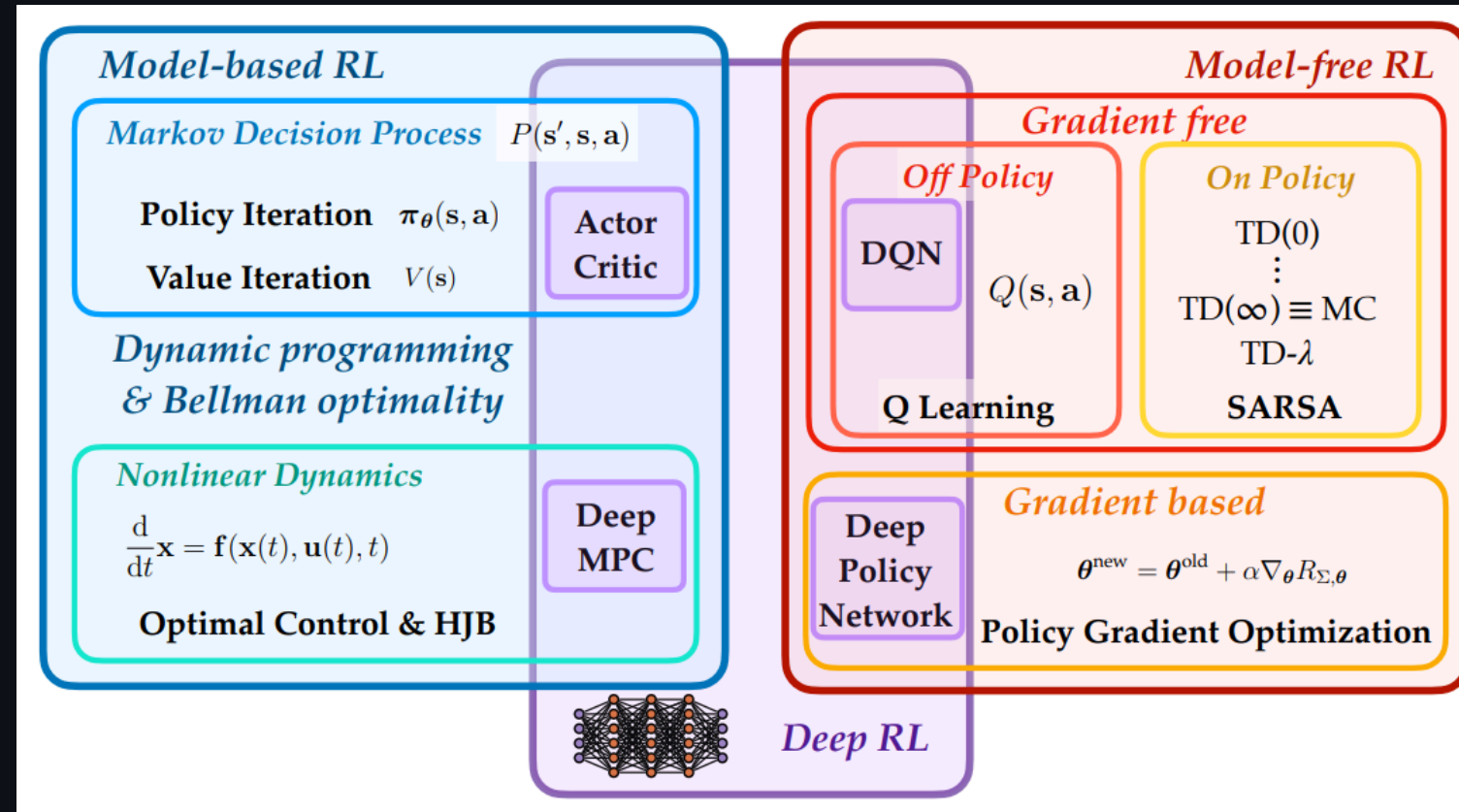
- We don't know *all* the right answer
 - but we have a way to conduct *trial-and-error* experiments.
- We let the machine *discover* the answers.

Applications

- ChatGPT
 - Enhanced by reinforcement learning through a technique called Reinforcement Learning from Human Feedback (RLHF). [\[1\]](#) [\[2\]](#)
- Spot
 - Utilize reinforcement learning (RL) to enhance their locomotion and manipulation capabilities. [\[3\]](#)

