ELTE | FACULTY OF INFORMATICS

FUNCTION APPROXIMATION

Deep Reinforcement Learning

Balázs Nagy, PhD

ELTE | IK

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

Function Approximation

- Function approximation in reinforcement learning
- Estimating the state-value function from on-policy data
- Approximating v_π from experience generated using a known policy

Approximation example

Gridworld:

1	2	3	4	5	10 6

Approximation example

Gridworld:

1	2	3	4	5	10 6

State-Values:

5 1	6 2	7 3	8 4	9 5	10 6

Approximation example

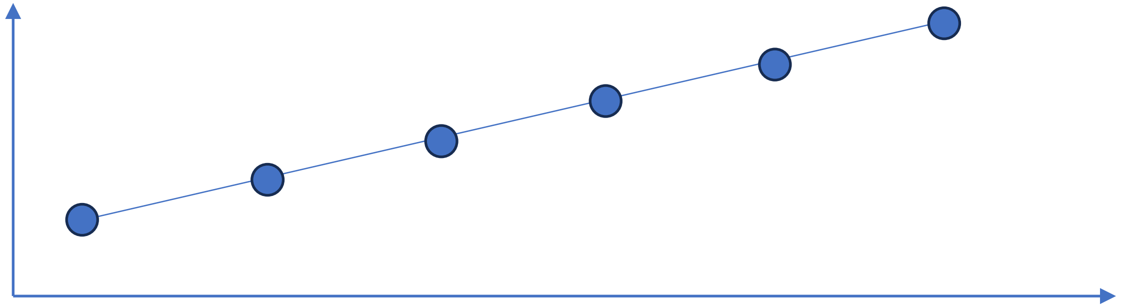
Gridworld:

1	2	3	4	5	6 10

State-Values:

1 5	2 6	3 7	4 8	5 9	6 10

State-Value Approximate Function:



$$\mathbf{w} \in \mathbb{R}^d$$

$$\hat{v}(s,\mathbf{w}) \approx v_\pi(s)$$

Approximation example

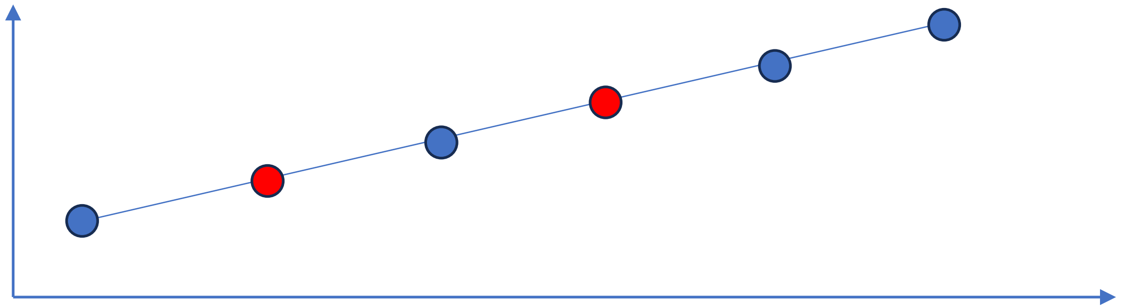
Gridworld:

					10
1	2	3	4	5	6

State-Values:

5	?	7	?	9	10
1	2	3	4	5	6

State-Value Approximate Function:



the approximate value function is represented not as a table but as a parameterized functional form with weight vector

$$\mathbf{w} \in \mathbb{R}^d$$

$$\hat{v}(s,\mathbf{w}) \approx v_\pi(s)$$

Why use Function Approximation?

- The dimensionality of *w* is much less than the number of states (*d*<<*S*)
- When a single state is updated, the change generalizes to affect the values of many other states
- Generalization makes the learning potentially more powerful
- Can be applicable to partially observable problems
- Potentially more difficult to manage and understand
- What function approximation can't do, however, is augment the state representation with memories of past observations

Trade-off

Tabular based approach:

- A continuous measure of prediction quality was not necessary because the learned value function could come to equal the true value function exactly
- The learned values at each state were decoupled (an update at one state affected no other)

Approximation based approach:

- By assumption there are far more states than weights, so making one state's estimate more accurate invariably means making others' less accurate

Objective function

- Mean Squared Value Error

$$\overline{\mathrm{VE}}(\mathbf{w}) \doteq \sum_{s \in \mathcal{S}} \mu(s) \Big[v_\pi(s) - \hat{v}(s,\mathbf{w}) \Big]^2$$

- On-policy distribution

$$\mu(s) \geq 0, \ \sum_s \mu(s) = 1$$

Stochastic-gradient and Semi-gradient Methods

- Weight update

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t - \frac{1}{2}\alpha\nabla\Big[v_\pi(S_t) - \hat{v}(S_t,\mathbf{w}_t)\Big]^2$$
$$= \mathbf{w}_t + \alpha\Big[v_\pi(S_t) - \hat{v}(S_t,\mathbf{w}_t)\Big]\nabla\hat{v}(S_t,\mathbf{w}_t)$$

- Partial derivate

$$\nabla f(\mathbf{w}) \doteq \left(\frac{\partial f(\mathbf{w})}{\partial w_1}, \frac{\partial f(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial f(\mathbf{w})}{\partial w_d}\right)^\top$$

- General update rule

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\Big[U_t - \hat{v}(S_t,\mathbf{w}_t)\Big]\nabla\hat{v}(S_t,\mathbf{w}_t)$$

Stochastic-gradient and Semi-gradient Methods

- Weight update

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t - \frac{1}{2}\alpha\nabla\Big[v_\pi(S_t) - \hat{v}(S_t,\mathbf{w}_t)\Big]^2$$
$$= \mathbf{w}_t + \alpha\Big[v_\pi(S_t) - \hat{v}(S_t,\mathbf{w}_t)\Big]\nabla\hat{v}(S_t,\mathbf{w}_t)$$

- Partial derivate

$$\nabla f(\mathbf{w}) \doteq \left(\frac{\partial f(\mathbf{w})}{\partial w_1}, \frac{\partial f(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial f(\mathbf{w})}{\partial w_d}\right)^\top$$

- General update rule

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\Big[\boxed{U_t} - \hat{v}(S_t,\mathbf{w}_t)\Big]\nabla\hat{v}(S_t,\mathbf{w}_t)$$

- General update

$$s \mapsto u$$

- MC

$$S_t \mapsto G_t$$

- TD(0)

$$S_t \mapsto R_{t+1} + \gamma\hat{v}(S_{t+1},\mathbf{w}_t)$$

- N-step TD

$$S_t \mapsto G_{t:t+n}$$

Pseudocode

Gradient Monte Carlo Algorithm for Estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using π
  Loop for each step of episode, $t = 0, 1, \ldots, T-1$:
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha \big[\underline{G_t} - \hat{v}(S_t,\mathbf{w}) \big] \nabla \hat{v}(S_t,\mathbf{w})$

Not biased

Semi-gradient (bootstrapping) methods

- Do not converge as robustly as gradient methods
- Converge reliably in important cases such as the linear case
- Typically enable significantly faster learning
- Enable learning to be continual and online
  (without waiting for the end of an episode)
- This enables them to be used on continuing problems and provides computational advantages

Pseudocode

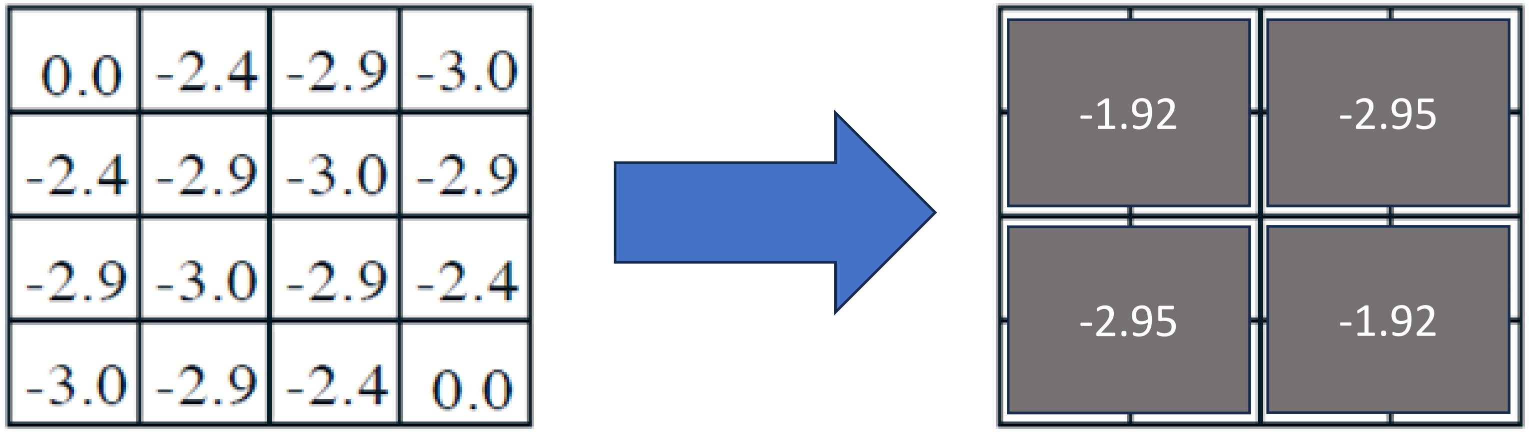
Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

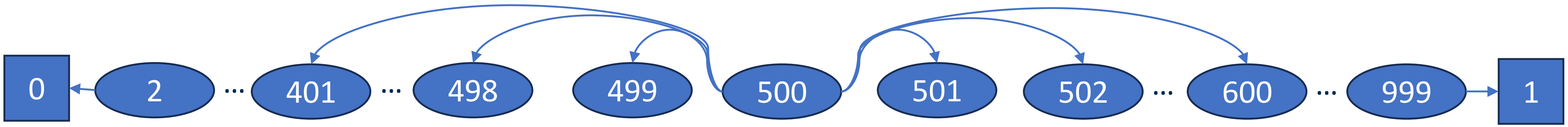
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose $A \sim \pi(\cdot|S)$
    Take action A, observe R, S'
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[\underbrace{R + \gamma\hat{v}(S',\mathbf{w})}_{\text{Bootstrapped}} - \hat{v}(S,\mathbf{w})\big]\nabla\hat{v}(S,\mathbf{w})$
    $S \leftarrow S'$
  until S' is terminal

State Aggregation

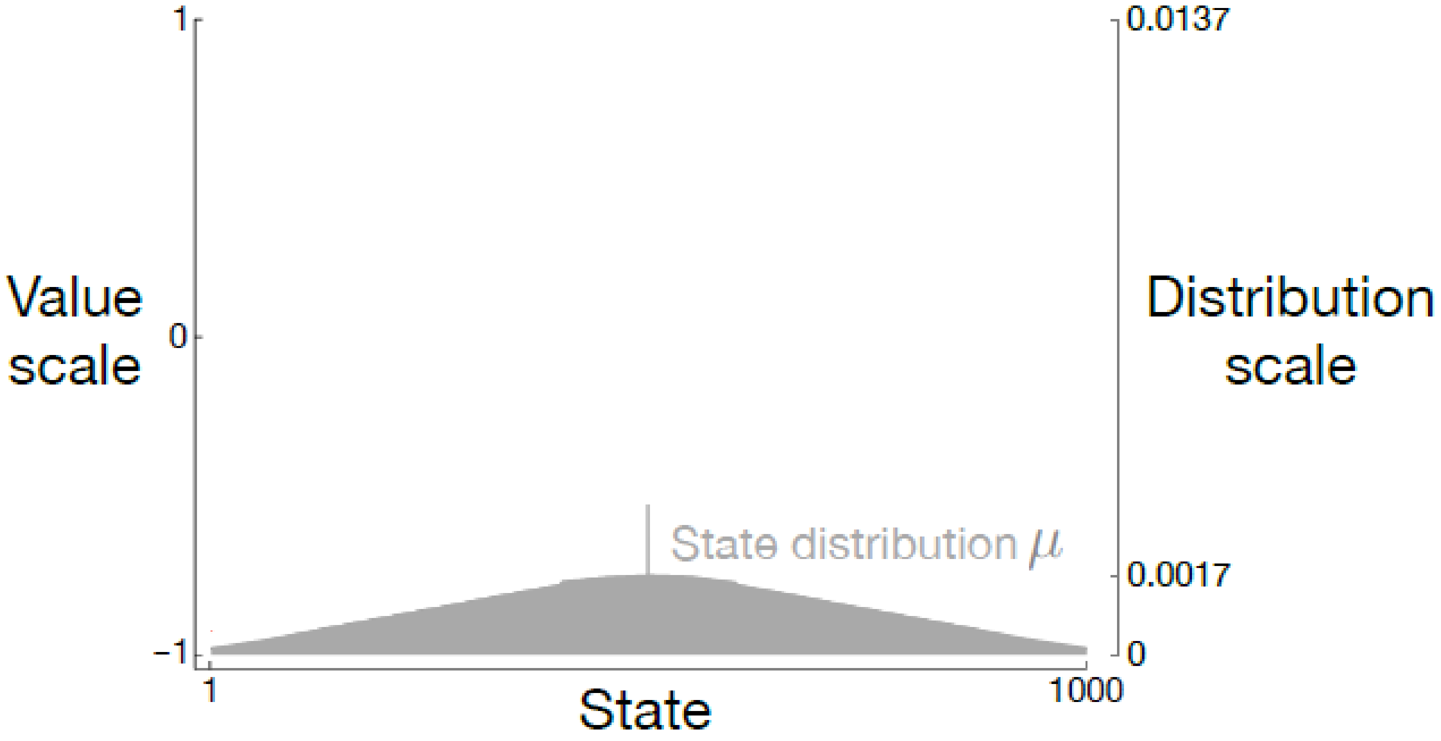
- A simple form of generalizing function approximation in which states are grouped together, with one estimated value



