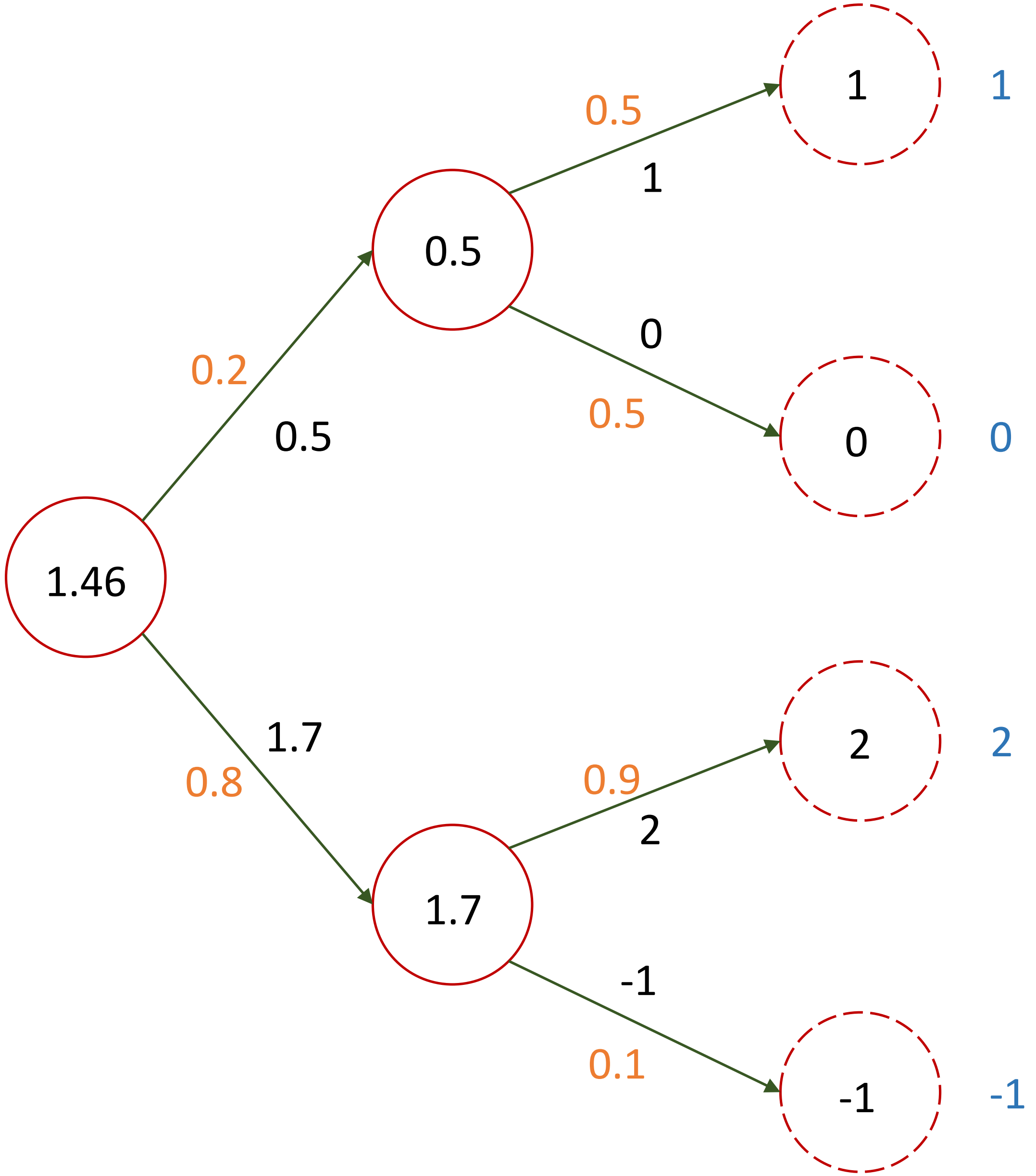
Asynchronous Methods for Deep Reinforcement Learning

Dominik Winkelbauer



State s

Action a

Reward r

Policy π

Value v

Action value q

Value function:

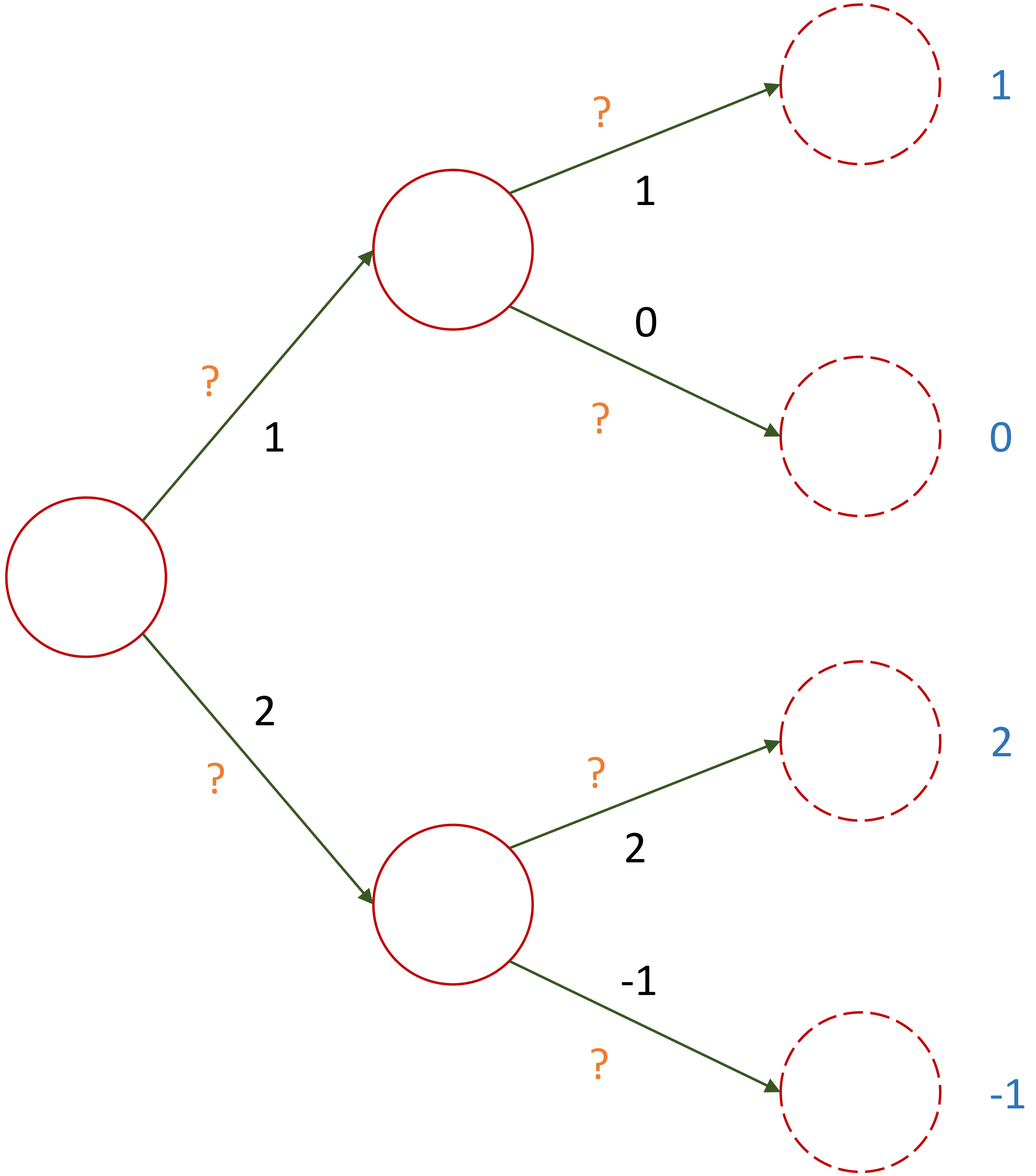
$$V^{\pi}(s) = \mathbb{E}[R_t | s_t = s]$$