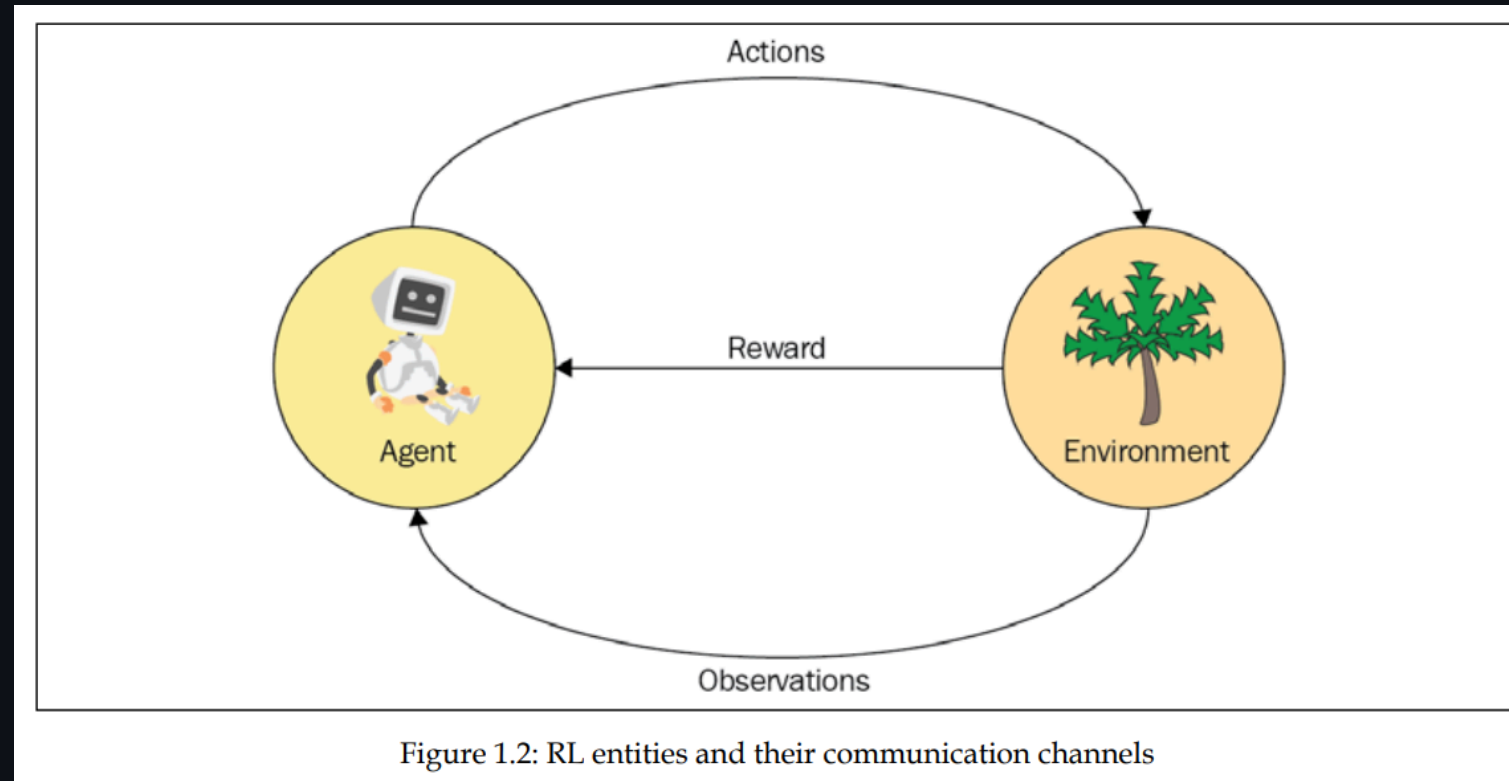
Types of RL

- Don't worry. We will come back later.



RL Formalism

- Entities
 - Agent
 - Environment
- Communication
 - Actions
 - Reward
 - Observation



Agent

- An agent is somebody or something that interact with the environment.
- *The thing that is going to solve our problem.*

Reward

- A scalar value we obtain periodically from the environment.
 - Can be positive or negative
- Tell our agent how well it has behaved.
- An agent wants to get the largest *accumulated* reward over its sequence of actions.

Environment

- The environment is everything outside of an agent.
- The agent's communication with the environment is limited to
 - Reward (obtained from the environment)
 - Actions (executed by the agent and given to the environment)
 - Observations (some information besides the reward that the agent receives from the environment).

Action

- Actions are things that an agent can do in the environment.
- Two types of actions
 - **Discrete actions** form the finite set of mutually exclusive things an agent can do, such as move left or right.
 - **Continuous actions** have some value attached to them, such as a car's action turn the wheel having an angle and direction of steering.

Observation

- Observations are pieces of information that the environment provides the agent with that say what's going on around the agent.
- *I am guessing it is something that agent can use to make action?*

Markov Processes (MP)

- Also called a Markov chain
- MP Models a system observed through a sequence of states .
 - You cannot influence the system, can only watch.

MP - Markov Property

- The future state depends only on the current state, not on the full history.
 - The current state is enough to predict the future.
- If you think you need history, you can add more quantities to the current state (e.g. adding velocity and acceleration, in addition to position, to model motion)

MP - Example (Weather Model)

- States: {sunny, rainy}
- Sequence example: [sunny, sunny, rainy, sunny, ...]
- The Markov property means the probability of rain tomorrow depends only on today's weather, not previous days.
 - To improve this we can include season with weather states.

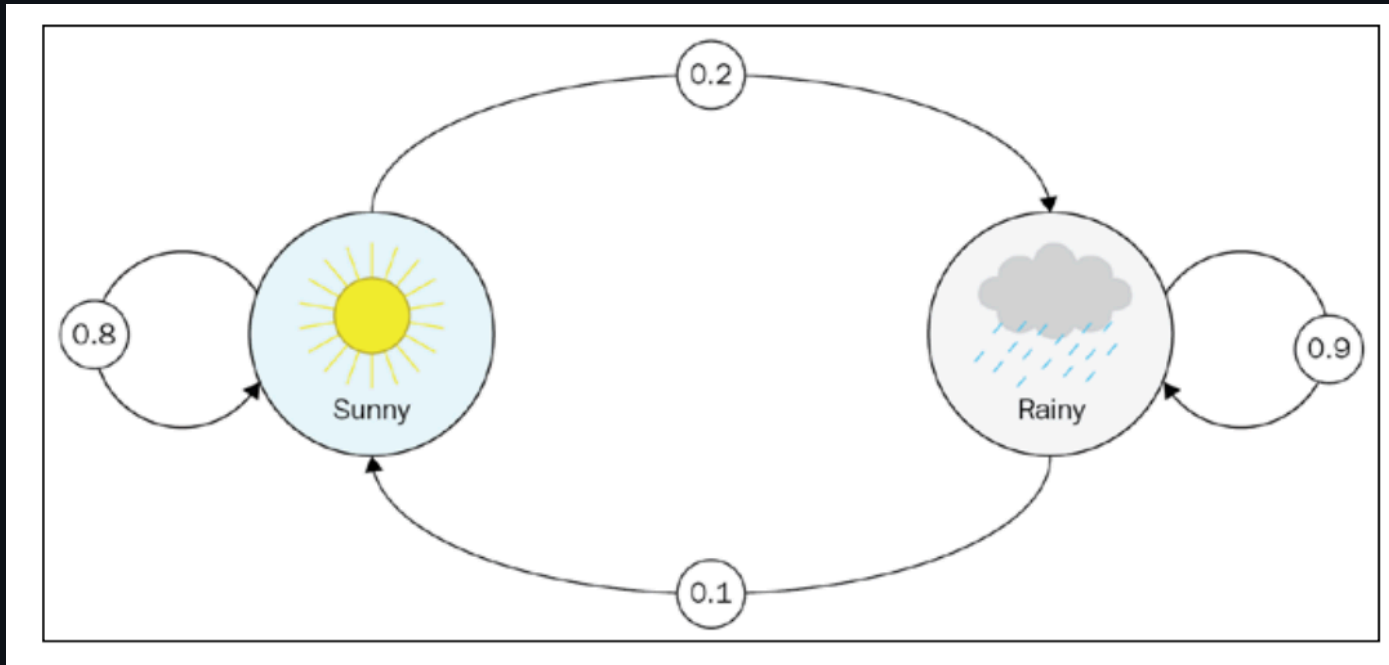
MP - Example (Weather Model)

- We can represent the probability of transitioning from state i to state j using the **transition matrix**.