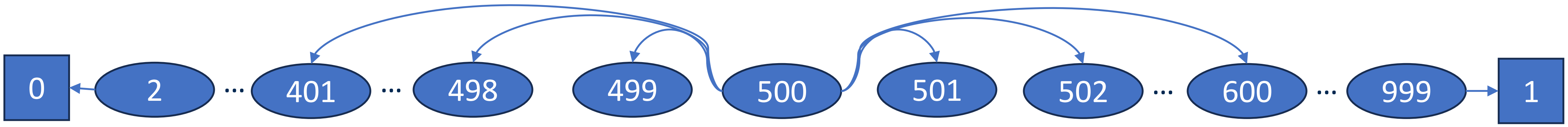
1000-state Random Walk



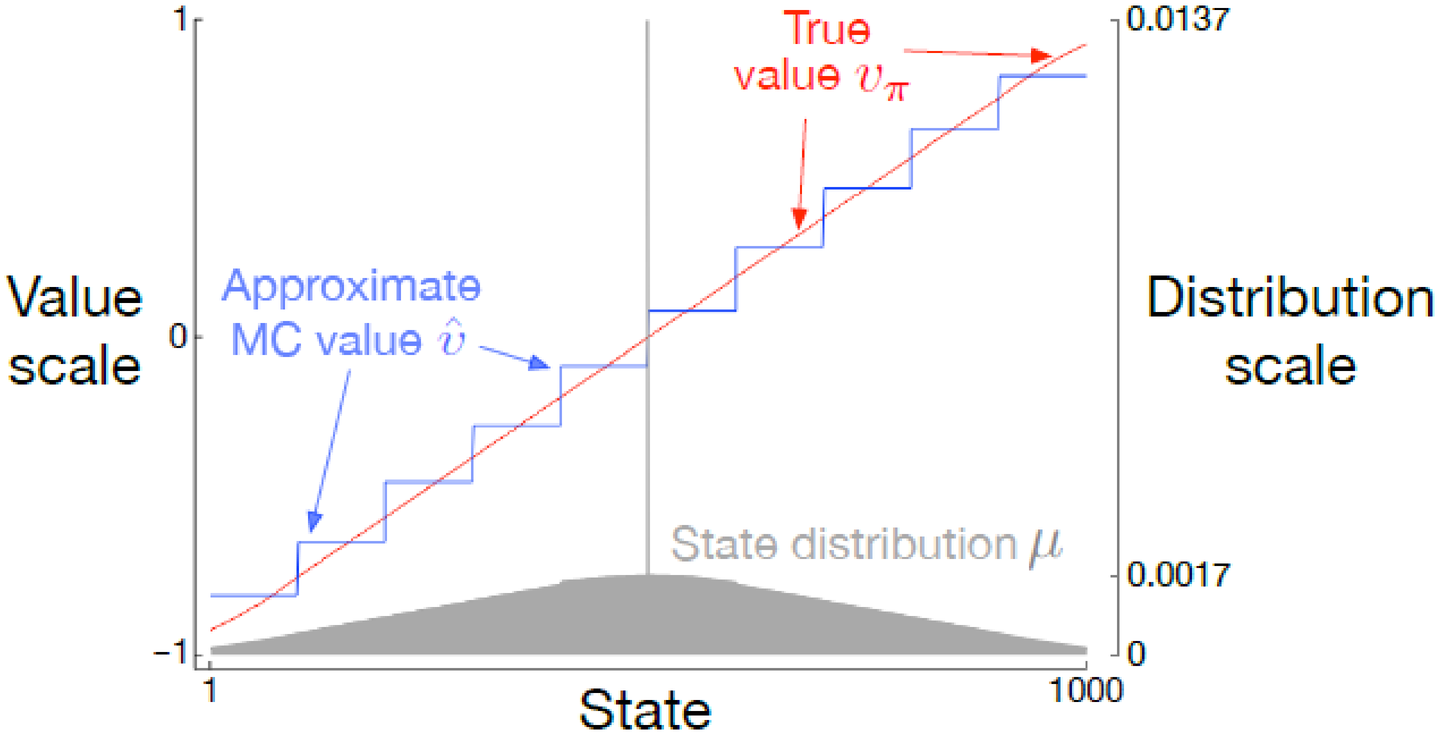
State aggregation:
10 groups
of 100 states



1000-state Random Walk



State aggregation:
10 groups
of 100 states



Linear function approximation

- special cases in which the approximate function is a linear function of the weight vector

$$\hat{v}(s,\mathbf{w}) \doteq \mathbf{w}^\top \underbrace{\mathbf{x}(s)}_{\text{feature vector}} \doteq \sum_{i=1}^{d} w_i x_i(s)$$

Linear function approximation

- Use SGD updates with linear function approximation
- The gradient of the approximate value function

$$\nabla \hat{v}(s,\mathbf{w}) = \mathbf{x}(s)$$

- SGD update in Linear form:

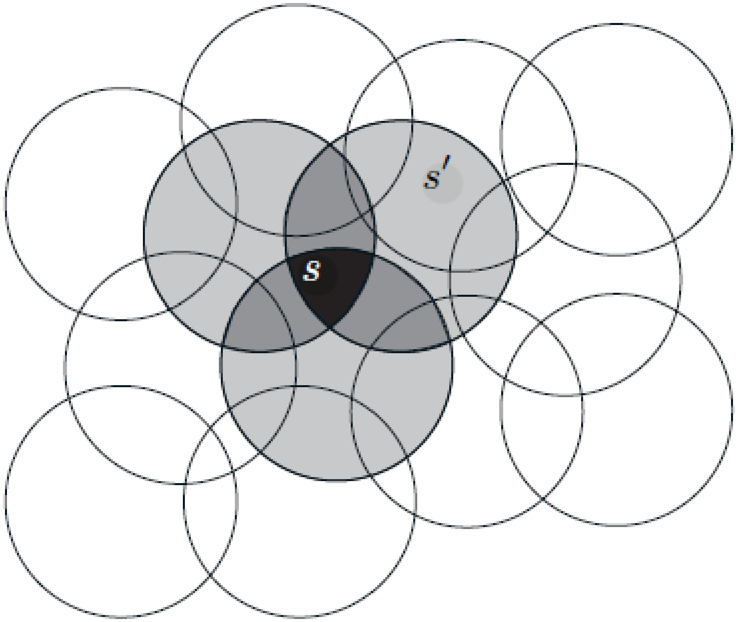
$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \Big[U_t - \hat{v}(S_t,\mathbf{w}_t) \Big] \mathbf{x}(S_t)$$

Feature Construction for Linear Methods

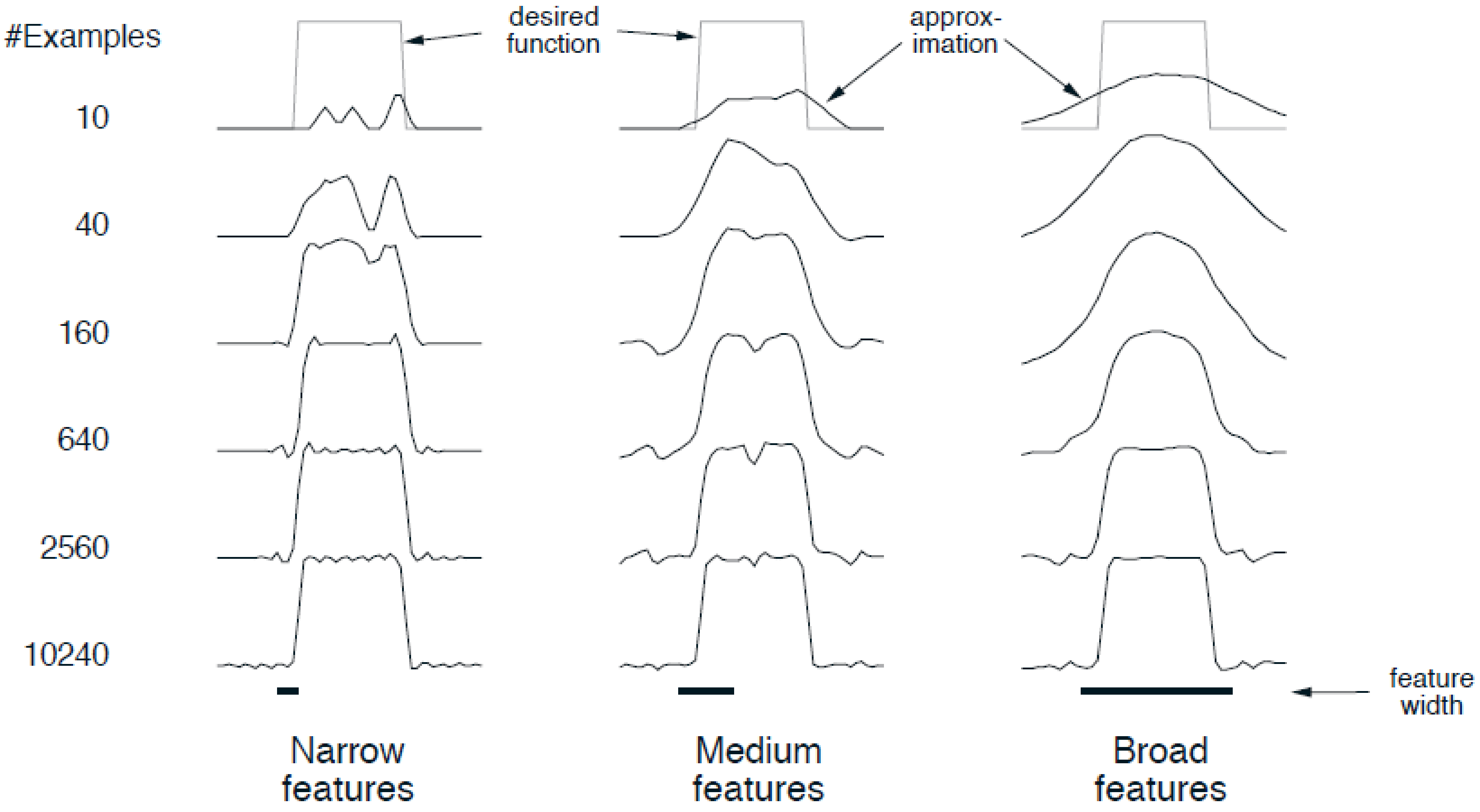
- Depends critically on how the states are represented in terms of features
- Features appropriate to the task is an important way of adding prior domain knowledge
- Limitation of the linear form:
  cannot take into account any interactions between features

Coarse coding

- Features corresponding to circles in state space
- Binary features
  - state is inside a circle: the corresponding feature value = 1 (otherwise 0)
- Coarsely code for the location of the state