Example:

$$\begin{aligned} V^{\pi}(s_t) = {} & 0.8 * 0.1 * (-1) + \\ & 0.8 * 0.9 * 2 + \\ & 0.2 * 0.5 * 0 + \\ & 0.2 * 0.5 * 1 = 1.46 \end{aligned}$$

Action value function:

$$Q^{\pi}(s, a) = \mathbb{E}[R_t | s_t = s, a]$$



State s

Action a

Reward r

Policy π

Value v

Action value q

Value function:

$$V^{\pi}(s) = \mathbb{E}[R_t | s_t = s]$$

Action value function:

$$Q^{\pi}(s, a) = \mathbb{E}[R_t | s_t = s, a]$$

Optimal action value function:

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

=> $Q^{*}(s, a)$ implicitly describes an optimal policy

Value-based algorithms

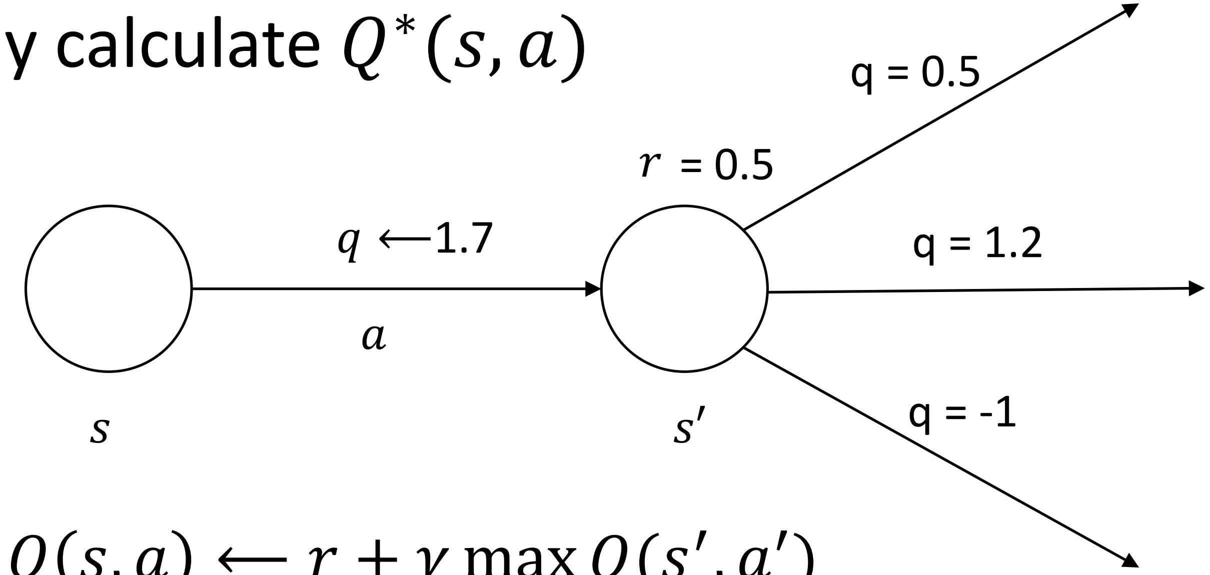
- Try to approximate $V^*(s)$ or $Q^*(s, a)$
- Implicitly learn policy

Policy-based algorithms

- Directly learn policy

Q-Learning

- Try to iteratively calculate $Q^*(s, a)$



$$Q(s,a) \longleftarrow r + \gamma \max_{a\prime} Q(s', a')$$

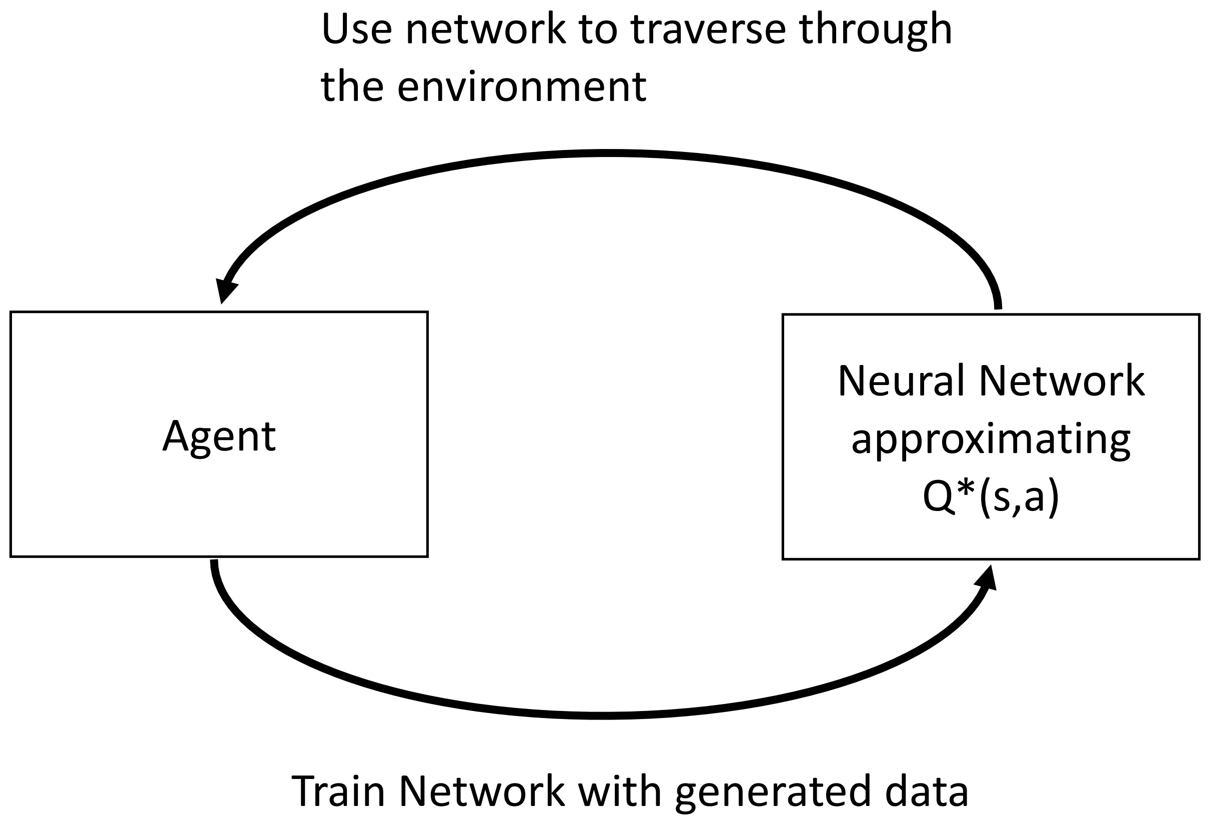
- Idea: Use neural network for approximating Q