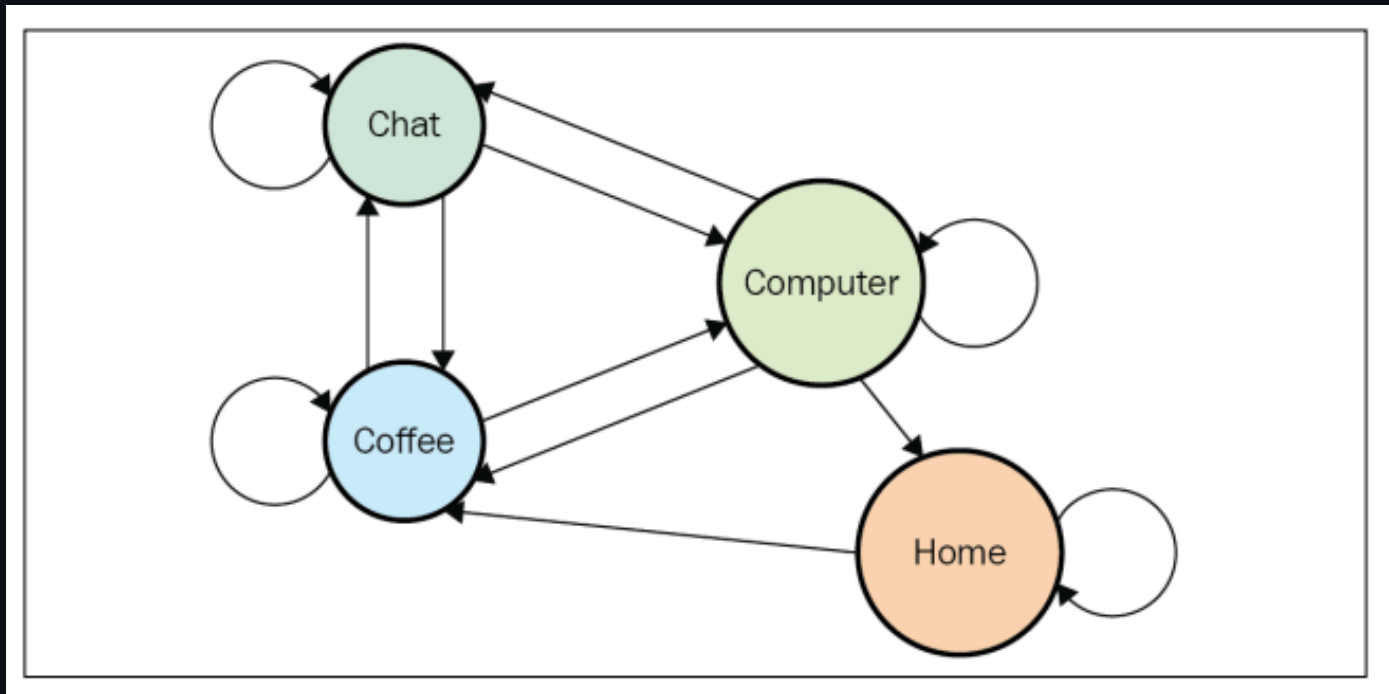
	Sunny	Rainy
Sunny	0.8	0.2
Rainy	0.1	0.9

MP - Example (Weather Model)

- Visual representation



MP - Example (Office Worker Model)



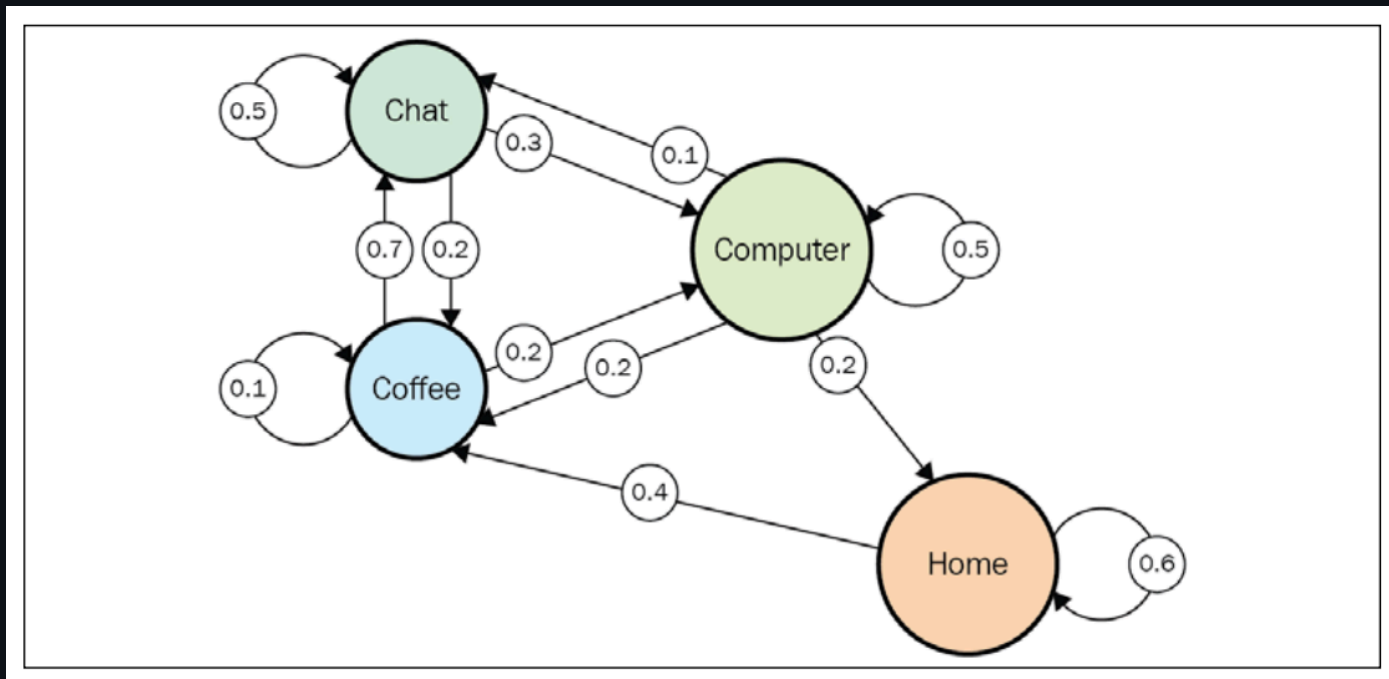
MP - Example (Office Worker Model)

- Transition matrix

From \ To	Home	Coffee	Chat	Computer
Home	60%	40%	0%	0%
Coffee	0%	10%	70%	20%
Chat	0%	20%	50%	30%
Computer	20%	20%	10%	50%

MP - Example (Office Worker Model)

- Visual representation



Estimating the transition matrix

- In real life, we don't know the transition matrix.
- Instead, we estimate transition matrix from **episodes** (sequences of states).
 - Count all observed transitions from each state to every other state.
 - Normalize these counts so that the probabilities from each state sum to 1.
 - With more episodes, our estimation improves.

Markov Reward Processes (MRP)

- We extend MP by associating a reward value with each state transition.
- For each **episode**, the return at time t (denoted as G_t) is the sum of future rewards, discounted by γ at each step:

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$$

- where γ is a scalar value between 0 and 1 called a **discount factor**.

MRP - Discount Factor

- γ determines how much future rewards are valued compared to immediate rewards.
 - $\gamma = 1$
 - The agent values all future rewards equally. This represents perfect foresight.
 - $\gamma = 0$
 - The agent only considers the immediate reward, ignoring all future rewards—total short-sightedness.