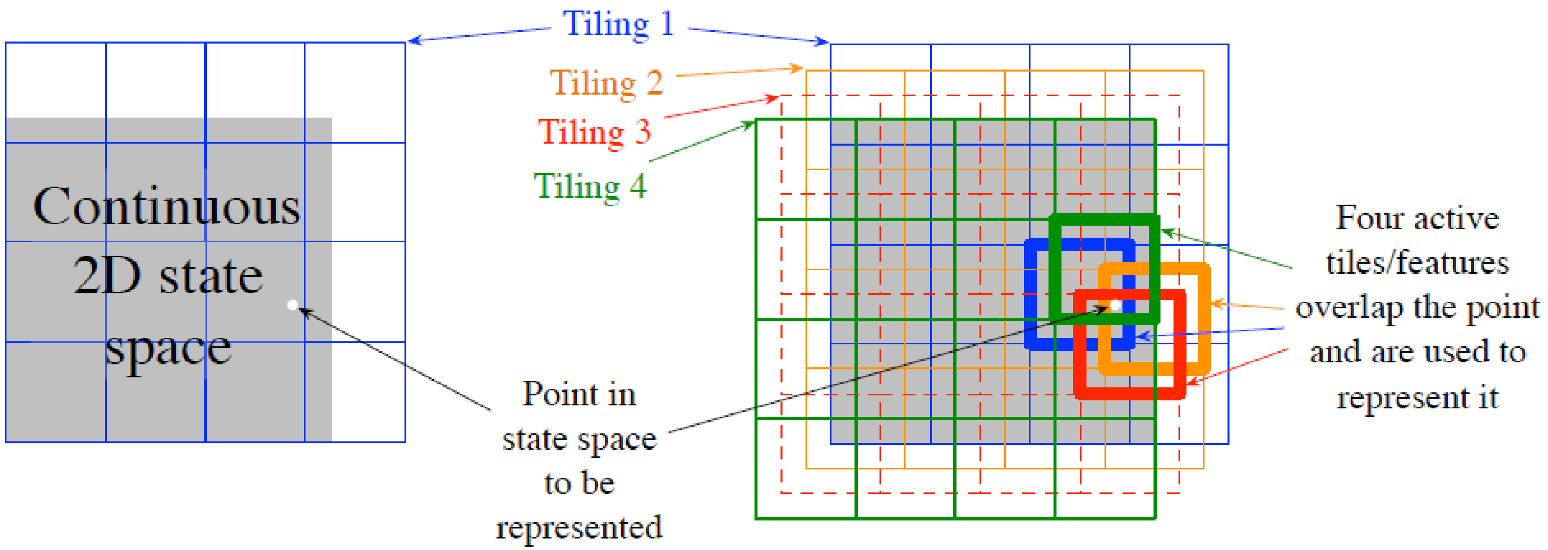


Effect of Coarse Coding



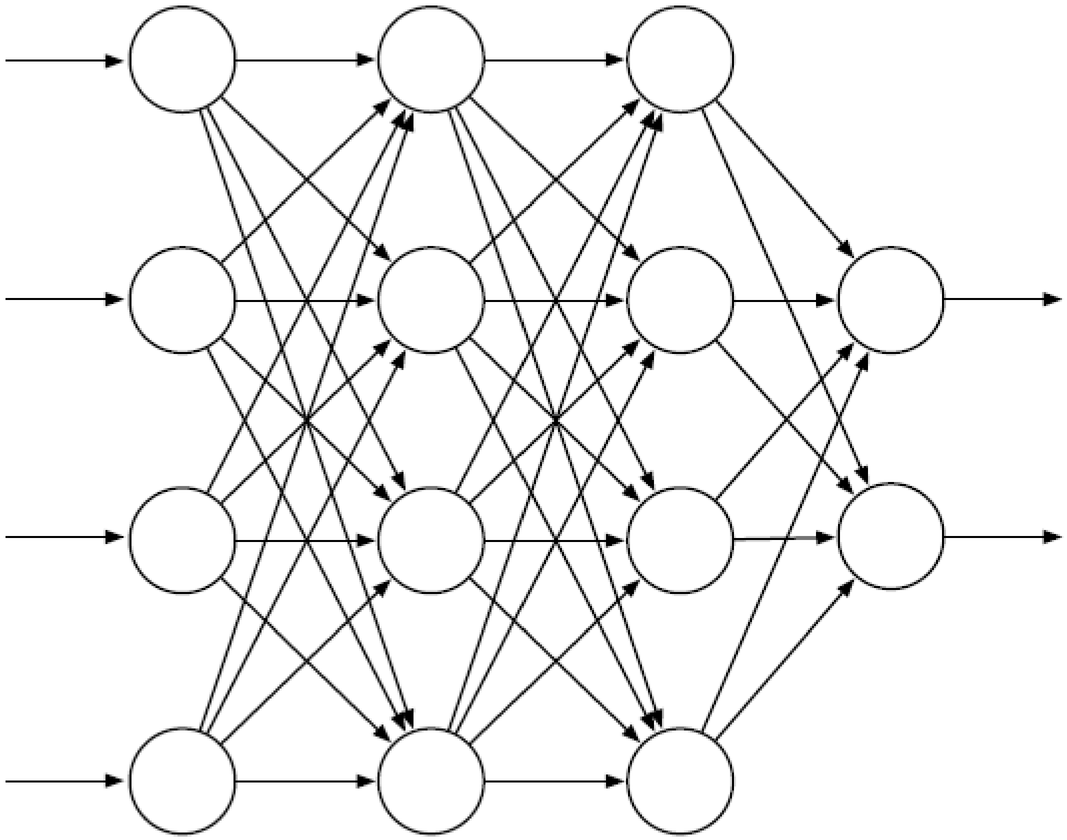
Tile coding

- A form of coarse coding for multi-dimensional continuous spaces
- Flexible and computationally efficient
- Practical feature representation for sequential data



Nonlinear function approximation

Universal approximation theorem:
any continuous function can be approximated arbitrarily well by a **neural network** with at least 1 hidden layer with a finite number of weights



Deep Reinforcement Learning Algorithms

Value Learning

Find $Q(s,a)$

$$a = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

Policy Learning

Find $\pi(s)$

Sample $a \sim \pi(s)$

Deep Reinforcement Learning Algorithms

Value Learning

Find $Q(s,a)$

$$a = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

Policy Learning

Find $\pi(s)$

Sample $a \sim \pi(s)$

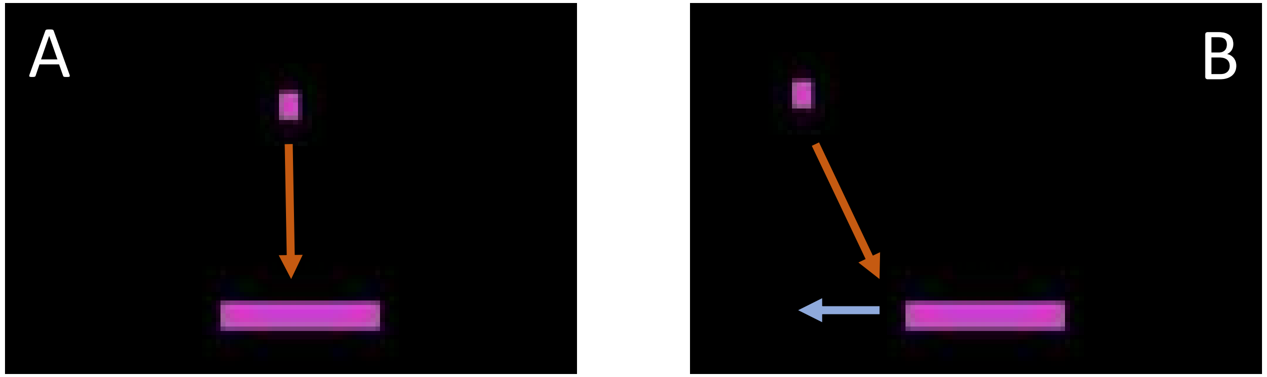
Q function intuition

Atari game - Breakout

Q function intuition

Atari game - Breakout

It can be very difficult for humans to accurately estimate Q-values

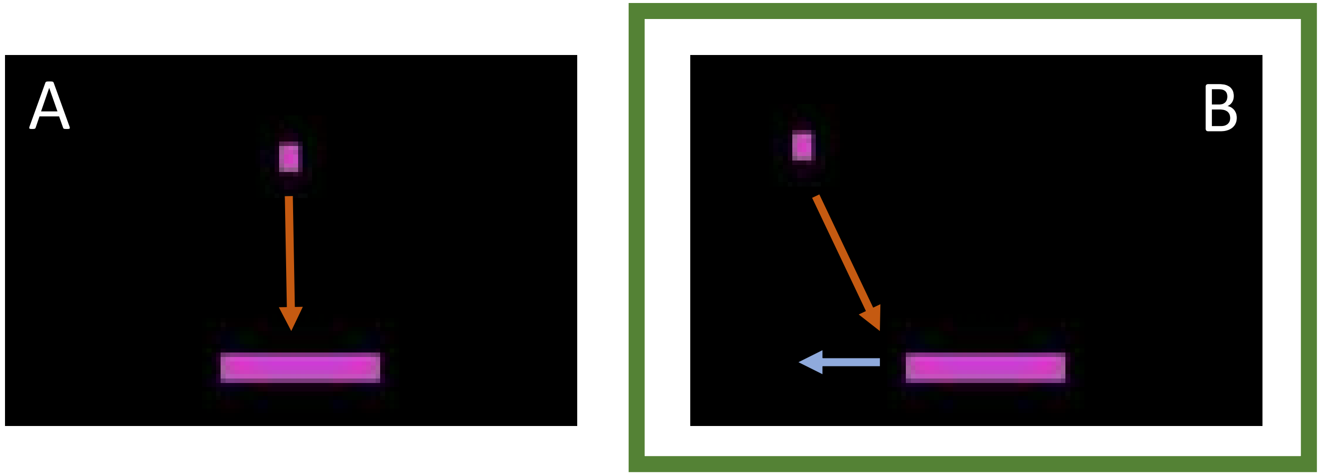


Which (s, a) pair has a higher Q-value?

Q function intuition

Atari game - Breakout

It can be very difficult for humans to accurately estimate Q-values



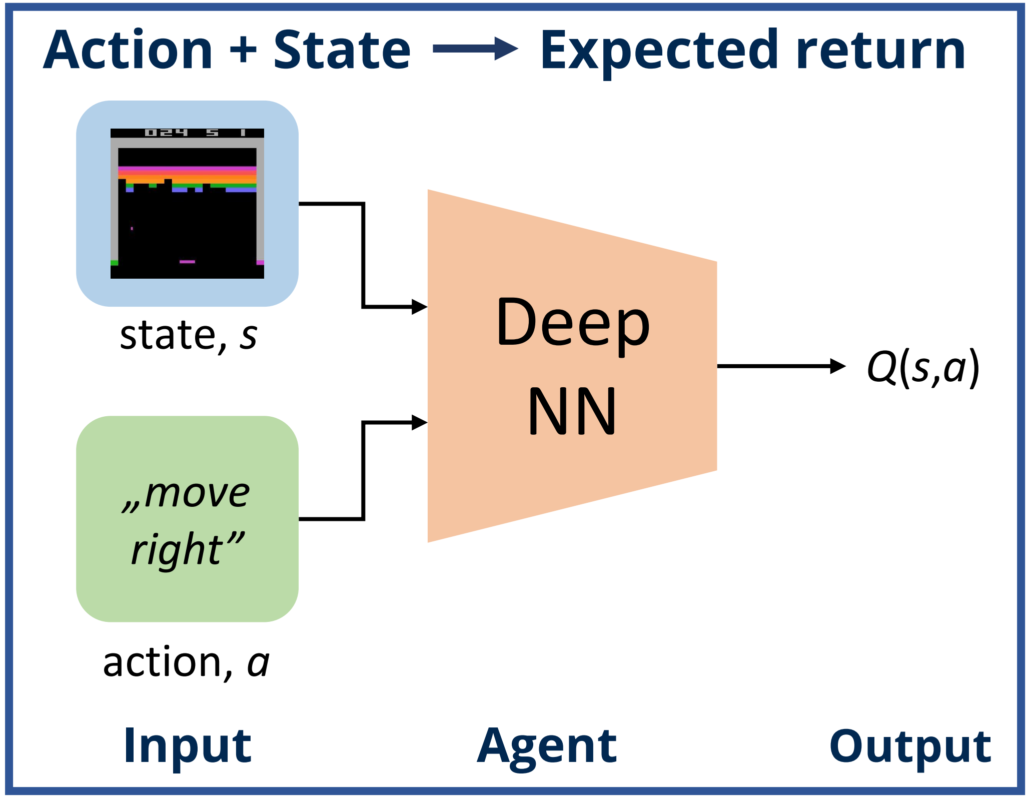
Which (s, a) pair has a higher Q-value? 🤔

