Deep Reinforcement Learning

Deep Reinforcement Learning

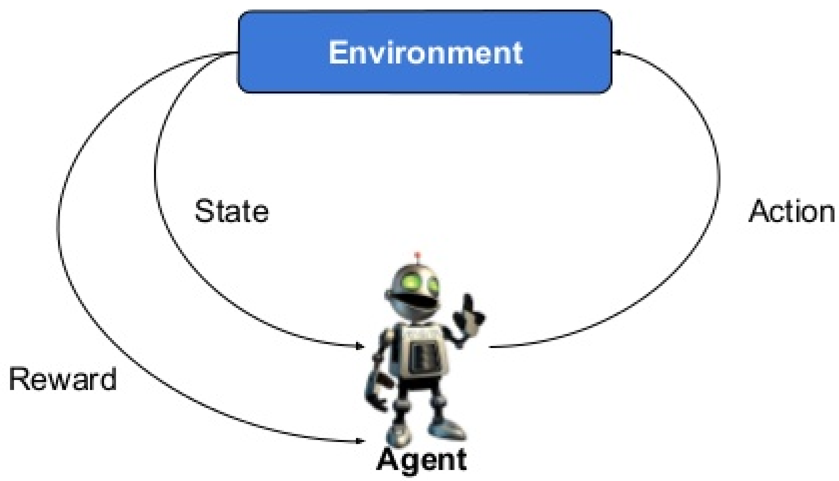
Using deep learning to improve reinforcement learning

Deep learning: Training (deep) neural networks to accurately map inputs to outputs

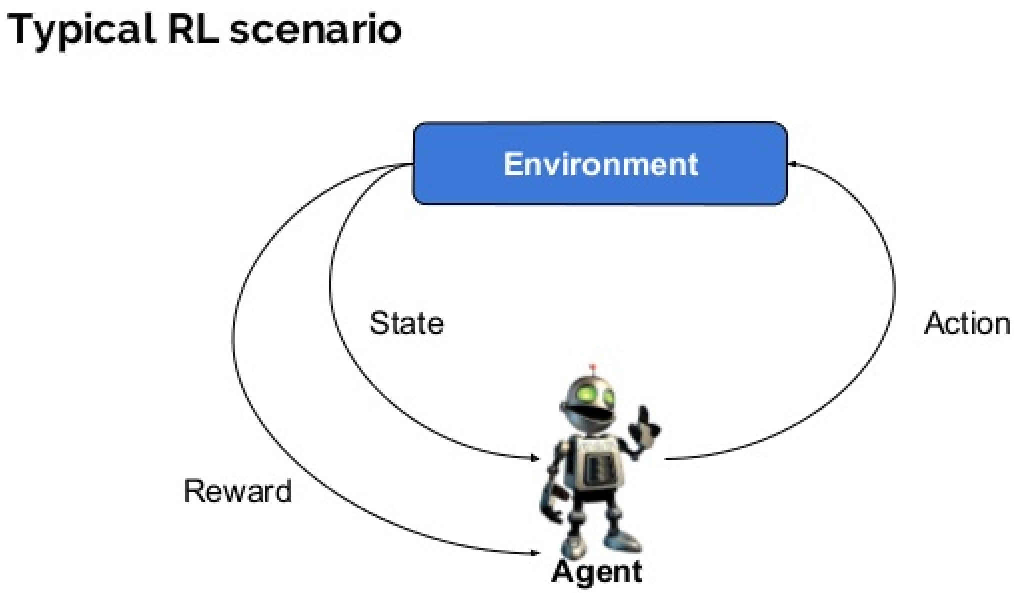
Reinforcement learning: Learning from experience