MRP

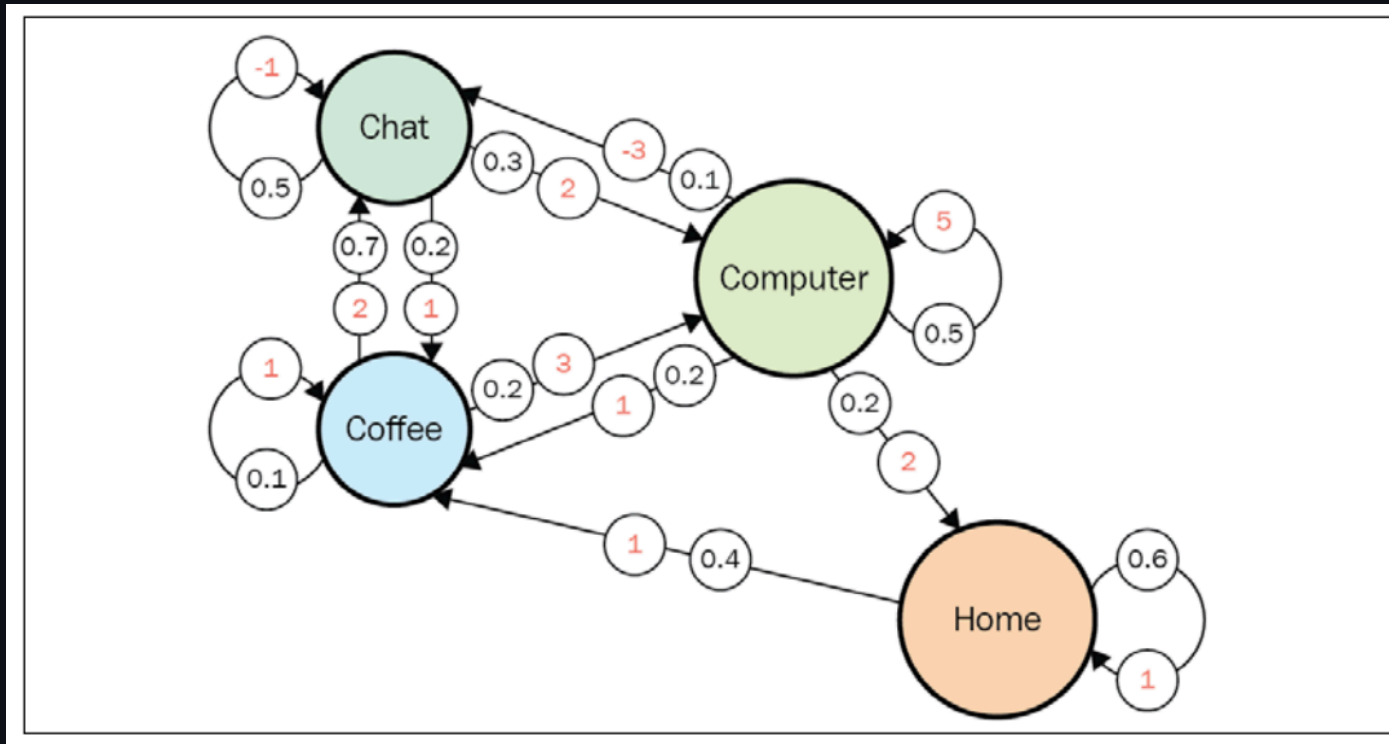
- Recall that state transition is probabilistic.
 - G_t can vary even for the same state.
- We want to know the **expected** return instead

$$v(s) = \mathbb{E}[G_t | S_t = s]$$

- Think about averaging return from many episodes.
- v is also called **value function**.

Practical example of $v(s)$

- Let $\gamma = 0$, calculate $v(s)$



Practical example of $v(s)$

$$v(chat) = -1 \times 0.5 + 2 \times 0.3 + 1 \times 0.2 = 0.3$$

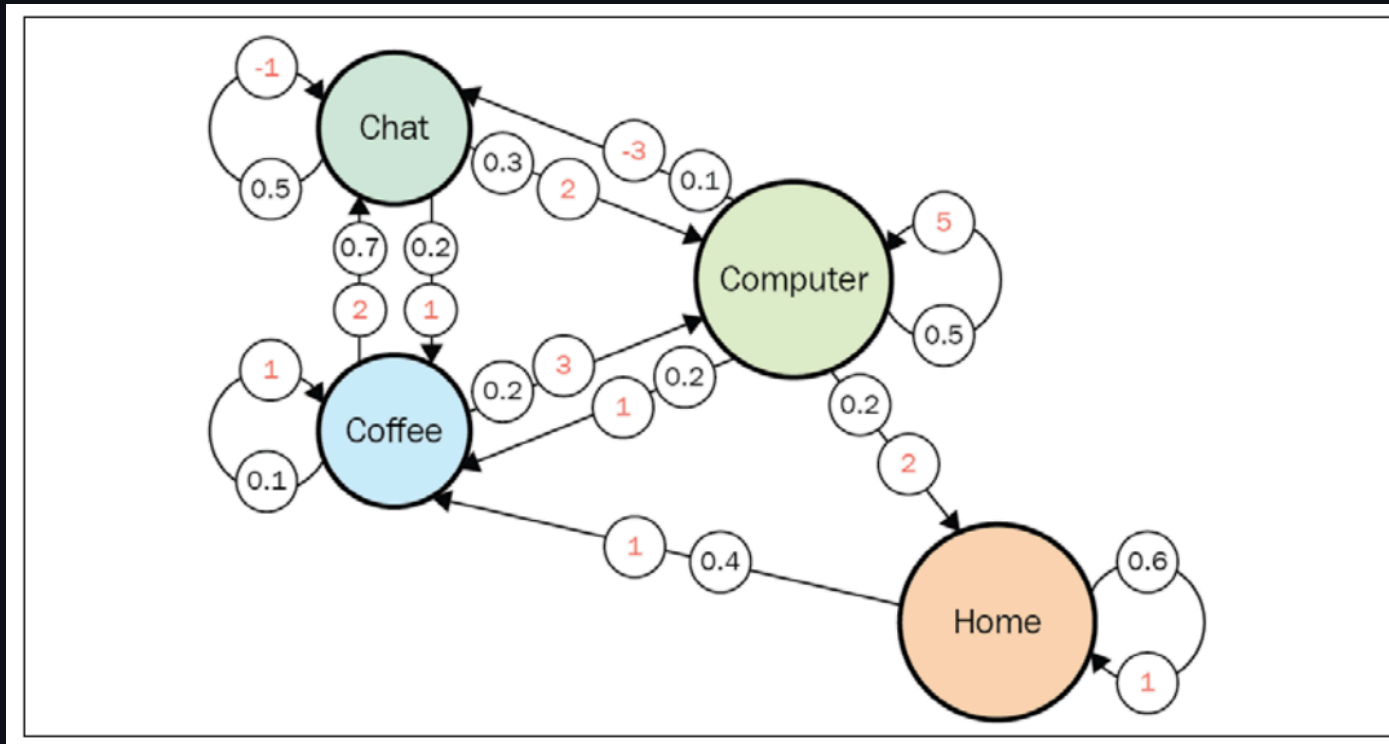
$$v(coffee) = 2 \times 0.7 + 1 \times 0.1 + 3 \times 0.2 = 2.1$$

$$v(home) = 1 \times 0.6 + 1 \times 0.4 = 1.0$$

$$v(computer) = 5 \times 0.5 + (-3) \times 0.1 + 1 \times 0.2 + 2 \times 0.2 = 2.8$$

- Computer is the most valuable state to be in.