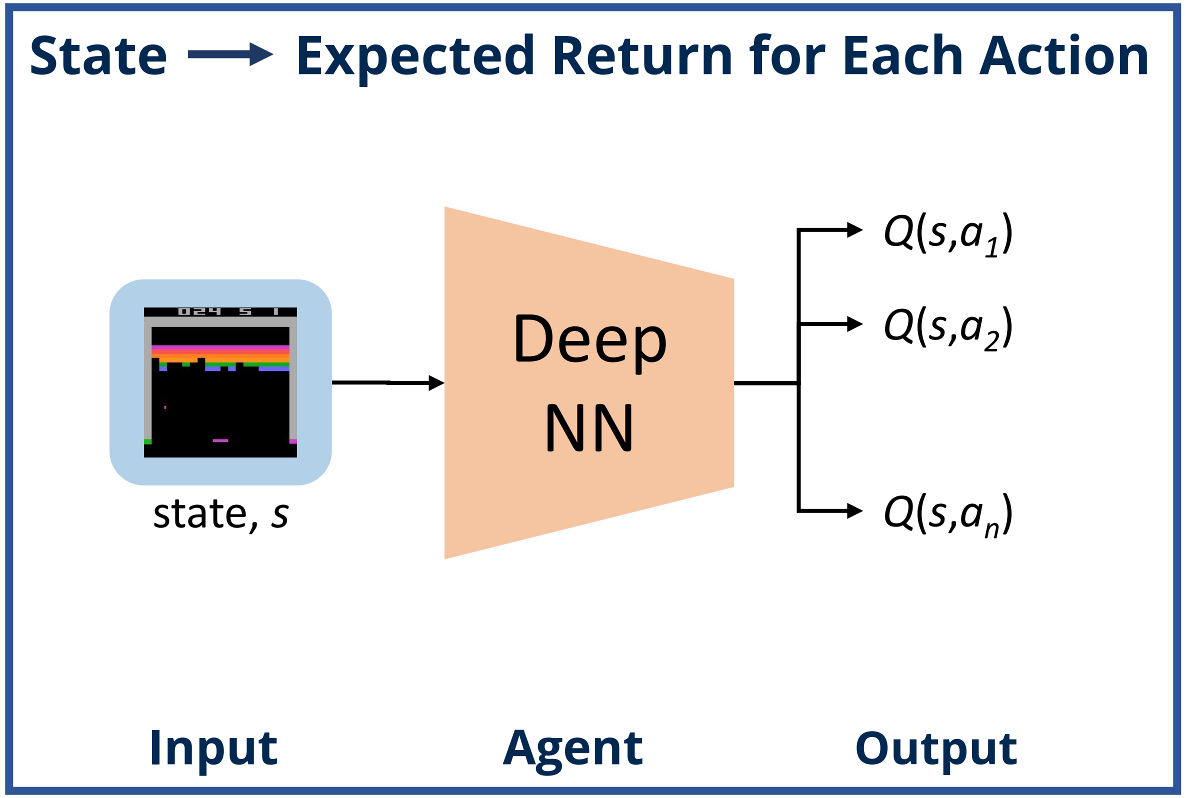
Deep Q Networks (DQN): Training

How can we use deep neural networks to model Q-functions?

Deep Q Networks (DQN): Training

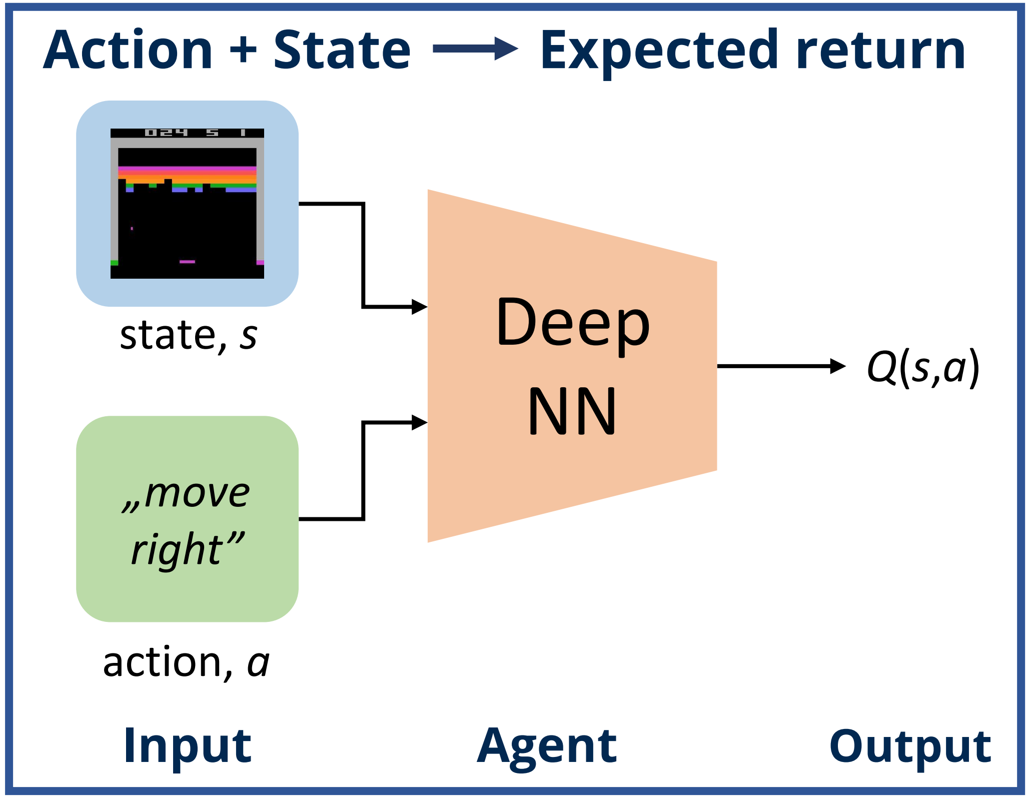
How can we use deep neural networks to model Q-functions?



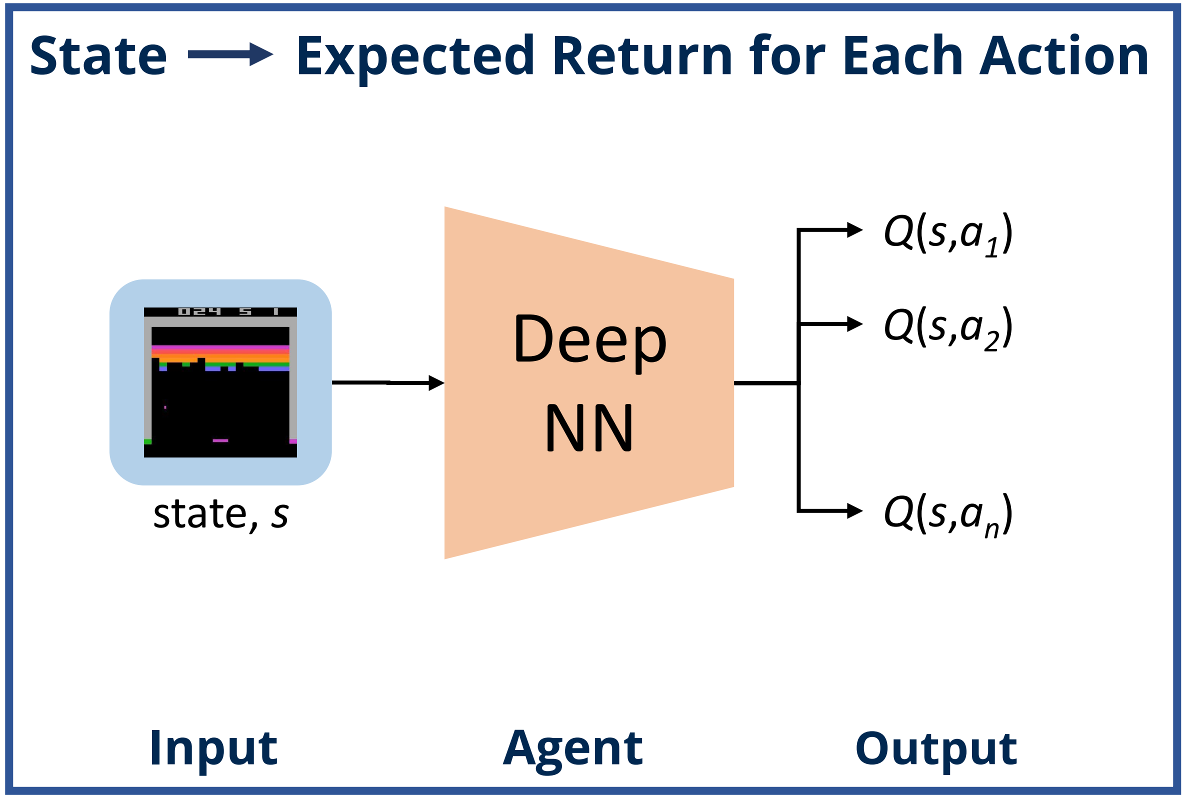


Deep Q Networks (DQN): Training

How can we use deep neural networks to model Q-functions?