Deep Reinforcement Learning

Typical RL scenario



Deep Reinforcement Learning



$$\pi(s) : \mathbb{S} \rightarrow \mathbb{A}$$

Q-Learning

$$Q^\pi(s,a) = E_\pi\{R_t | s_t = s, a_t = a\}$$

$$= E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

$$= r(s,a) + \gamma Q^\pi(s', a')$$

Q-Learning

$$Q^\pi(s,a) = E_\pi\{R_t | s_t = s, a_t = a\}$$

$$= E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

$$= r(s,a) + \gamma Q^\pi(s', a')$$

$$\pi^*(s) = \arg\max_a Q(s,a)$$

Q-Learning

$$Q^\pi(s,a) = E_\pi\{R_t | s_t = s, a_t = a\}$$

$$= E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

$$= r(s,a) + \gamma Q^\pi(s', a')$$

$$\pi^*(s) = \arg\max_a Q(s,a)$$

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

Q-Learning

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

If $\mathbb{S}$ and $\mathbb{A}$ are finite we can simply store the q-value for each (s, a) pair in a lookup table.

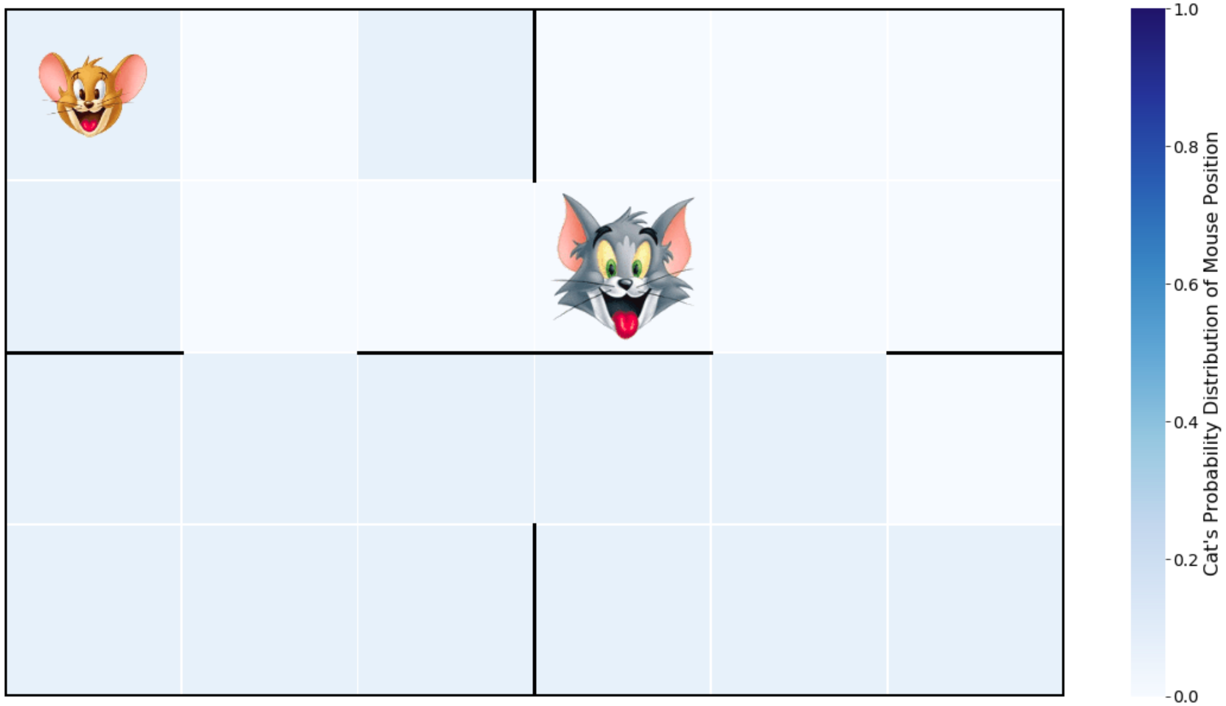
Q-Learning

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

If $\mathbb{S}$ and $\mathbb{A}$ are finite we can simply store the q-value for each (s, a) pair in a lookup table.