Practical example of $v(s)$

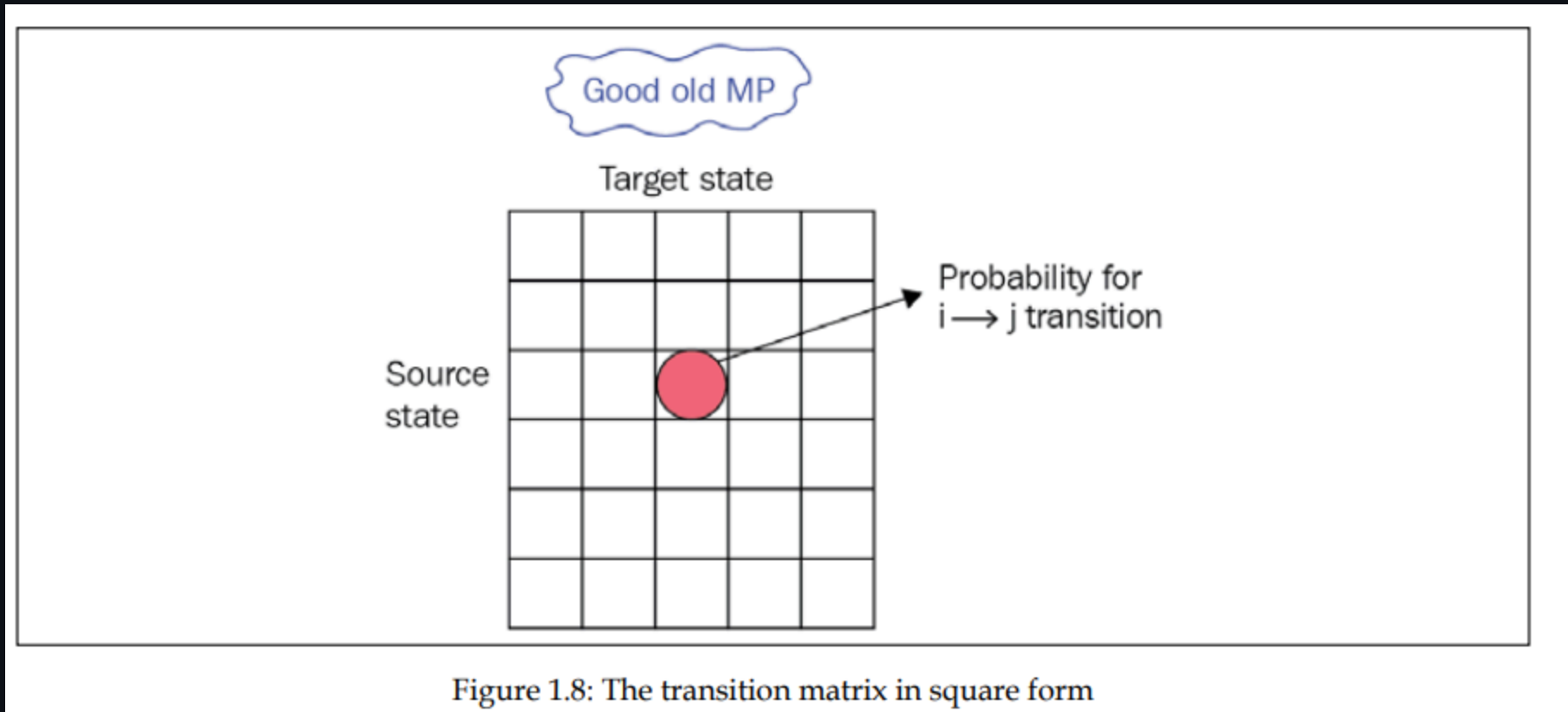


- If $\gamma = 1$, then $v(s) = \infty$
- This is why we usually introduce $\gamma < 1$ in MRP.

Markov Decision Process

- Add a set of actions (A)
- Agent can now choose an action to take.
- Our transition matrix will now have "action" dimension.