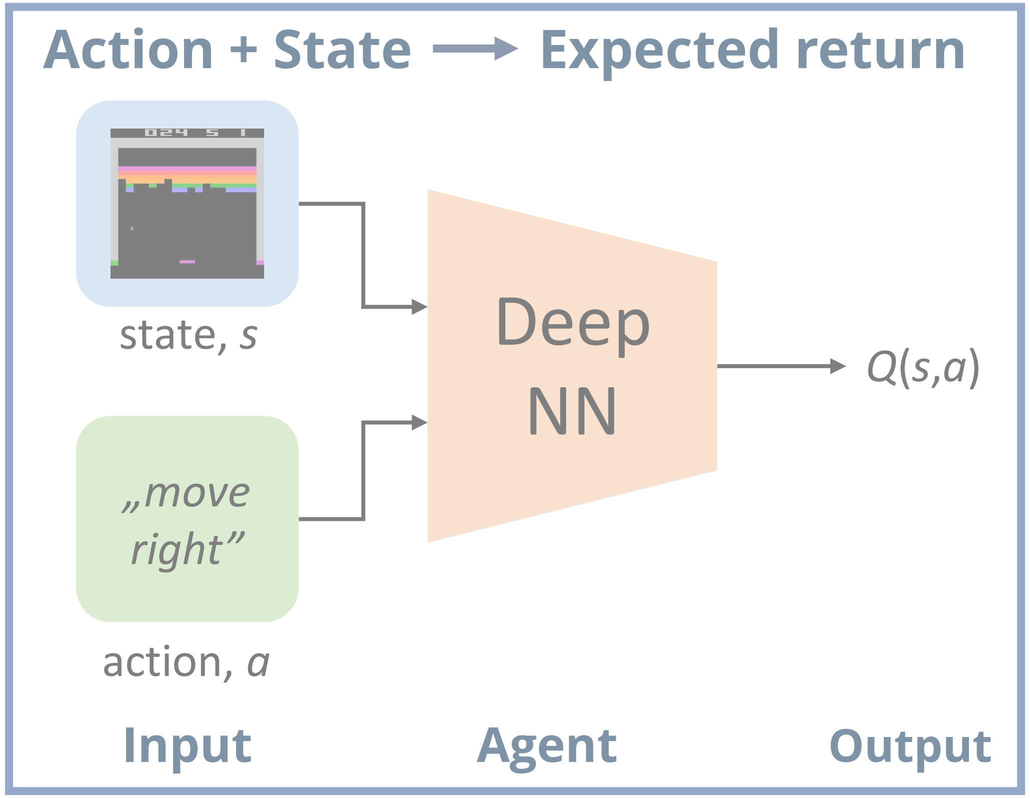
Multiple evaluation needed



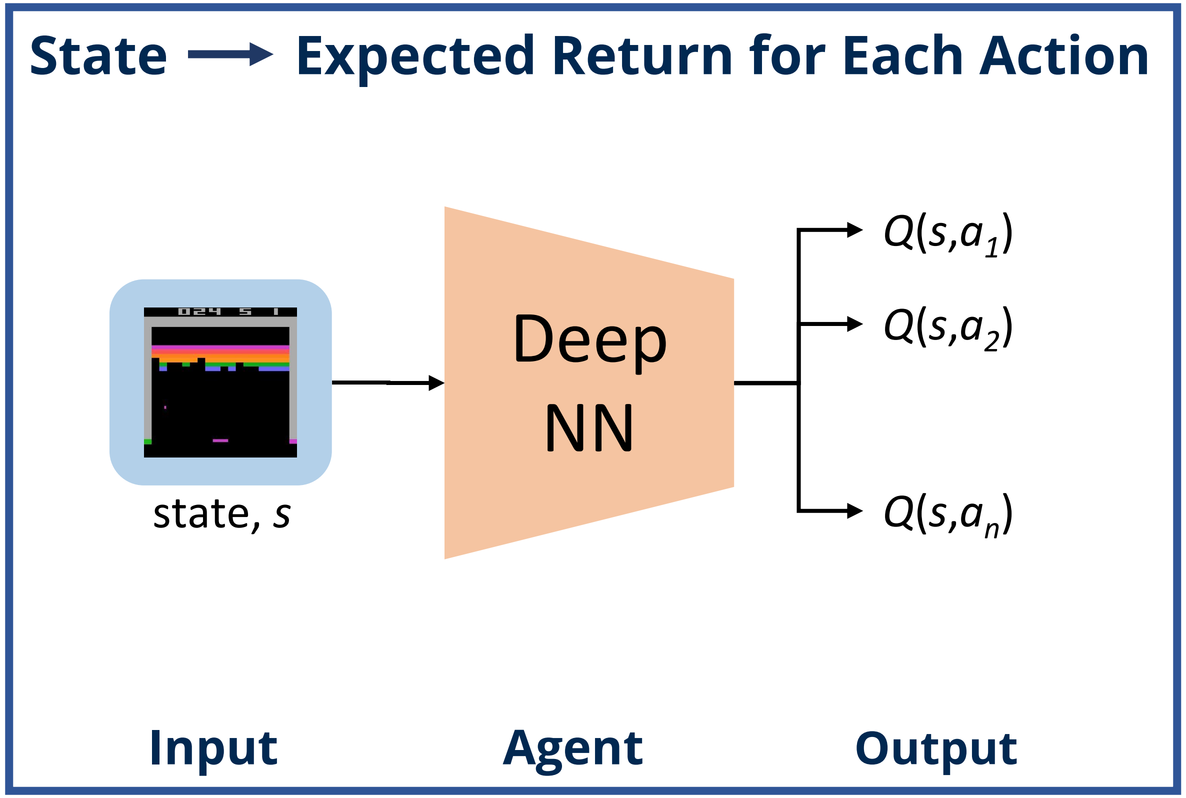
More efficient

Deep Q Networks (DQN): Training

How can we use deep neural networks to model Q-functions?