But if either of them are infinite, or if there are simply too many states (or actions) to store the q-values in a lookup table, we need an alternative.

Deep Q-Learning



Q-Learning

Universal Approximation Theorem

A feed-forward neural network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of R^n, under mild assumptions on the activation function.

In other words:

Simple neural networks can represent a wide variety of interesting functions…

Q-Learning

Universal Approximation Theorem

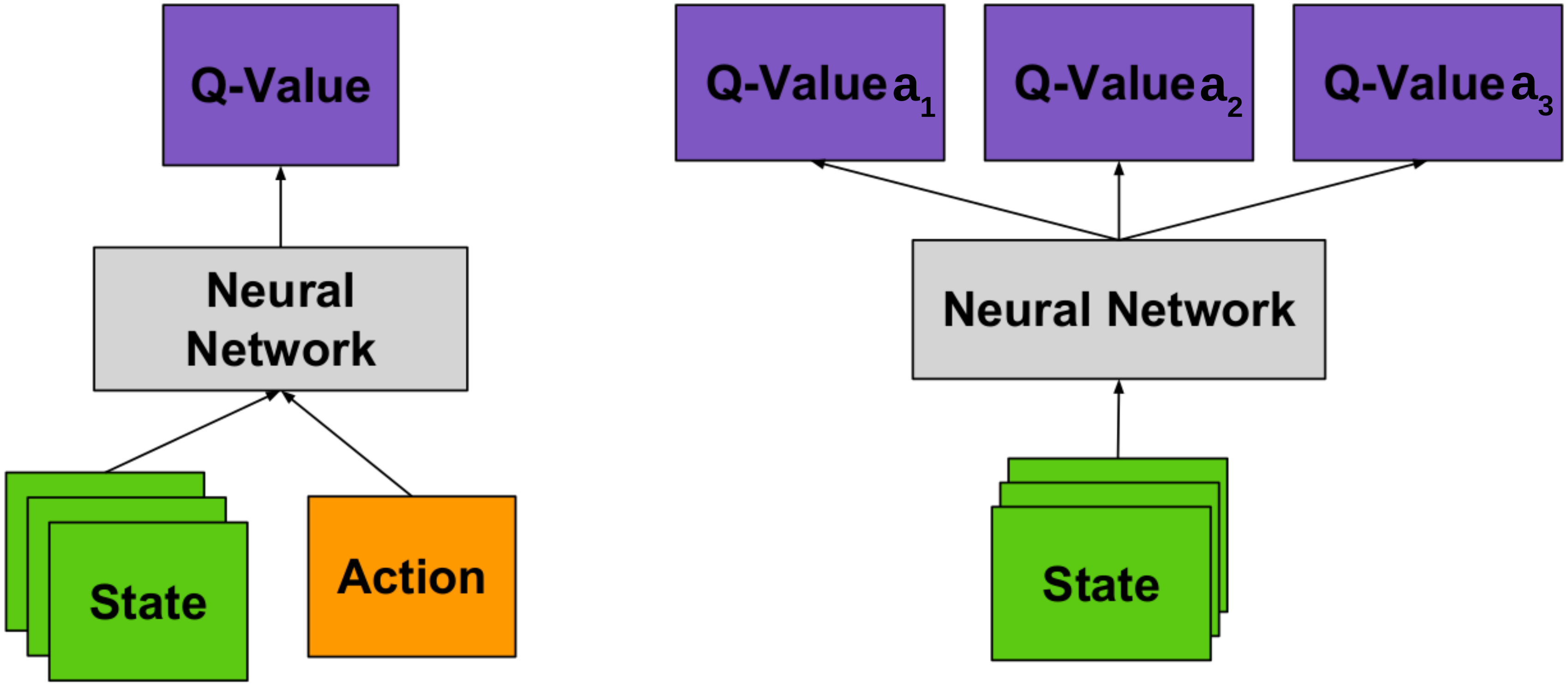
A feed-forward neural network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of R^n*, under mild assumptions on the activation function.*