Before: MP



After: MDP

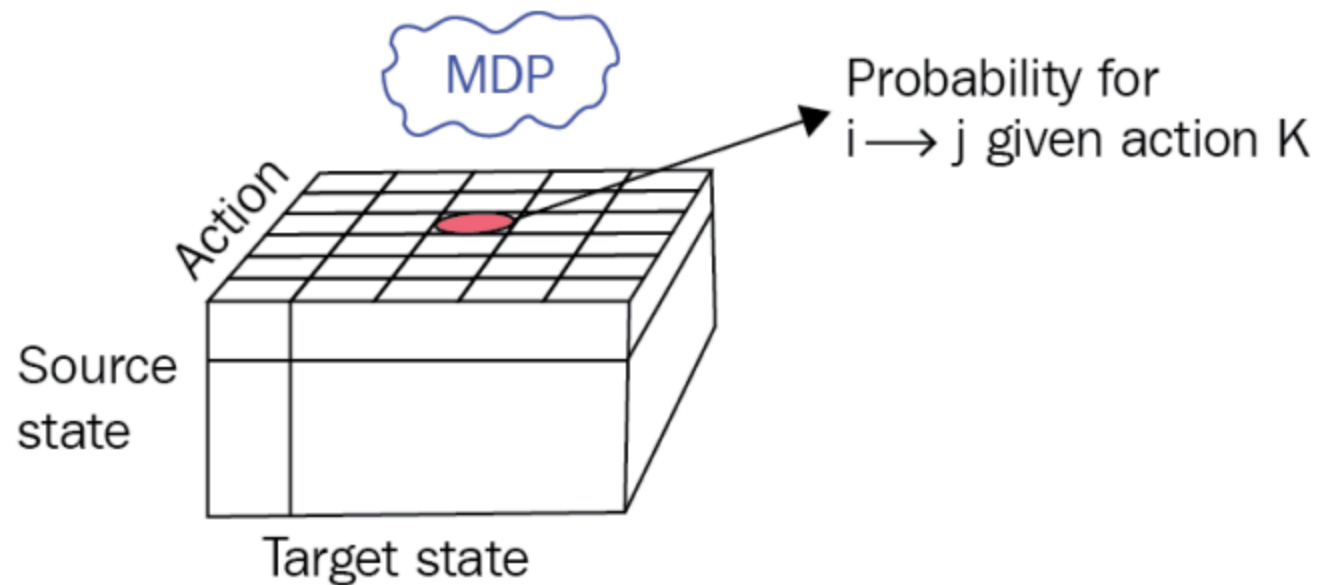
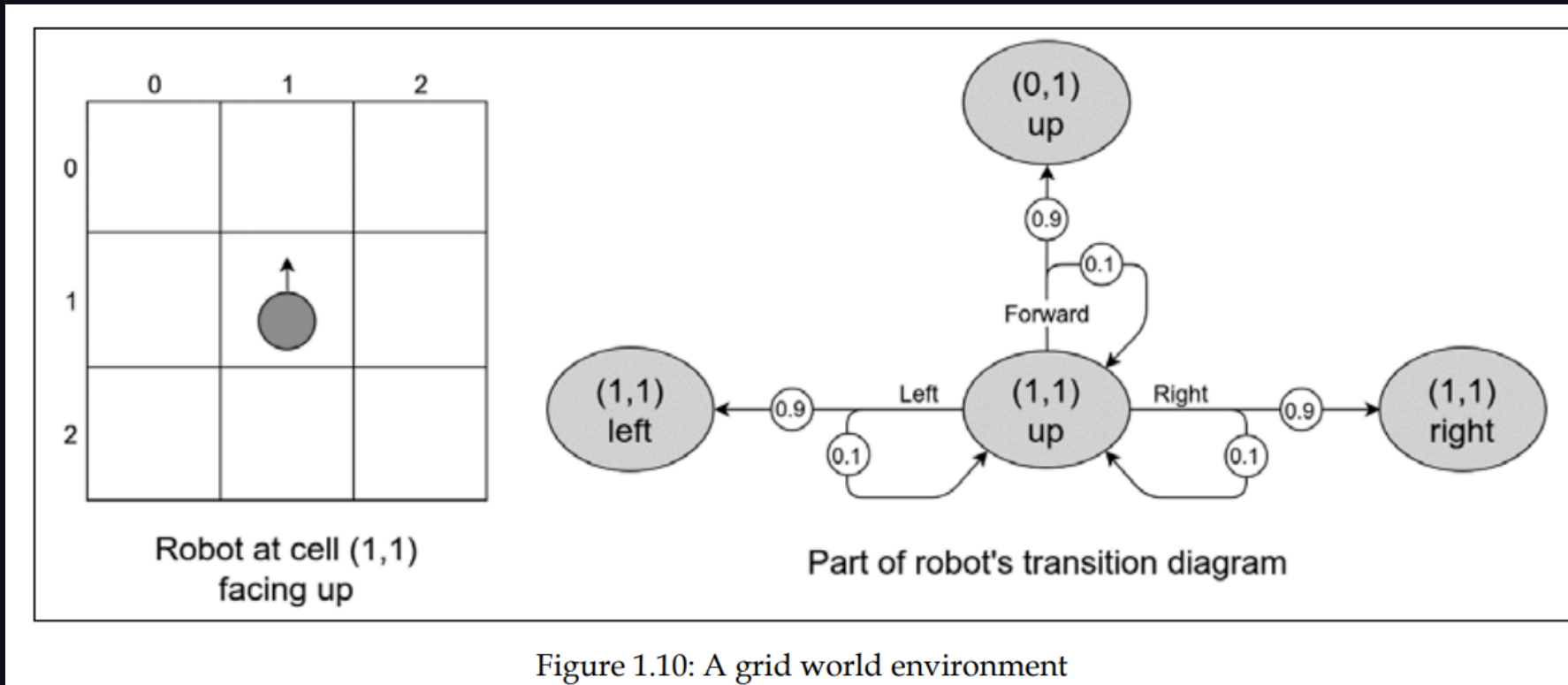


Figure 1.9: Transition probabilities for the MDP

MDP: Example (Grid World)



MDP: Example (Grid World)

- State = [Robot Position] + [Orientation]
- Action = `turn left`, `turn right`, `go forward`
- Even if robot execute action (e.g. `up`), there is a chance that the robot stay the same (due to motor imperfection).

Policy

- Rule that control agent behavior.
- In the robot example, policies can be
 - Always turn right when the state is $(0, 1, \text{left})$. (**Deterministic**)
 - turn right 50% of the time and go forward 50% of the time when the state is $(0, 0, \text{up})$. (**Stochastic**)
 - Randomly move 10% of the time, but for the rest of the time, turn right when the state is $(0, 1, \text{left})$ (**Stochastic / Explorative**)