$$L(\theta) = \mathbb{E}[r + \gamma \max_{a} Q(s', a'; \theta) - Q(s, a; \theta)]$$

How to traverse through the environment

- We follow an ϵ–greedy policy with $\epsilon \in [0,1]$
- In every state:
  - Sample random number $k \in [0,1]$
  - If $k > \epsilon$ => choose action with maximum q value
  - else => choose random action
- Exploration vs. Exploitation

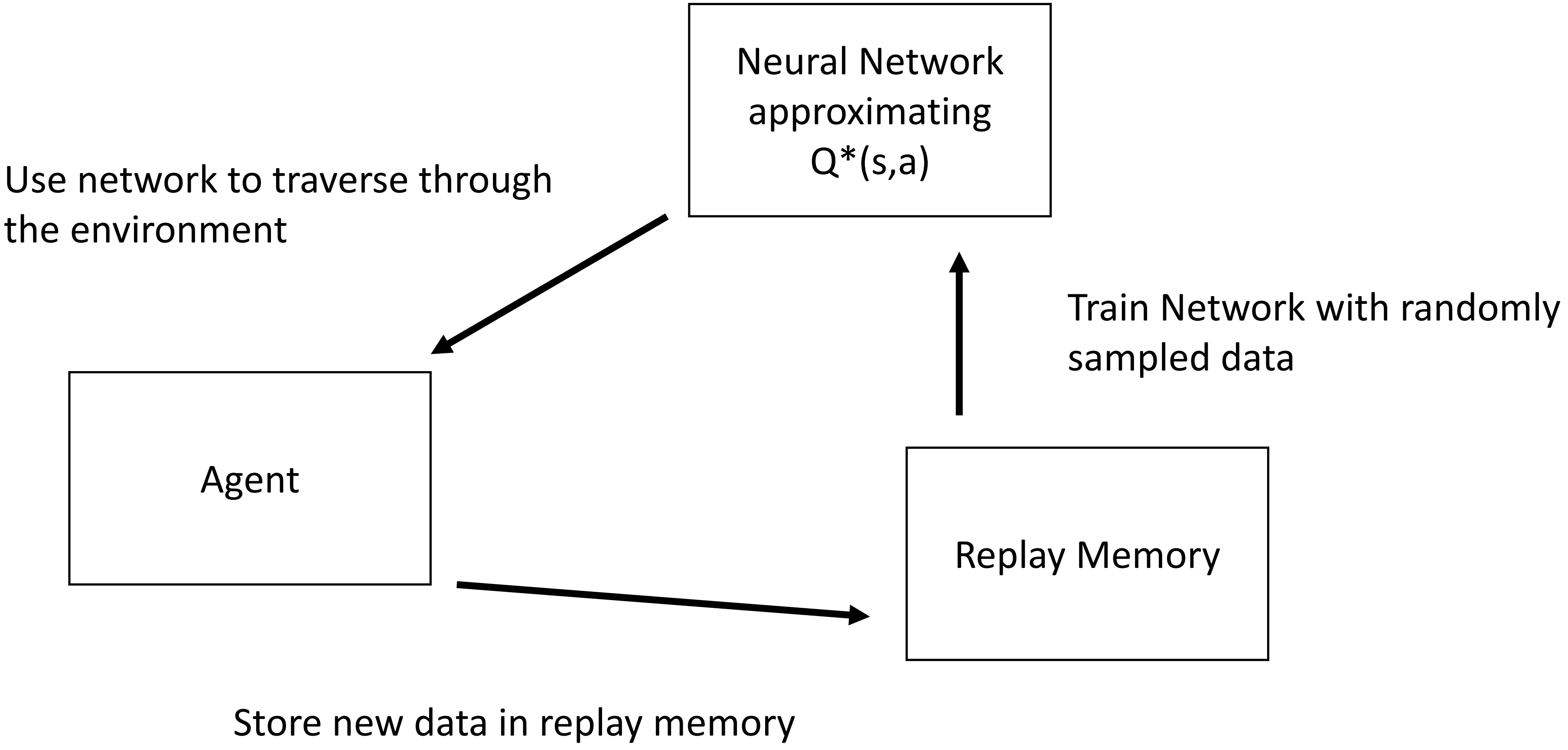
Q-Learning with Neural Networks



=> Data is non-stationary => Training with NN is instable

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan and Riedmiller, Martin

Playing atari with deep reinforcement learning



=> Data is stationary => Training with NN is stable

On-policy vs. off-policy

- On-policy: The data which is used to train our policy, has to be generated using the exact same policy.