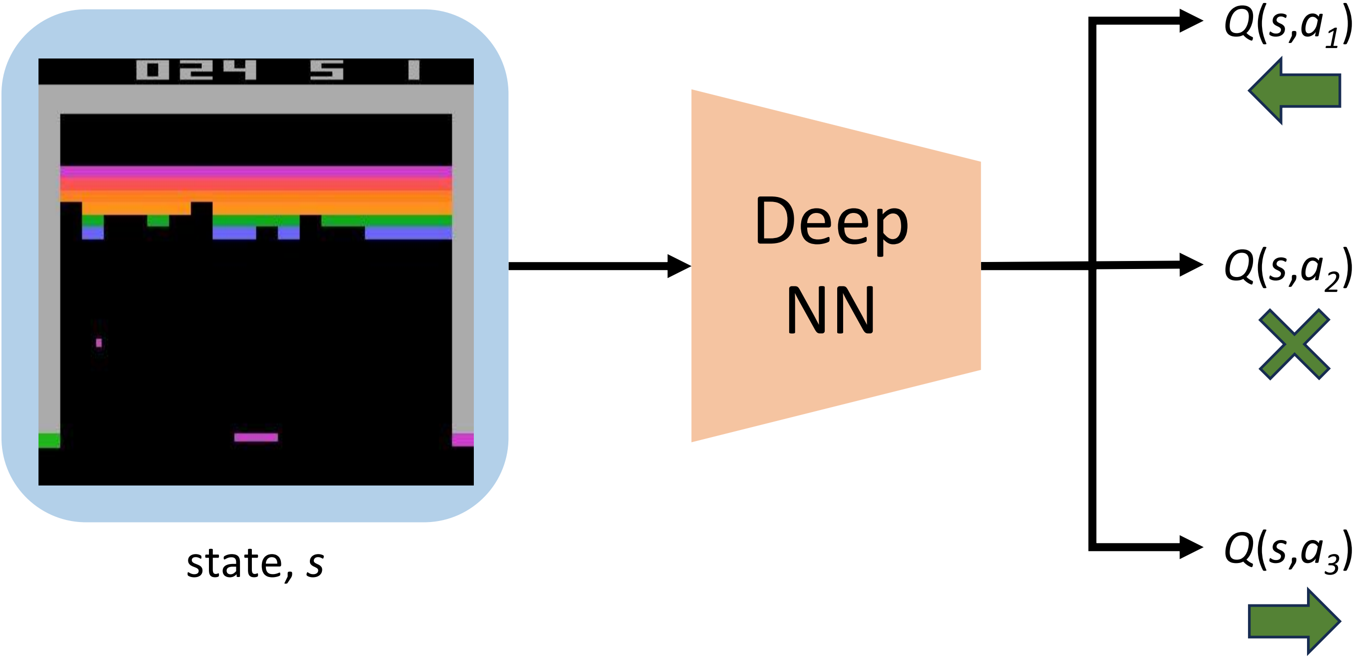
Multiple evaluation needed



More efficient

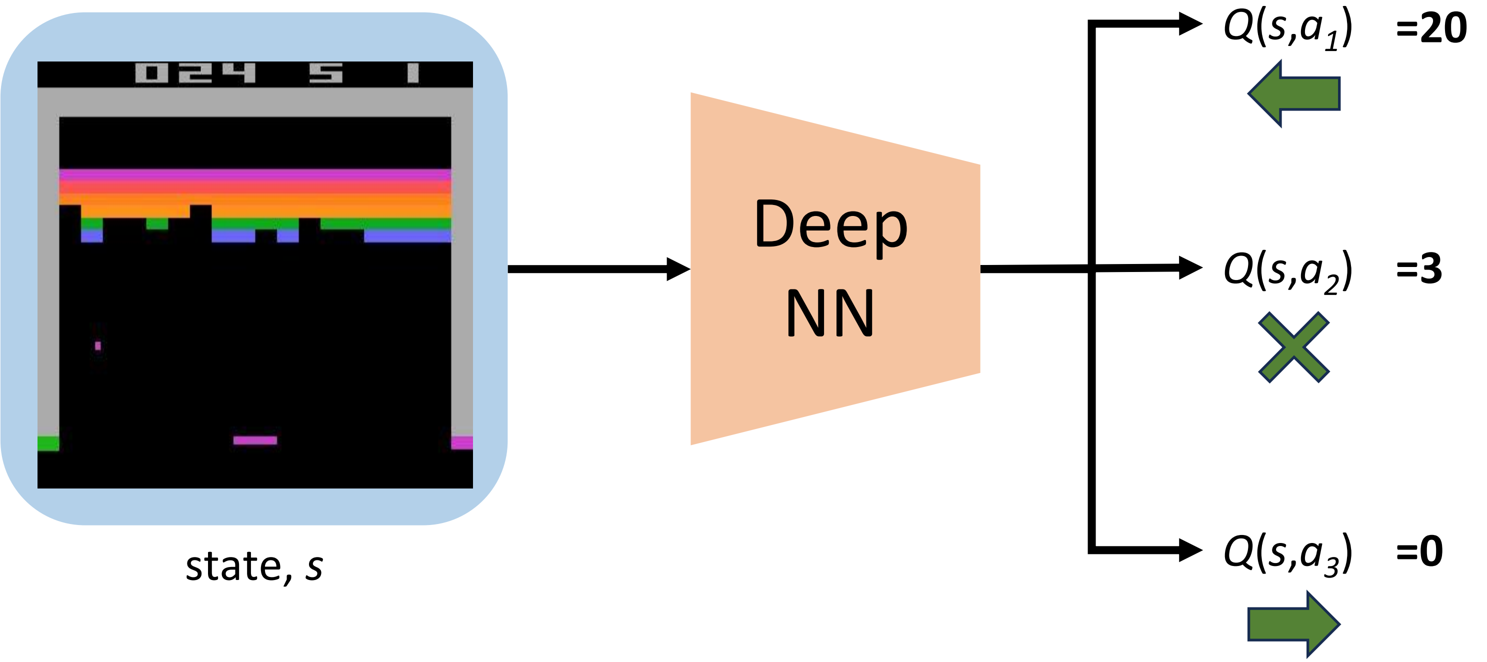
Deep Q Networks Summary

Use NN to learn Q-function and then use to infer the optimal policy, $\pi(s)$



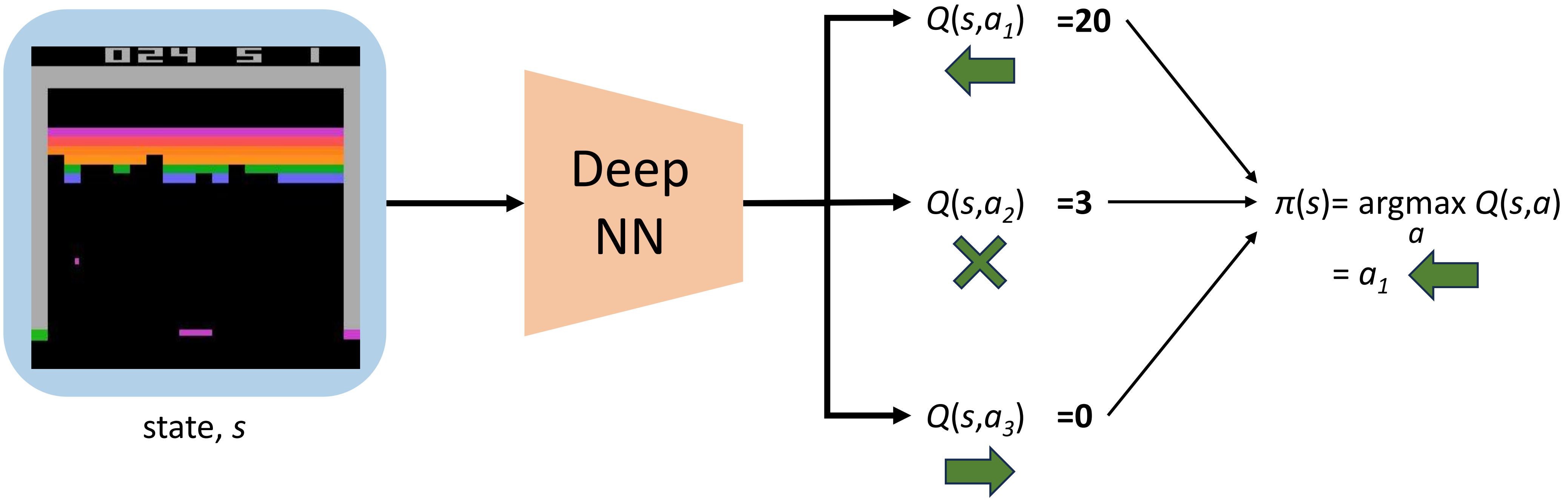
Deep Q Networks Summary

Use NN to learn Q-function and then use to infer the optimal policy, $\pi(s)$



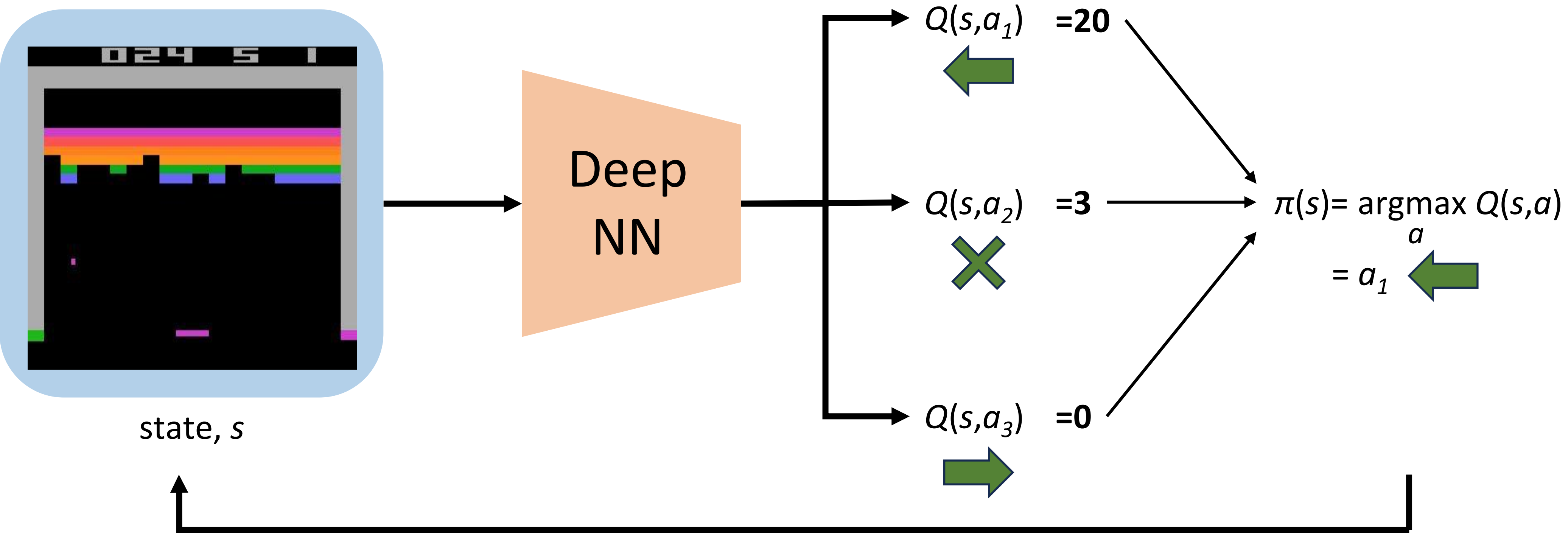
Deep Q Networks Summary

Use NN to learn Q-function and then use to infer the optimal policy, $\pi(s)$



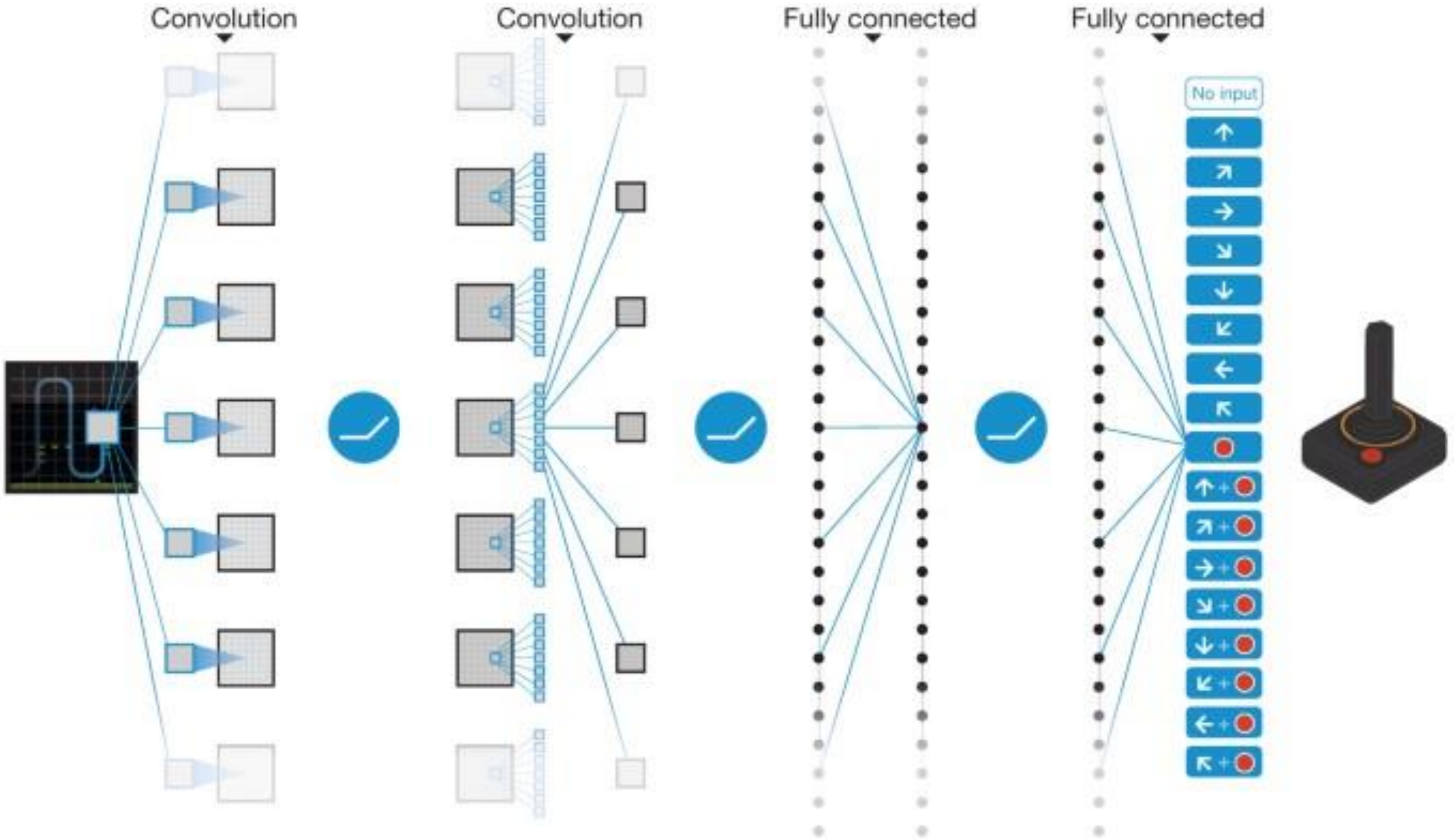
Deep Q Networks Summary

Use NN to learn Q-function and then use to infer the optimal policy, $\pi(s)$



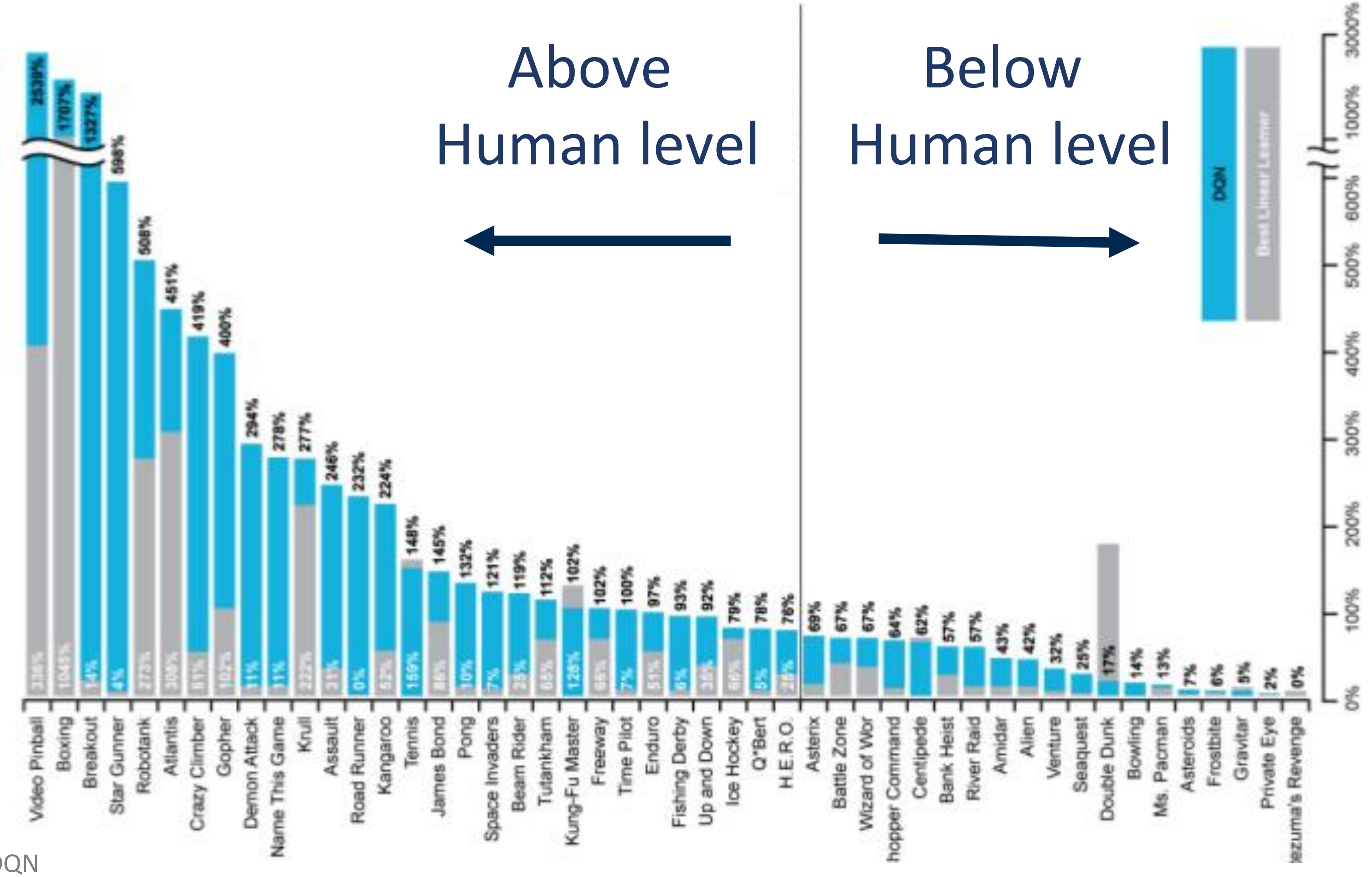
Send action back to the environment and receive next state

DQN Atari playing Network



Mnih, V., Kavukcuoglu, K., Silver, D. *et al.* Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015). https://doi.org/10.1038/nature14236

DQN Atari Results



https://github.com/Neo-47/Atari-DQN

Downsides of Q-learning

Complexity:

- Can model scenarios where the action space is discrete and small
- Cannot handle continuous action spaces

Flexibility:

- Policy is deterministically computed from the Q function by maximizing the reward → cannot learn stochastic policies

Downsides of Q-learning

Complexity:

- Can model scenarios where the action space is discrete and small
- Cannot handle continuous action spaces

Flexibility:

- Policy is deterministically computed from the Q function by maximizing the reward → cannot learn stochastic policies

To address these, consider a new class of RL training algorithms:

Policy gradient methods

Deep Reinforcement Learning Algorithms

Value Learning

Find $Q(s,a)$

$$a = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

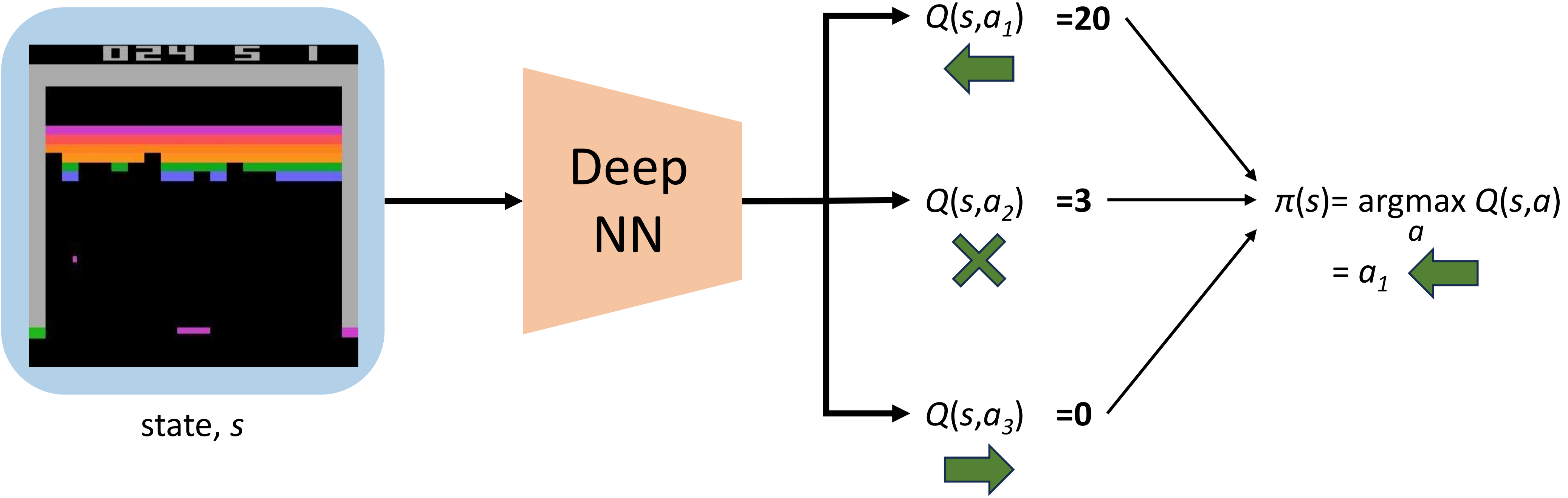
Policy Learning

Find $\pi(s)$

Sample $a \sim \pi(s)$

Deep Q Networks Summary

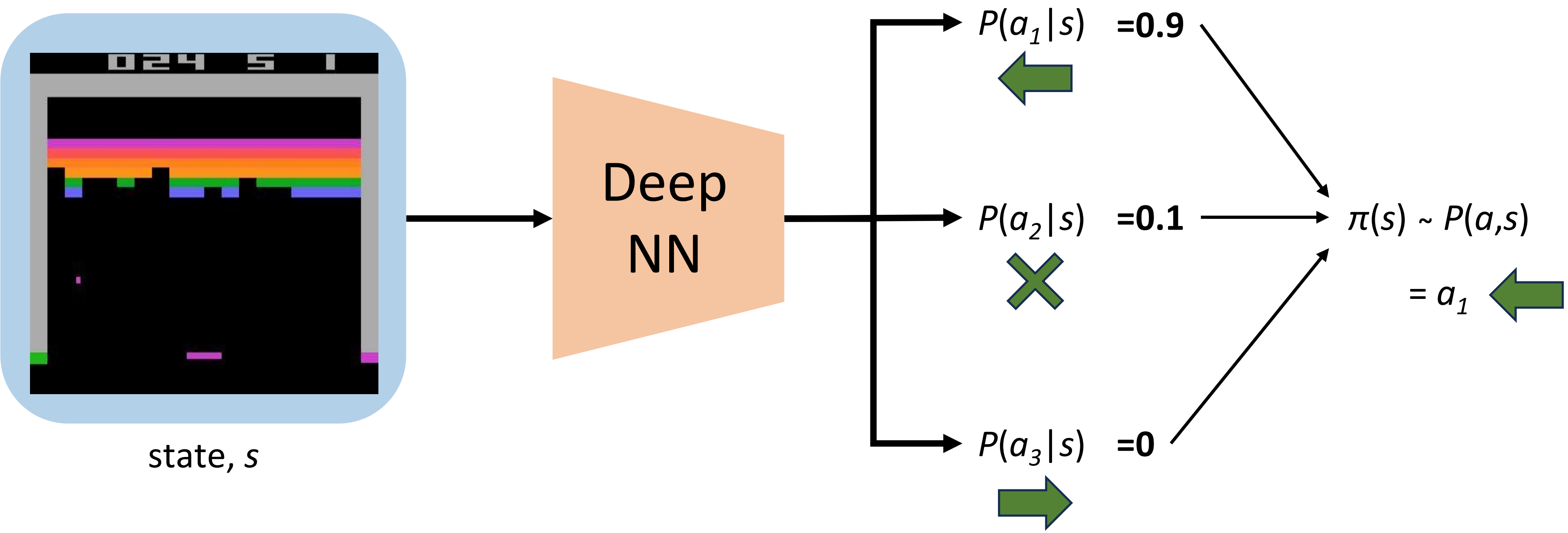
DQN: Approximate Q function and use to infer the optimal policy, $\pi(s)$



Policy Gradient (PG): Key Idea

DQN: Approximate Q function and use to infer the optimal policy, $\pi(s)$

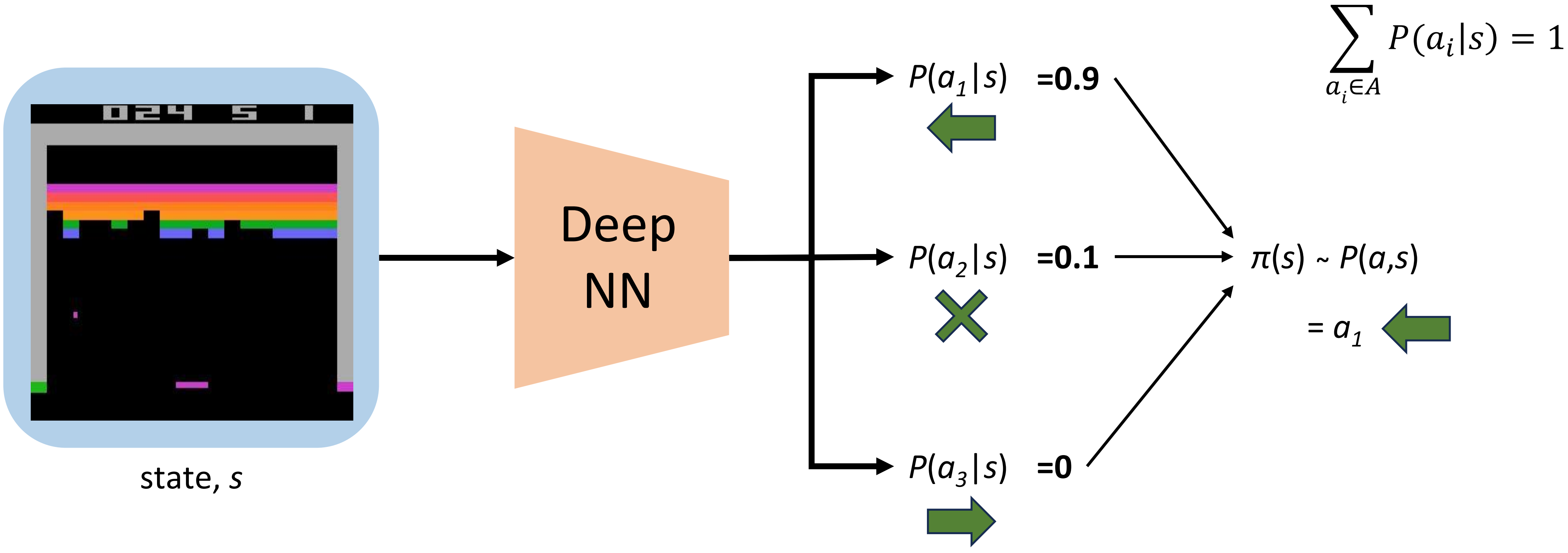
Policy Gradient: Directly optimize the policy $\pi(s)$



Policy Gradient (PG): Key Idea

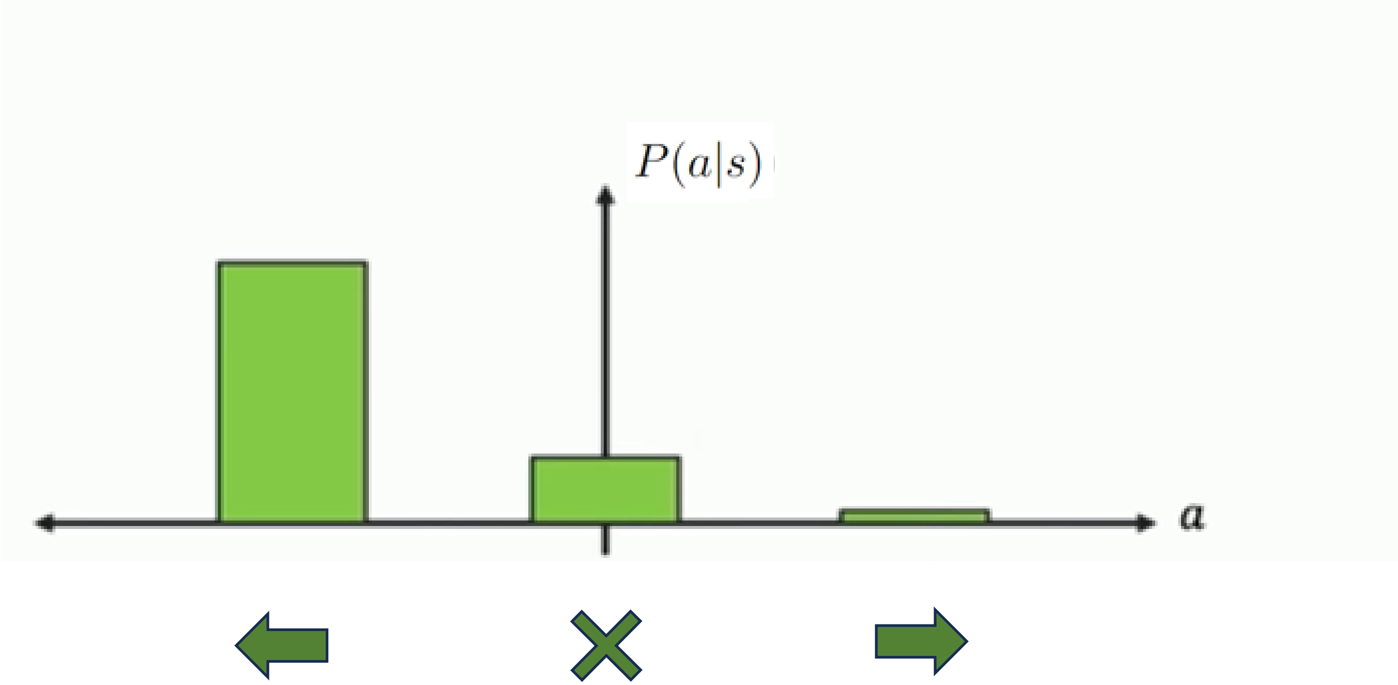
DQN: Approximate Q function and use to infer the optimal policy, $\pi(s)$

Policy Gradient: Directly optimize the policy $\pi(s)$



Discrete vs Continuous Action Spaces

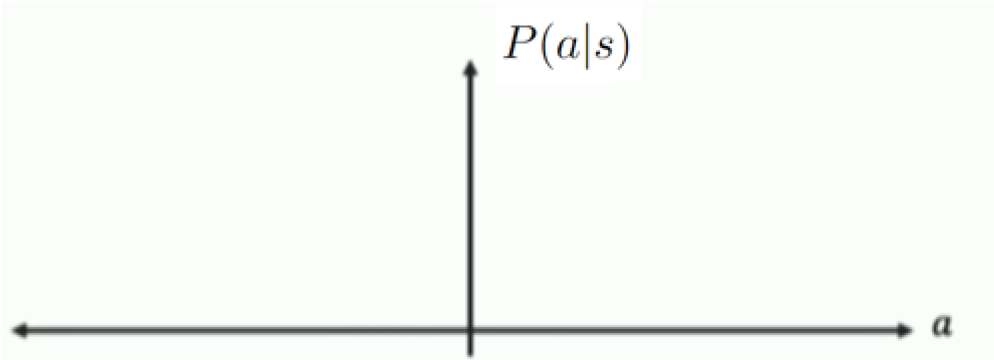
Discrete action space: which direction should I move?



Discrete vs Continuous Action Spaces

Discrete action space: which direction should I move?