Policy

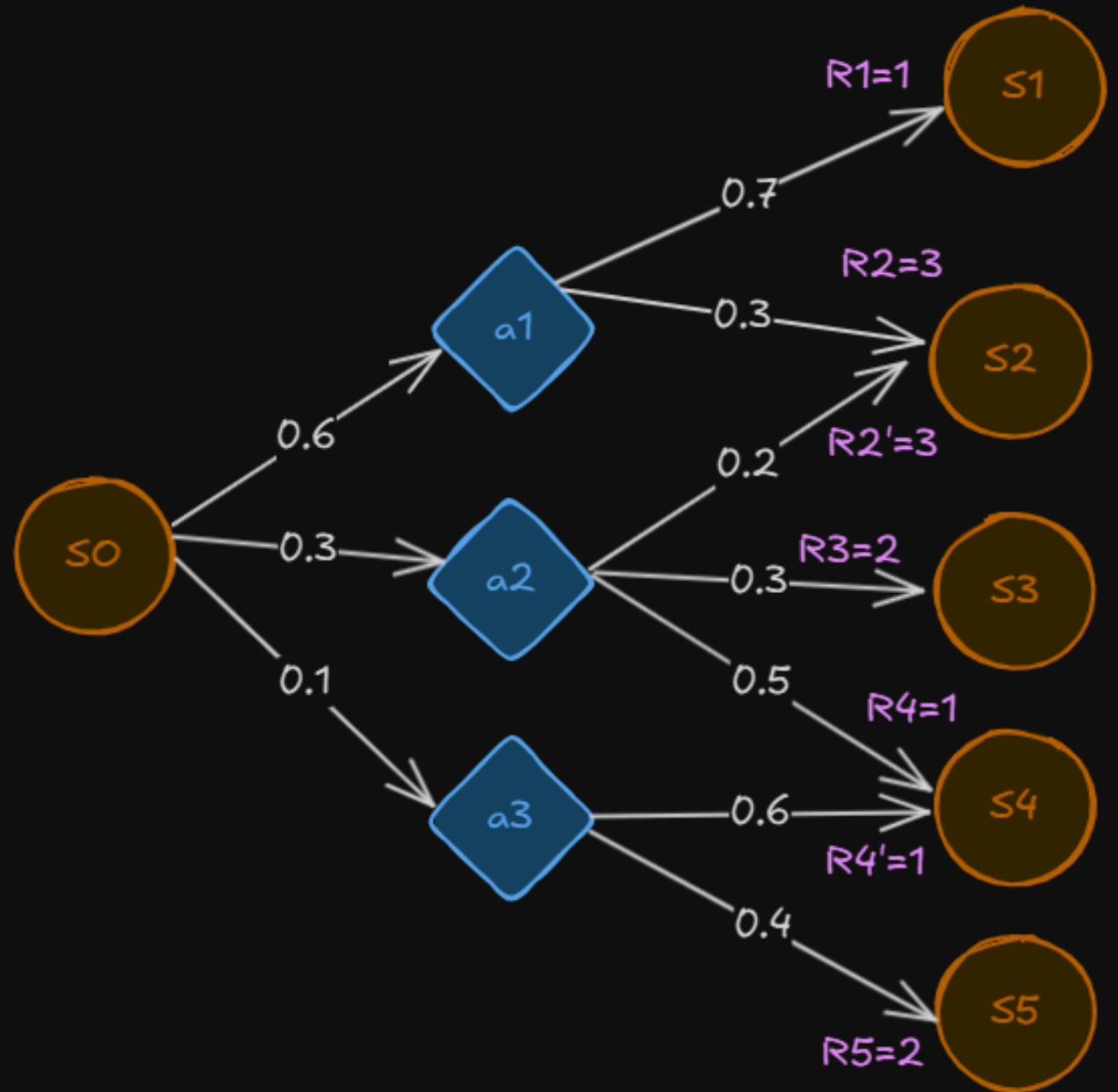
- Formally

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

- Policy can be deterministic and stochastic.
- If policy is not changing, MDP becomes MRP.

MDP Example

- Stochastic policy
- Given $\gamma = 0$
- Calculate $v(S_0)$



Solution

$$\begin{aligned}v(S_0) &= 0.6 \times (0.7R_1 + 0.3R_2) + 0.3 \times (0.2R'_2 + 0.3R_3 + 0.5R_4) \\&\quad + 0.1 \times (0.6R'_4 + 0.4R_5) \\&= 0.6 \times (0.7 + 0.9) + 0.3 \times (0.6 + 0.6 + 0.5) + 0.1 \times (0.6 + 0.8) \\&= 0.6 \times (1.6) + 0.3 \times (1.7) + 0.1 \times (1.4) \\&= 1.61\end{aligned}$$

Recap

- Our policy is the choose a_1, a_2, a_3 with [60%, 30%, 10%] chance
 - We get $v = 1.61$
- Can you do better?

Reformulation

Recall

$$v(S_0) = 0.6 \times (0.7R_1 + 0.3R_2) + 0.3 \times (0.2R'_2 + 0.3R_3 + 0.5R_4) \\ + 0.1 \times (0.6R'_4 + 0.4R_5)$$

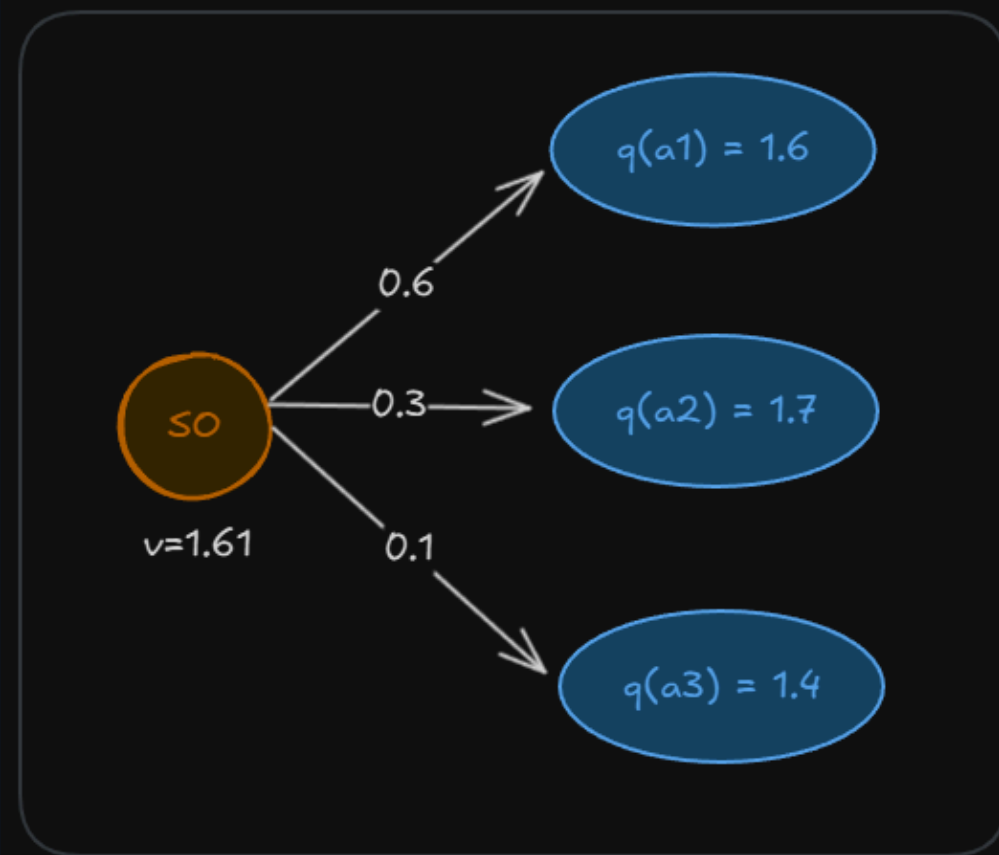
We can write

$$v(S_0) = 0.6 \times [\text{Reward from } a_1] + 0.3 \times [\text{Reward from } a_2] \\ + 0.1 \times [\text{Reward from } a_3] \\ = 0.6 \times q(a_1) + 0.3 \times q(a_2) + 0.1 \times q(a_3)$$

where $q(a_1) = 1.6$, $q(a_2) = 1.7$, and $q(a_3) = 1.4$.