  => Example: REINFORCE

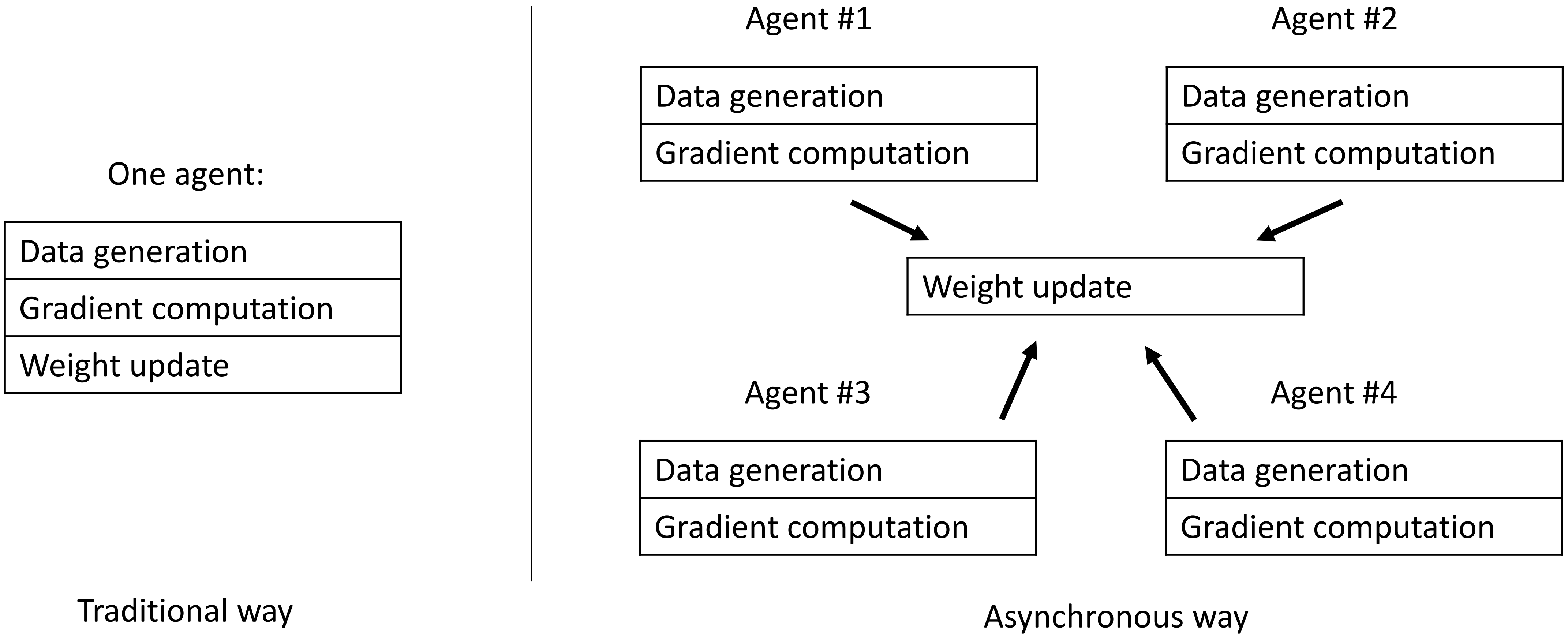
- Off-policy: The data which is used to train our policy, can also be generated using another policy.

  => Example: Q-Learning

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D. & Kavukcuoglu, K. (ICML, 2016)

Asynchronous Methods for Deep RL

- Alternative method to make RL work better together with neural networks



Asynchronous Q-Learning

- Combine Idea with Q-Learning
- Generated data is stationary

=> Training is stable

=> No replay memory necessary

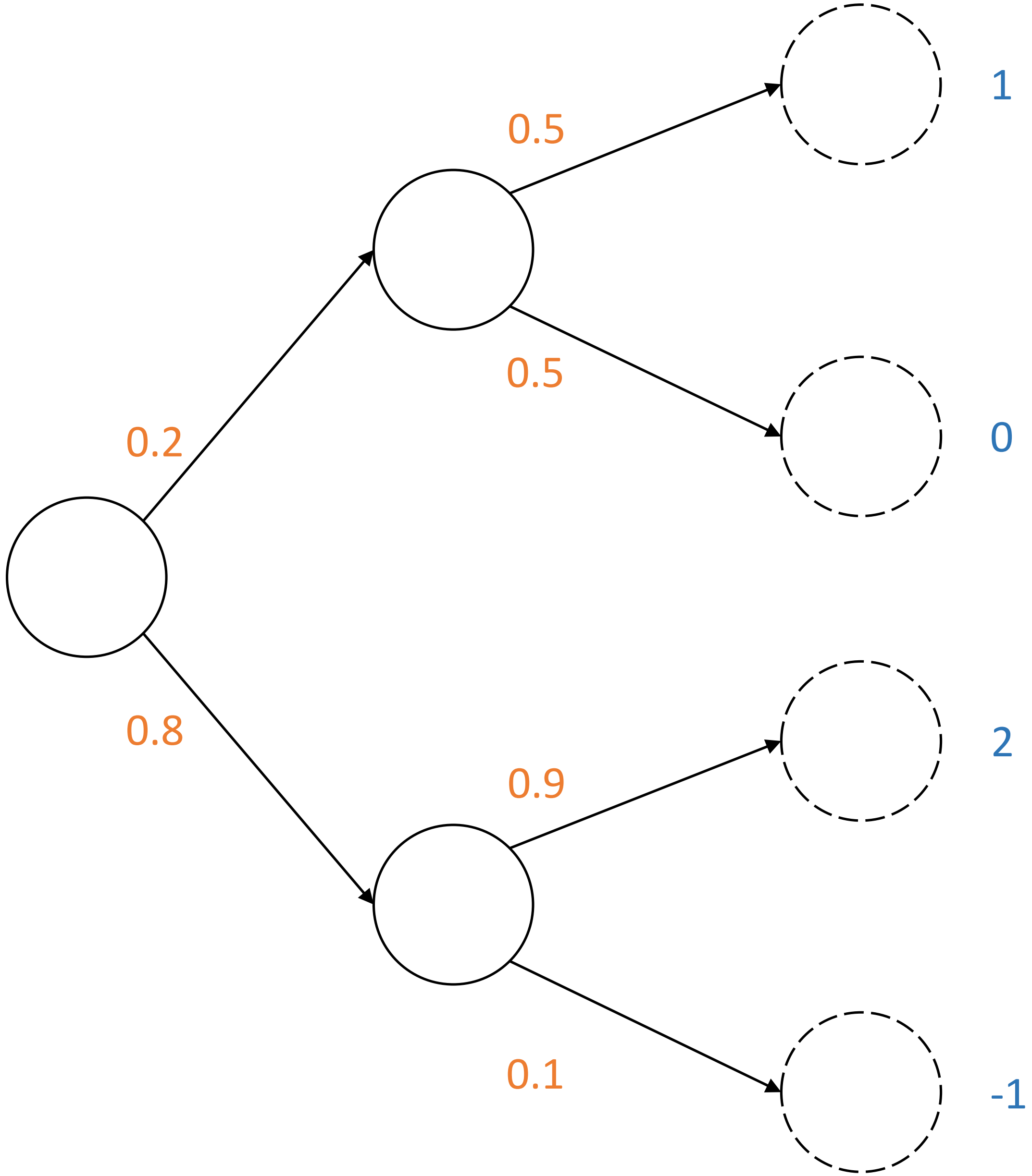
=> Data can be used directly while training is still stable