Continuous action space: how fast should I move? 7 m/s



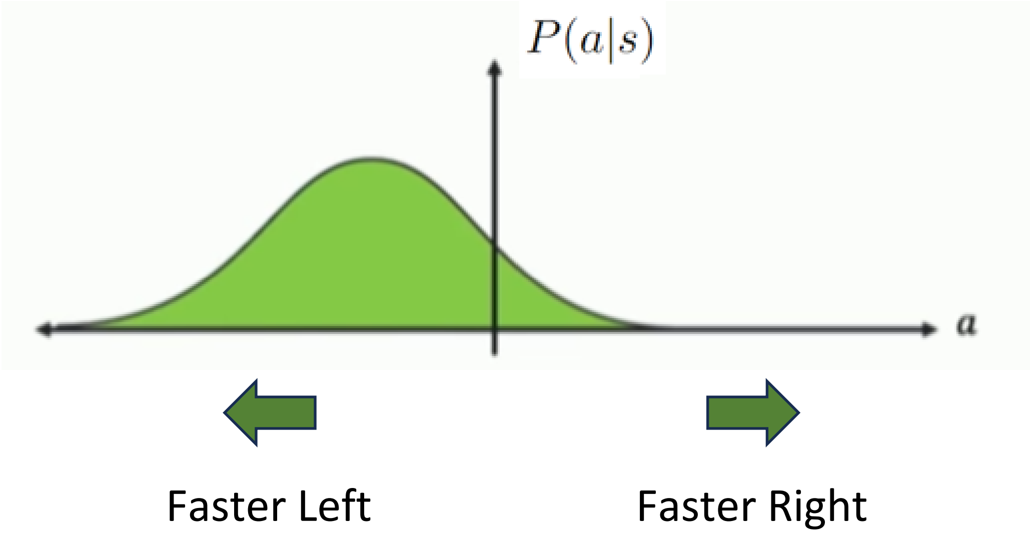
Faster Left

Faster Right

Discrete vs Continuous Action Spaces

Discrete action space: which direction should I move?

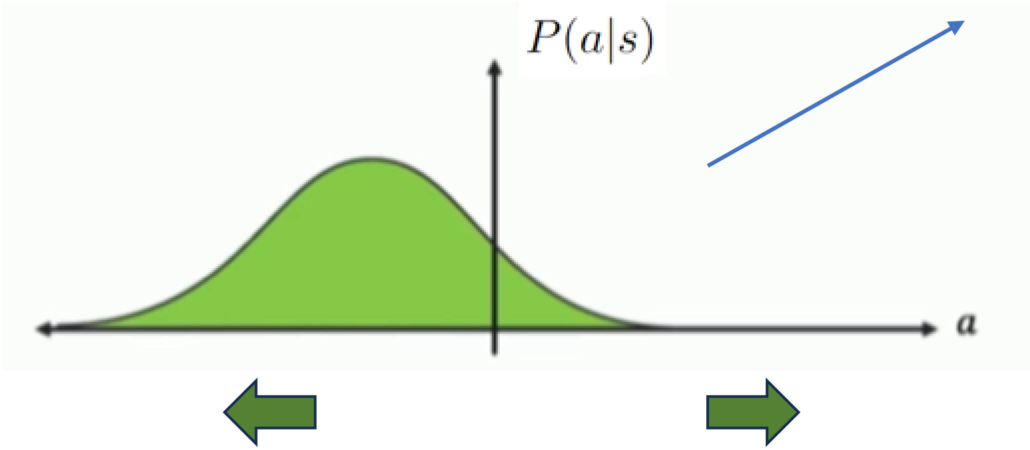
Continuous action space: how fast should I move? 7 m/s



Discrete vs Continuous Action Spaces

Discrete action space: which direction should I move?

Continuous action space: how fast should I move? 7 m/s



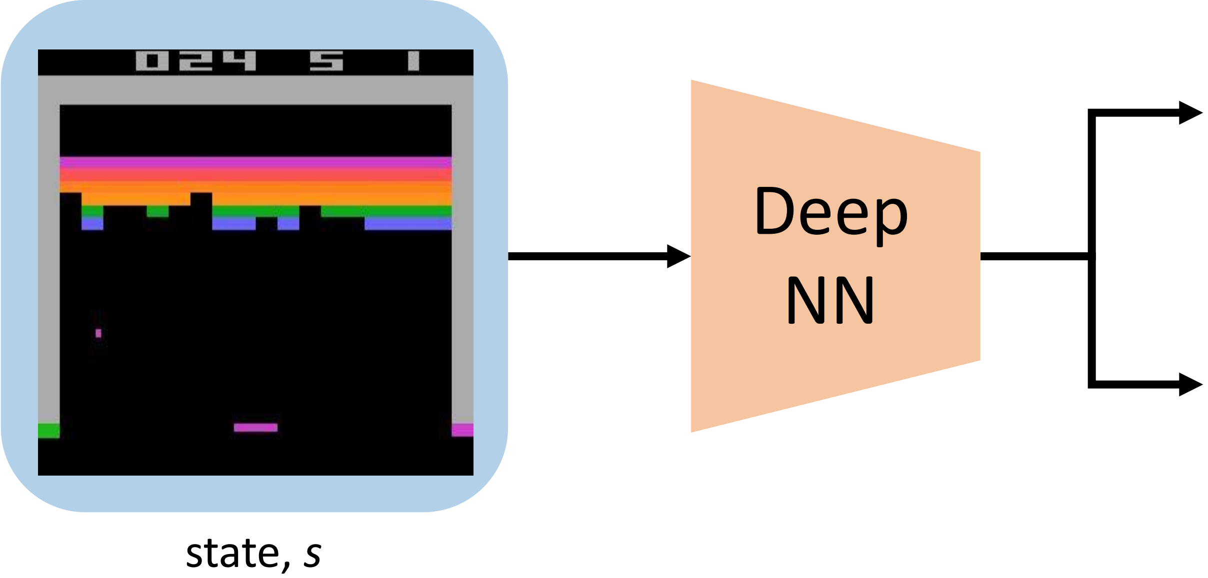
Parametrization

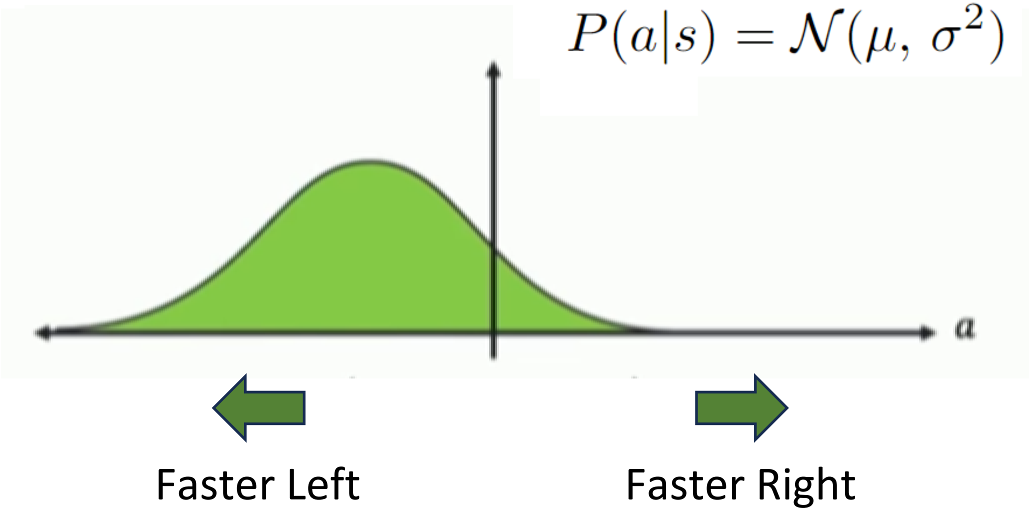
Faster Left

Faster Right

Policy Gradient (PG): Key Idea

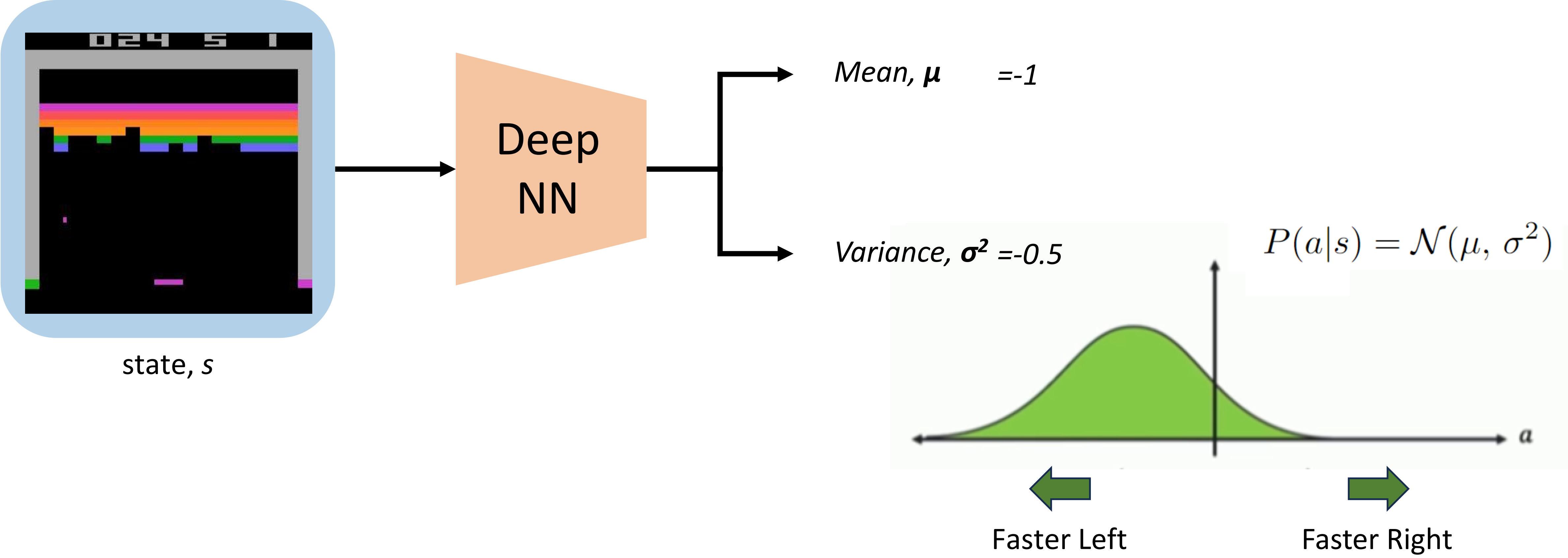
Policy Gradient: Enables modeling of continuous action space





Policy Gradient (PG): Key Idea

Policy Gradient: Enables modeling of continuous action space



Policy Gradient (PG): Key Idea

Policy Gradient: Enables modeling of continuous action space

state, *s*

Deep NN

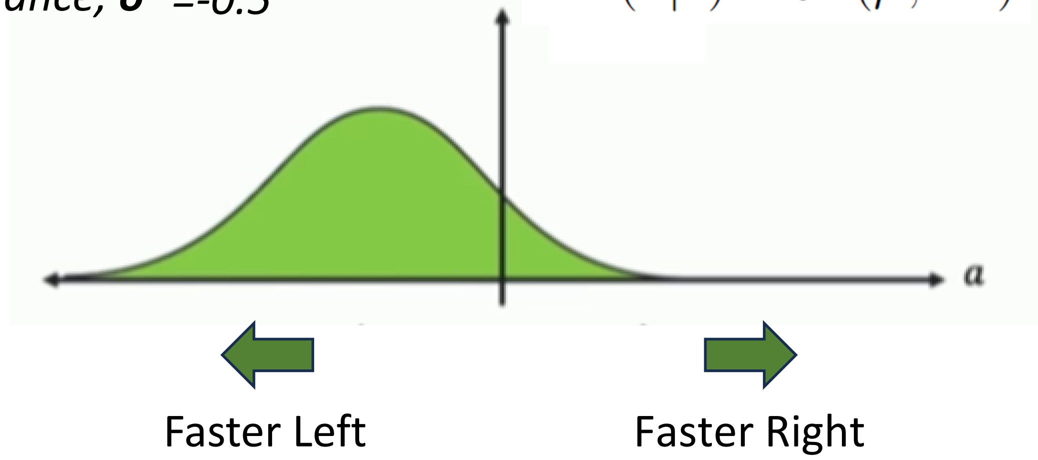
Mean, $\boldsymbol{\mu}$ =-1

Variance, $\boldsymbol{\sigma^2}$ =-0.5

$$P(a|s) = \mathcal{N}(\mu, \sigma^2)$$
$$\pi^*(s) \sim P(a|s)$$
$$= -0.8[m/s]$$

$$\int_{a=-\infty}^{\infty} P(a|s) = 1$$

$$P(a|s) = \mathcal{N}(\mu, \sigma^2)$$



ELTE | FACULTY OF INFORMATICS

Thank you for your attention!