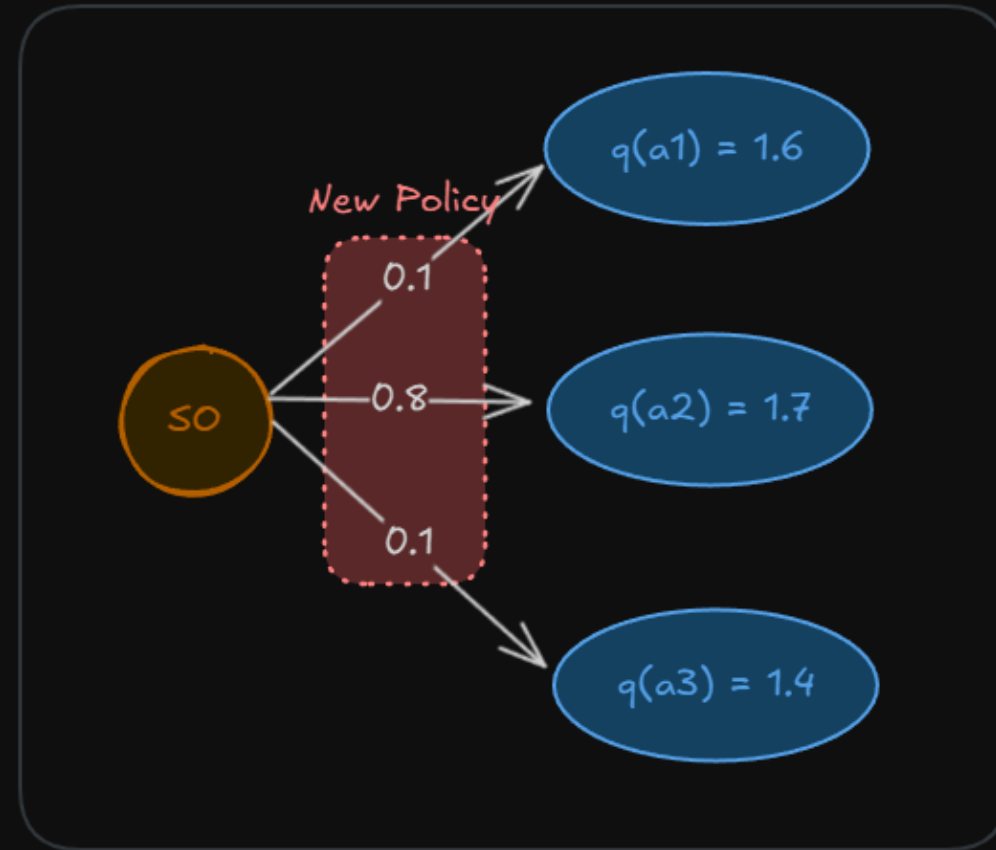
Reformulation

- We can then rewrite the diagram like this.
- q is called **action-value functions**.
 - It tells you how good the action is.



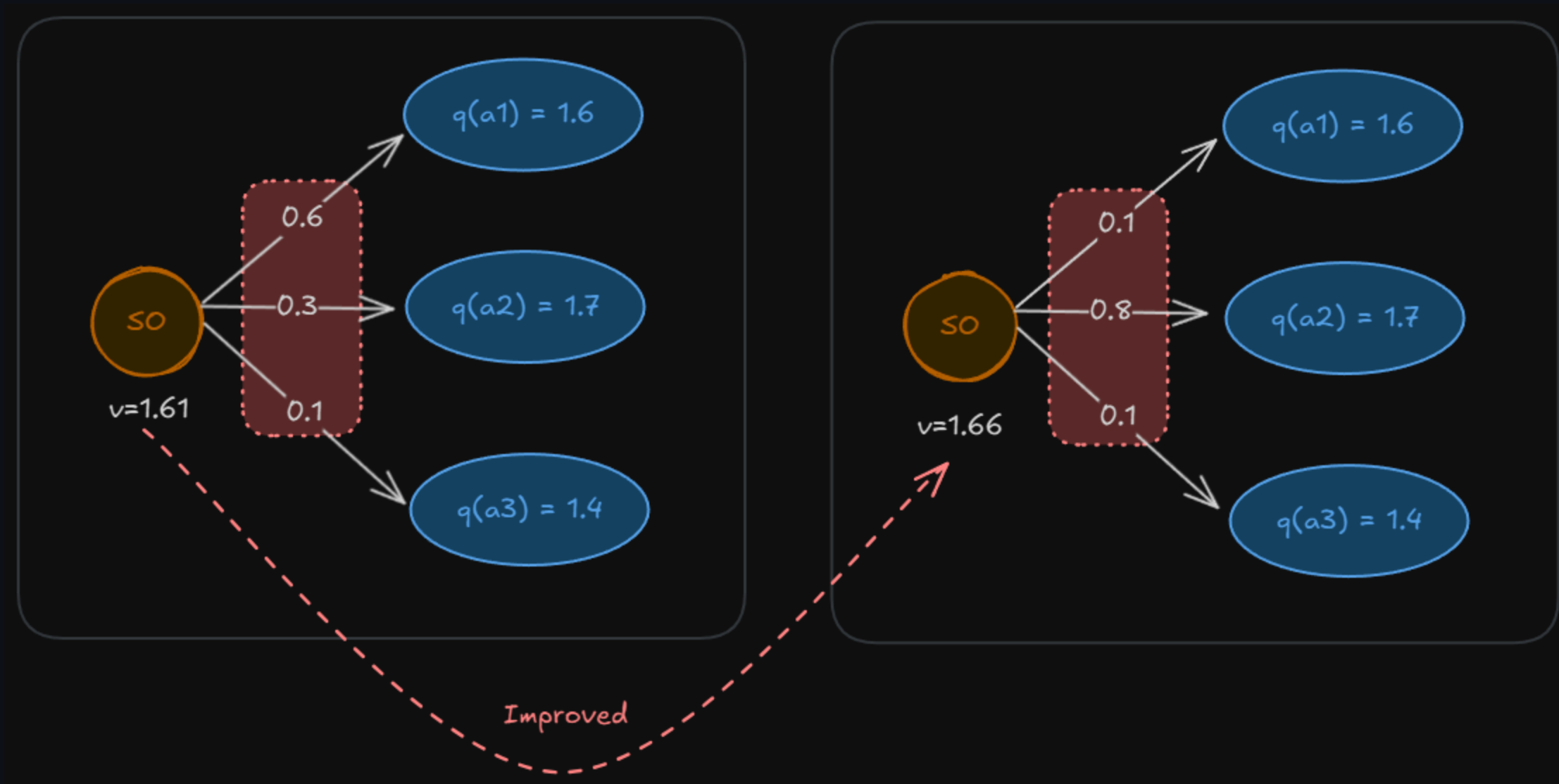
Policy Improvement

- Let's try a new policy.
- What is v ?



Policy Improvement

$$\begin{aligned} v(S_0) &= 0.1 \times q(a_1) + 0.8 \times q(a_2) + 0.1 \times q(a_3) \\ &= 0.1 \times 1.6 + 0.8 \times 1.7 + 0.1 \times 1.4 \\ &= 1.66 \end{aligned}$$



Optimal Policy

- If we can find a policy that can improve v , that policy must be better than the previous policy.
- The best policy is the policy that **maximize** v .
- Can you find the **optimal** policy in our example?

Optimal Policy

- The optimal policy is just choose action a_2 all the time.
 - Essentially, this is acting greedily.
- We can write more formally

$$\pi^* = \operatorname{argmax}_a q(a)$$

Discussion

- Notice that the optimum policy is deterministic.
- In fact, in most cases you will find that most environment that can be modelled by a MDP has an optimal policy which is deterministic [\[Ref\]](#).
- However, if the environment is adversarial and can exploit predictable behavior, adopting a stochastic policy can actually be more advantageous.
 - This environment violates Markov property (non-stationary).