Value-based algorithms

- Try to approximate $V^*(s)$ or $Q^*(s, a)$
- Implicitly learn policy

Policy-based algorithms

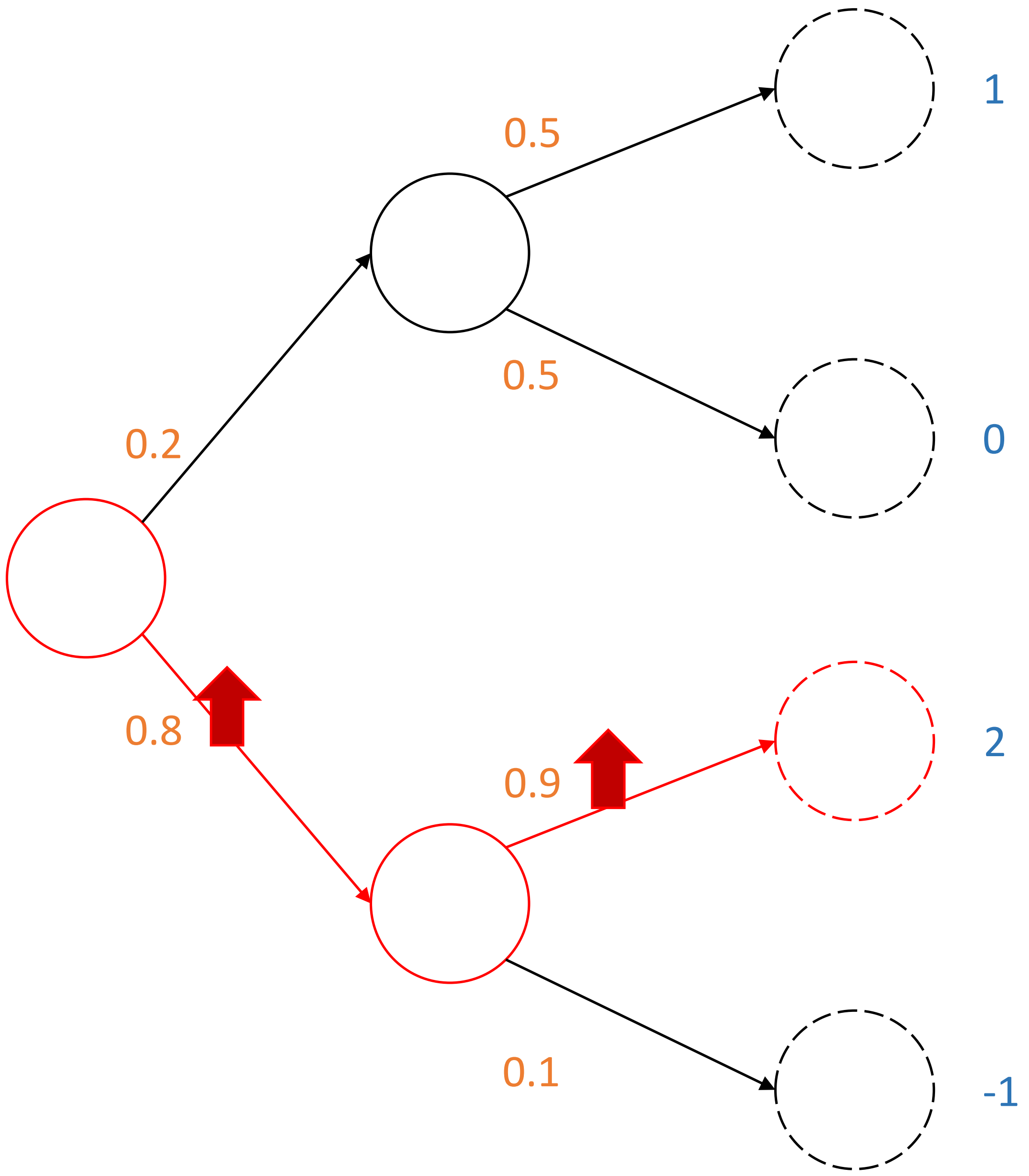
- Directly learn policy



REINFORCE:

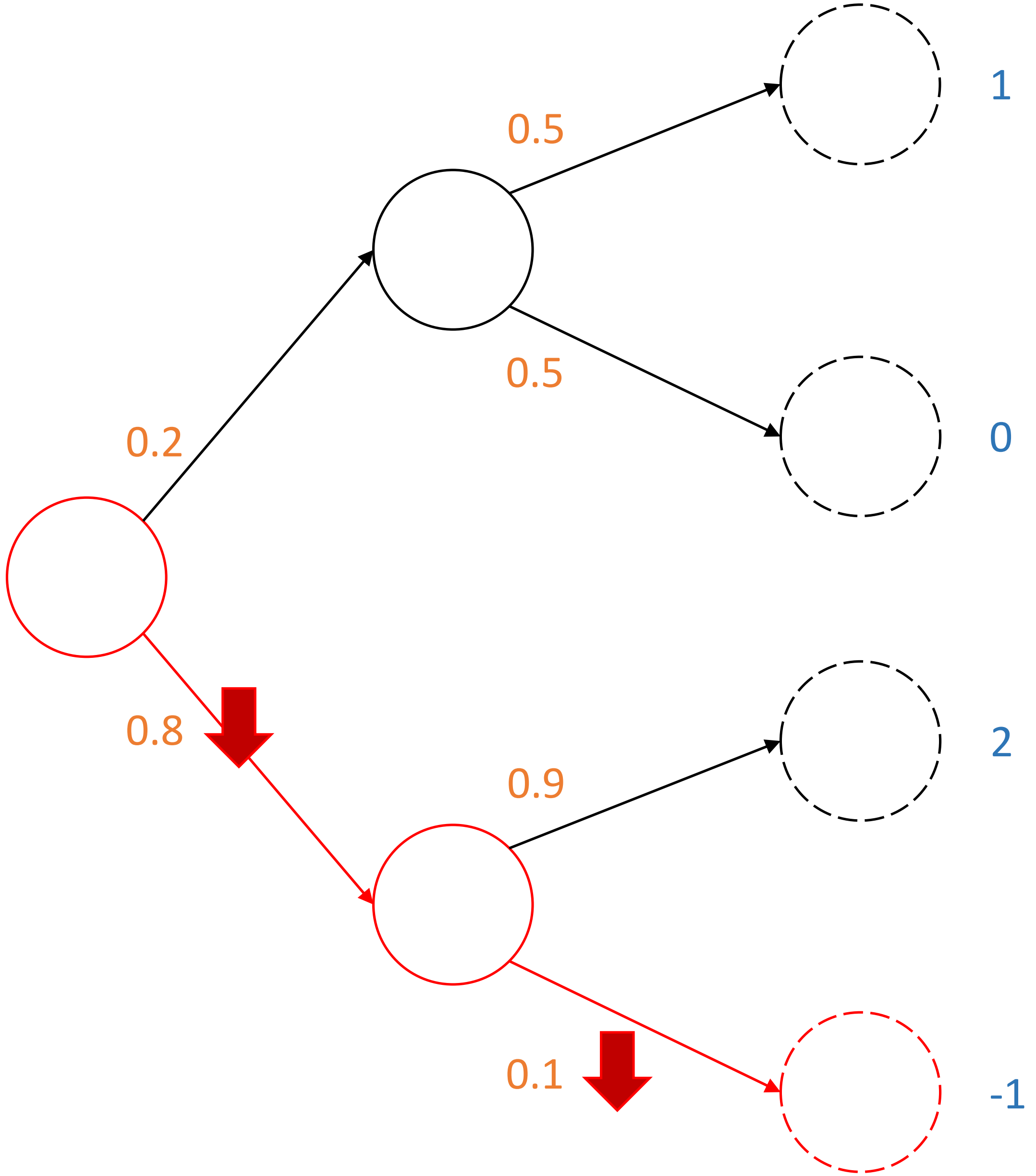
$$\nabla_\theta \log \pi(a_t|s_t,\theta)\ R_t$$