In other words:

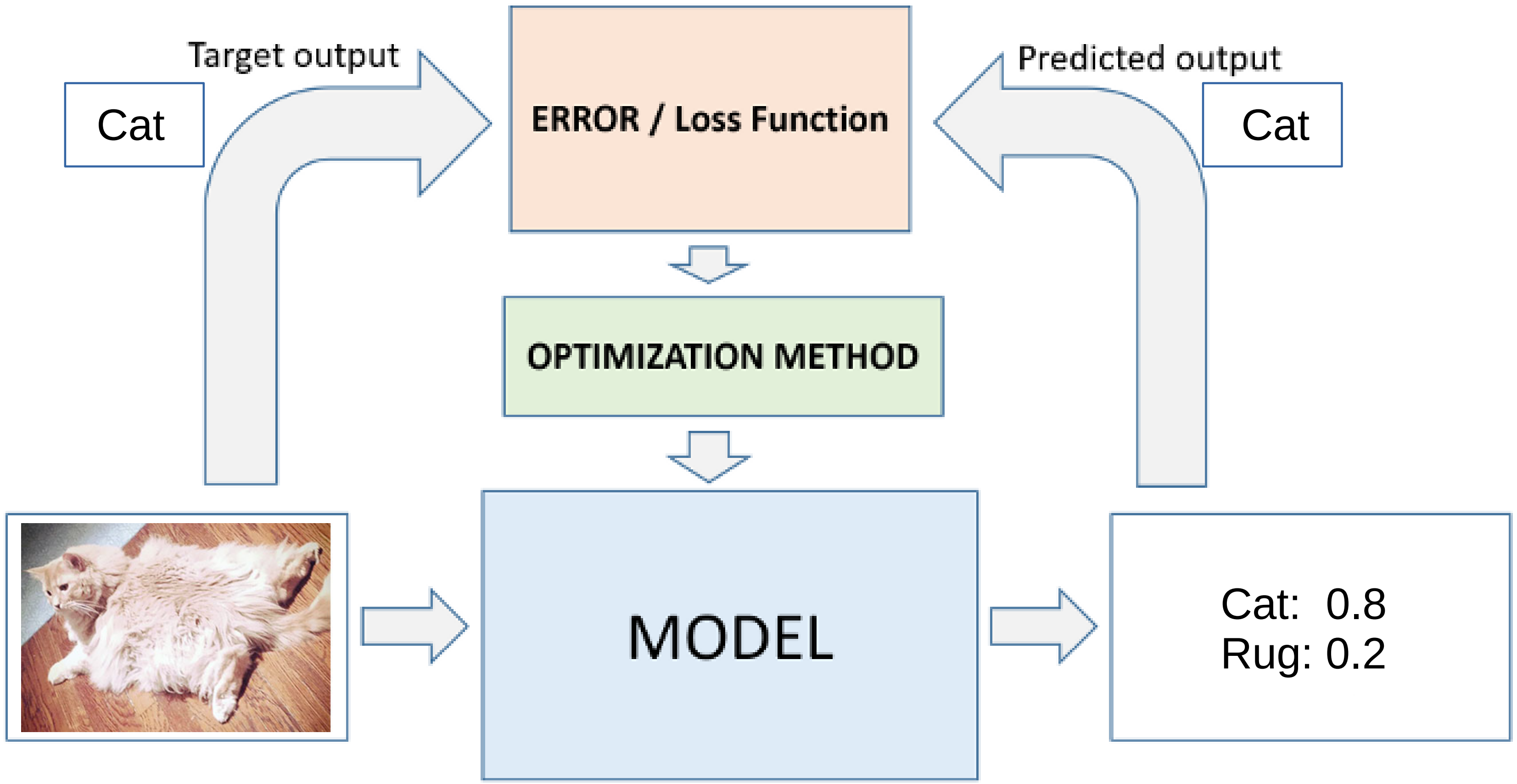
Simple neural networks can represent a wide variety of interesting functions…

... Including Q functions!