Sample trajectories and enforce actions which lead to high rewards



REINFORCE:

$$\nabla_\theta \log \pi(a_t|s_t,\theta)\, R_t$$

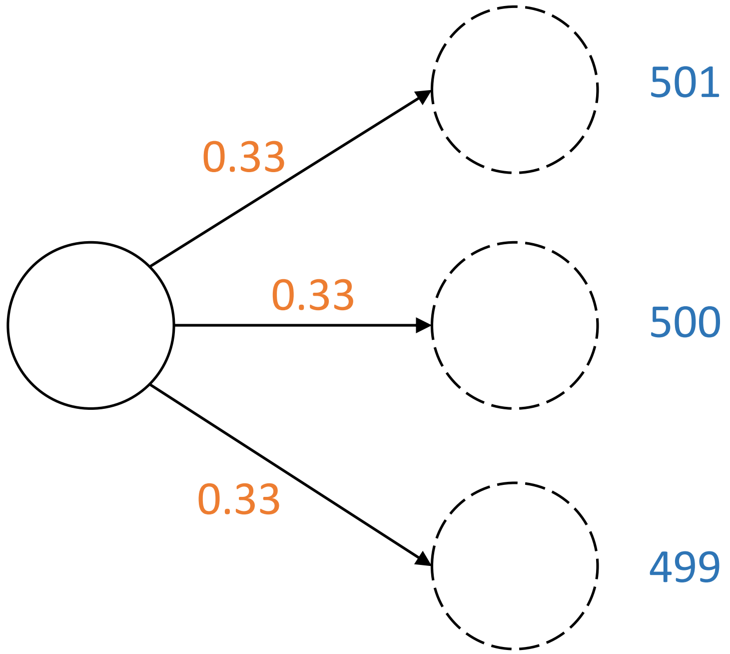


REINFORCE:

$$\nabla_\theta \log \pi(a_t|s_t,\theta)\ R_t$$

Problem: High Variance

$\nabla_{\theta_i} \log \pi(a_t \vert s_t, \theta)$	R_t	$\Delta\theta_i$
0.9	500	450
0.2	501	100,2
-0.3	499	-149,7



REINFORCE:

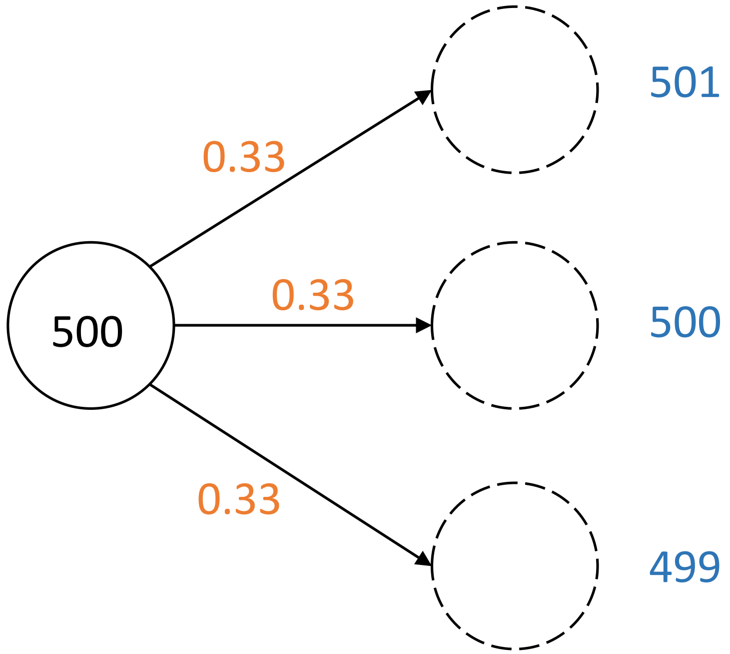
$$\nabla_\theta \log \pi(a_t|s_t,\theta)\, R_t$$

Substract baseline:

$$\nabla_\theta \log \pi(a_t|s_t,\theta)\, (R_t - b_t(s_t))$$

Problem: High Variance

$\nabla_{\theta_i} \log \pi(a_t \mid s_t, \theta)$	A_t	$\Delta\theta_i$
0.9	0	0
0.2	1	0.2
-0.3	-1	0.3



REINFORCE:

$$\nabla_\theta \log \pi(a_t \mid s_t, \theta)\; R_t$$

Substract baseline:

$$\nabla_\theta \log \pi(a_t \mid s_t, \theta)\; (R_t - b_t(s_t))$$

Use value function as baseline:

$$\nabla_\theta \log \pi(a_t \mid s_t, \theta)\; (R_t - V(s_t, \theta_v))$$

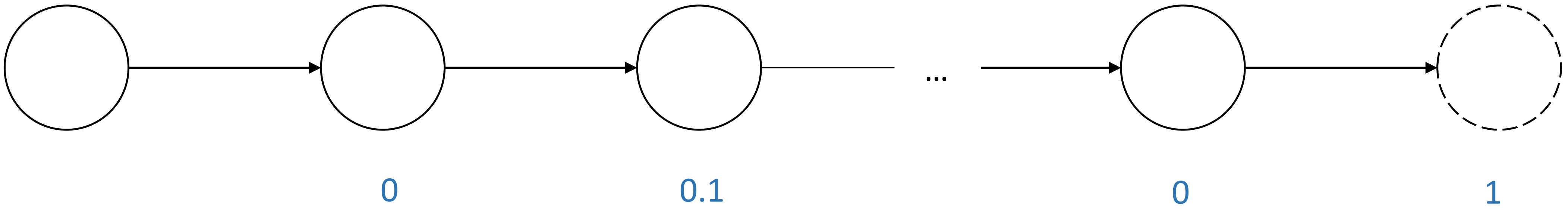
$$(R_t - V(s_t, \theta_v))$$

Can be seen as estimate of advantage:
$A(a_t, s_t) = Q(a_t, s_t) - V(s_t)$

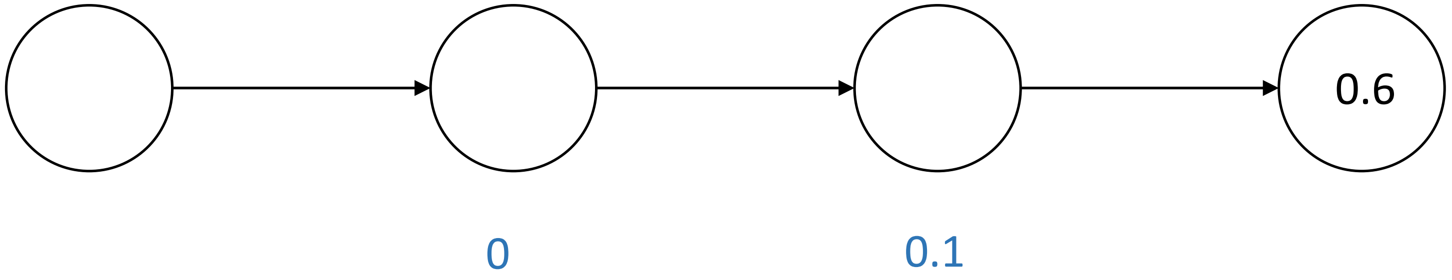
Actor: policy network
Critic: value network

Update interval

REINFORCE



Actor-critic with advantage



Asynchronous advantage actor-critic (A3C)

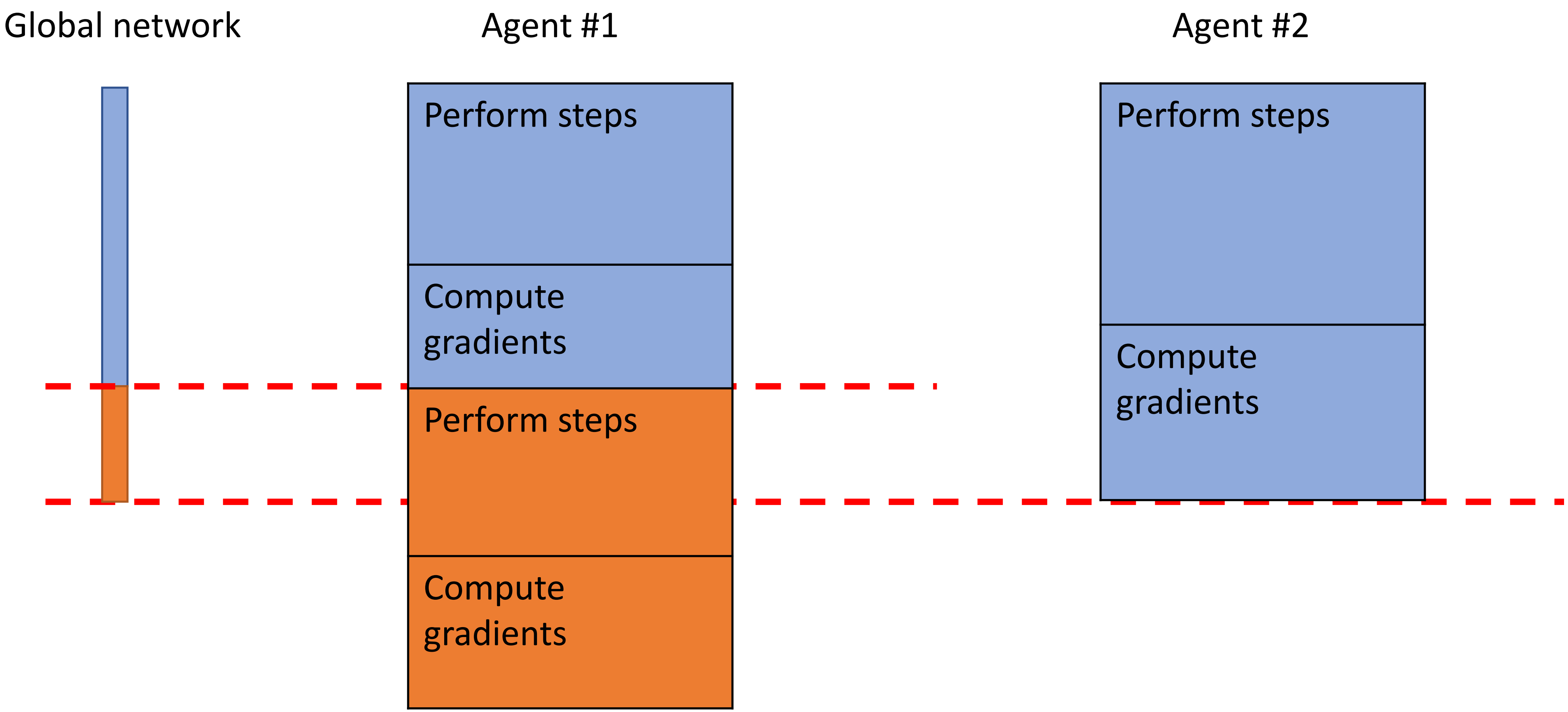
- Update local parameters from global shared parameters

- Explore environment according to policy $\pi(a_t|s_t;\theta)$ for N steps

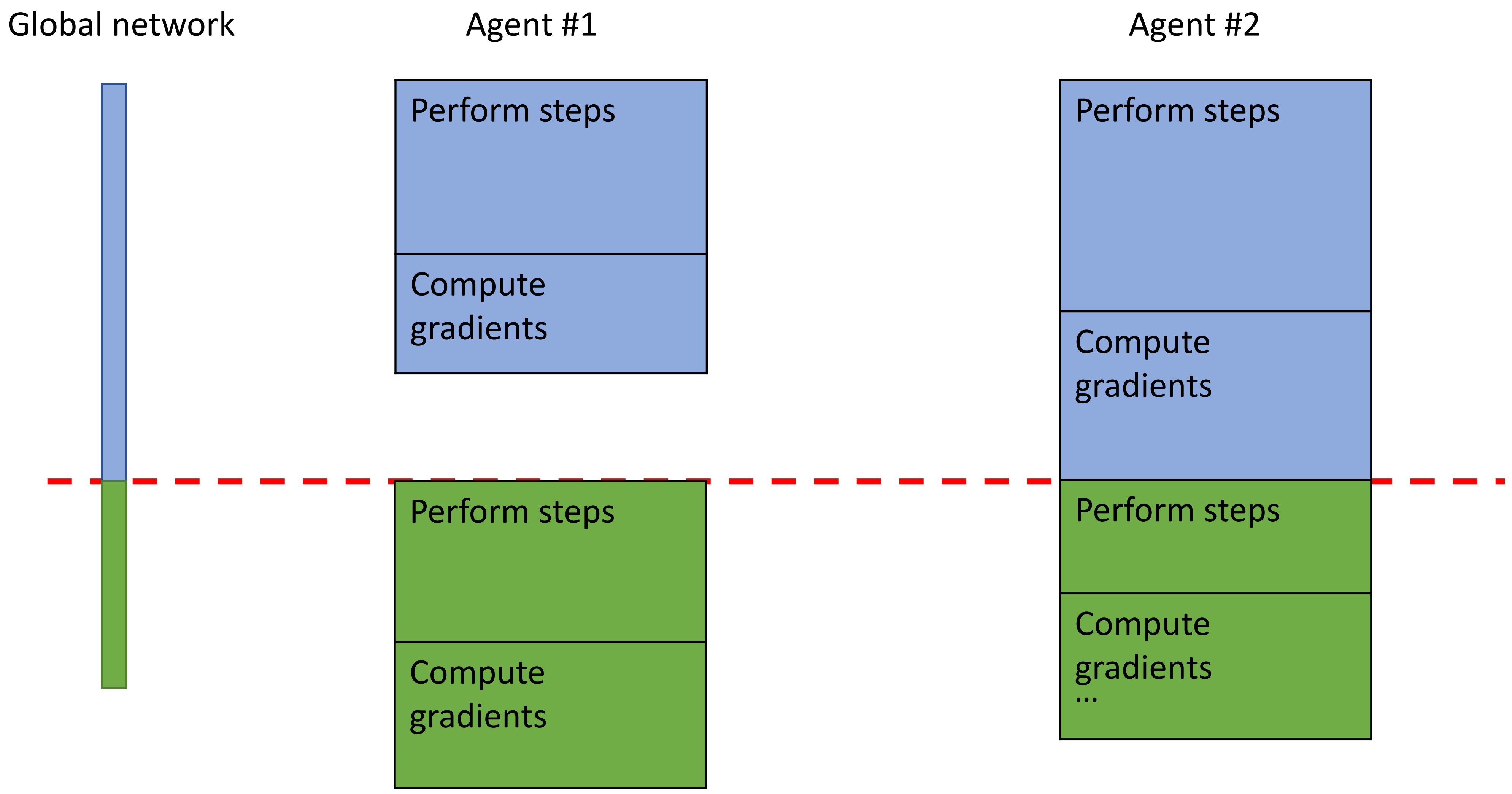
- Compute gradients for every visited state
  - Policy network: $\nabla_\theta \log \pi(a_t|s_t,\theta)\ (R_t - V(s_t,\theta_v))$
  - Value network: $\nabla_{\theta_v}\left(R - V(s_i;\theta_v)\right)^2$

- Update global shared parameters with computed gradients

Disadvantage of A3C

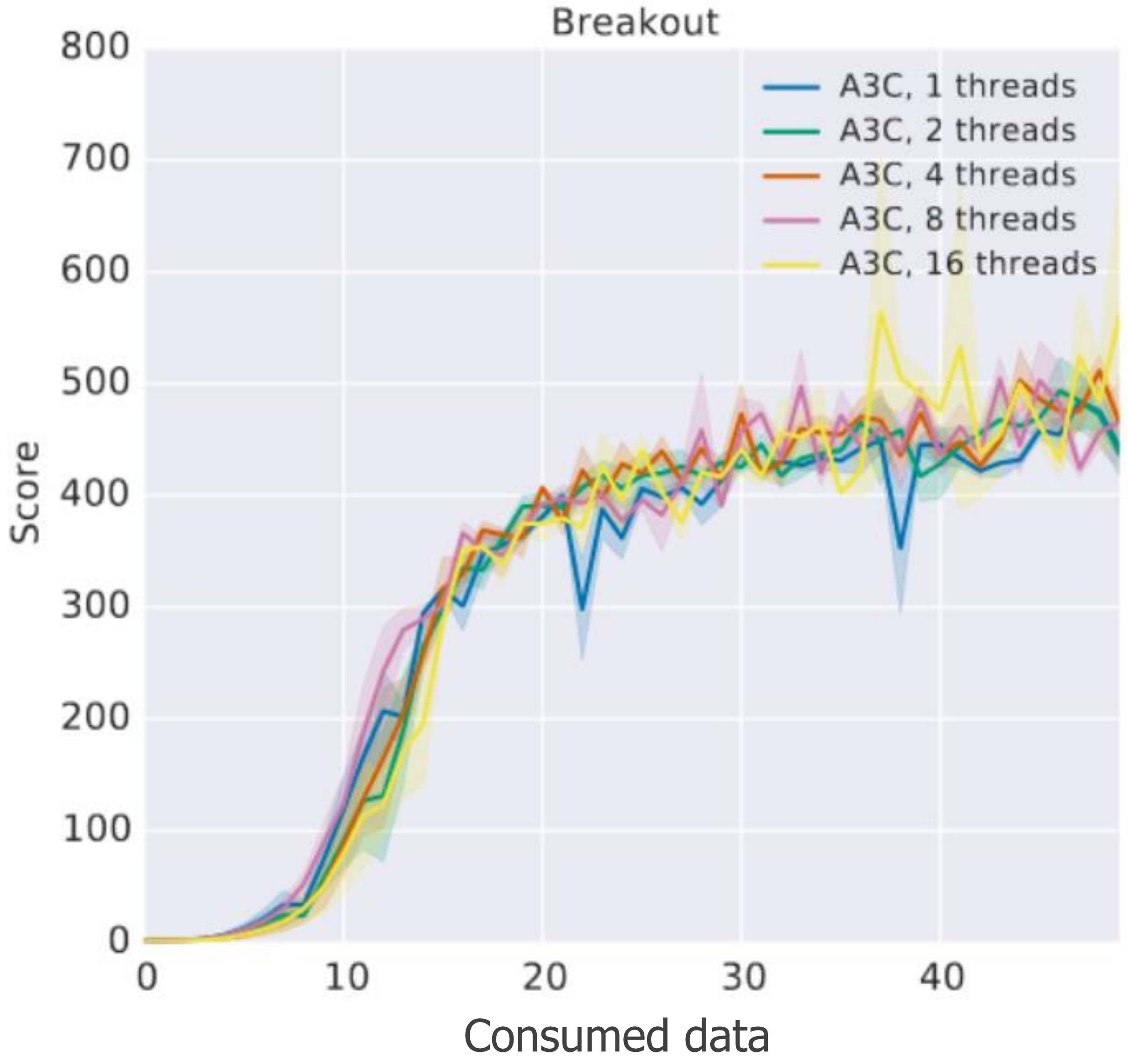


Synchronous version of A3C => A2C



Advantages of „Asynchronous methods“

- Simple extension
- Can be applied to a big variety of algorithms
- Makes robust NN training possible
- Linear speedup


Breakout
A3C, 1 threads
A3C, 2 threads
A3C, 4 threads
A3C, 8 threads
A3C, 16 threads
Score
Consumed data
0
10
20
30
40
0
100
200
300
400
500
600
700
800


Breakout
A3C, 1 threads
A3C, 2 threads
A3C, 4 threads
A3C, 8 threads
A3C, 16 threads
Training time (hours)
0
2
4
6
8
10
12
14
0
100
200
300
400
500
600