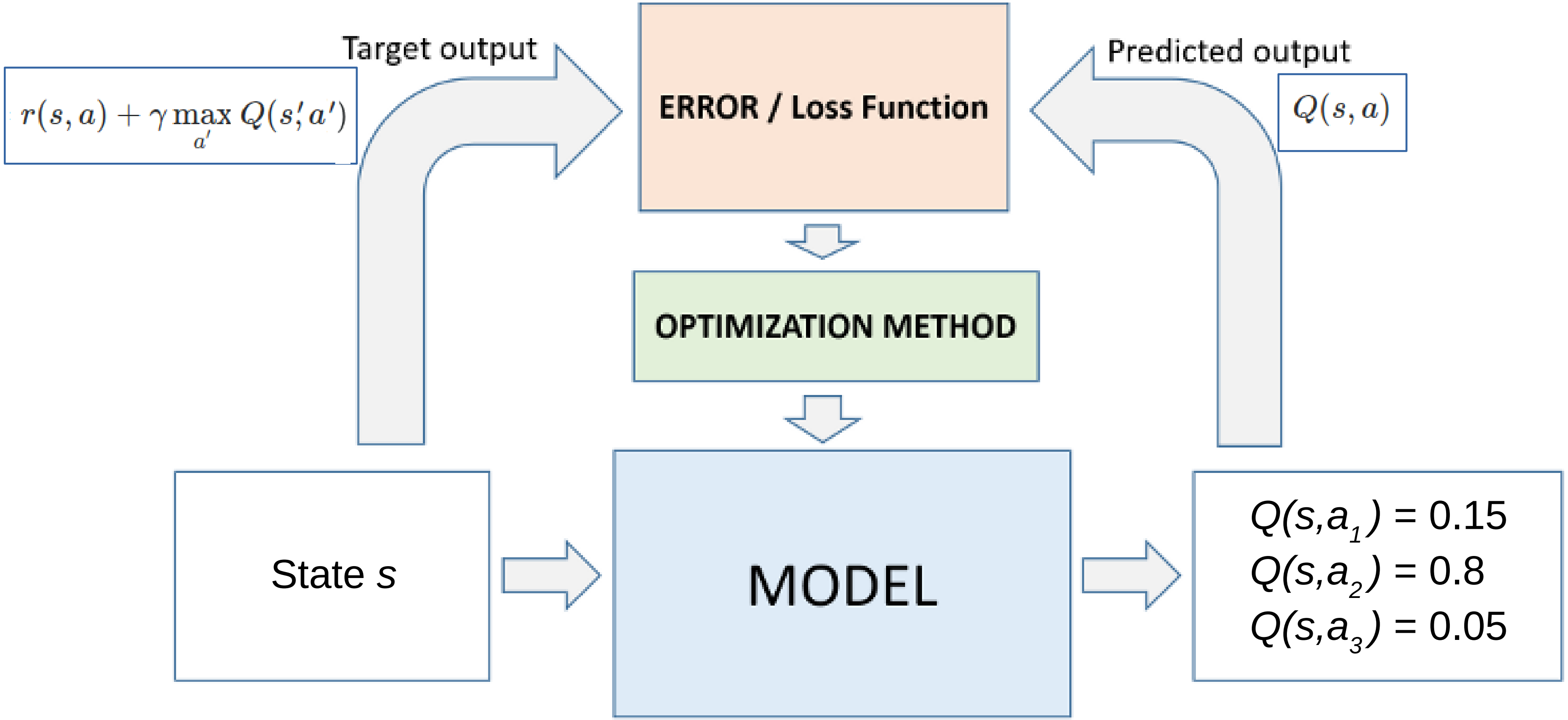
Deep Q-Learning



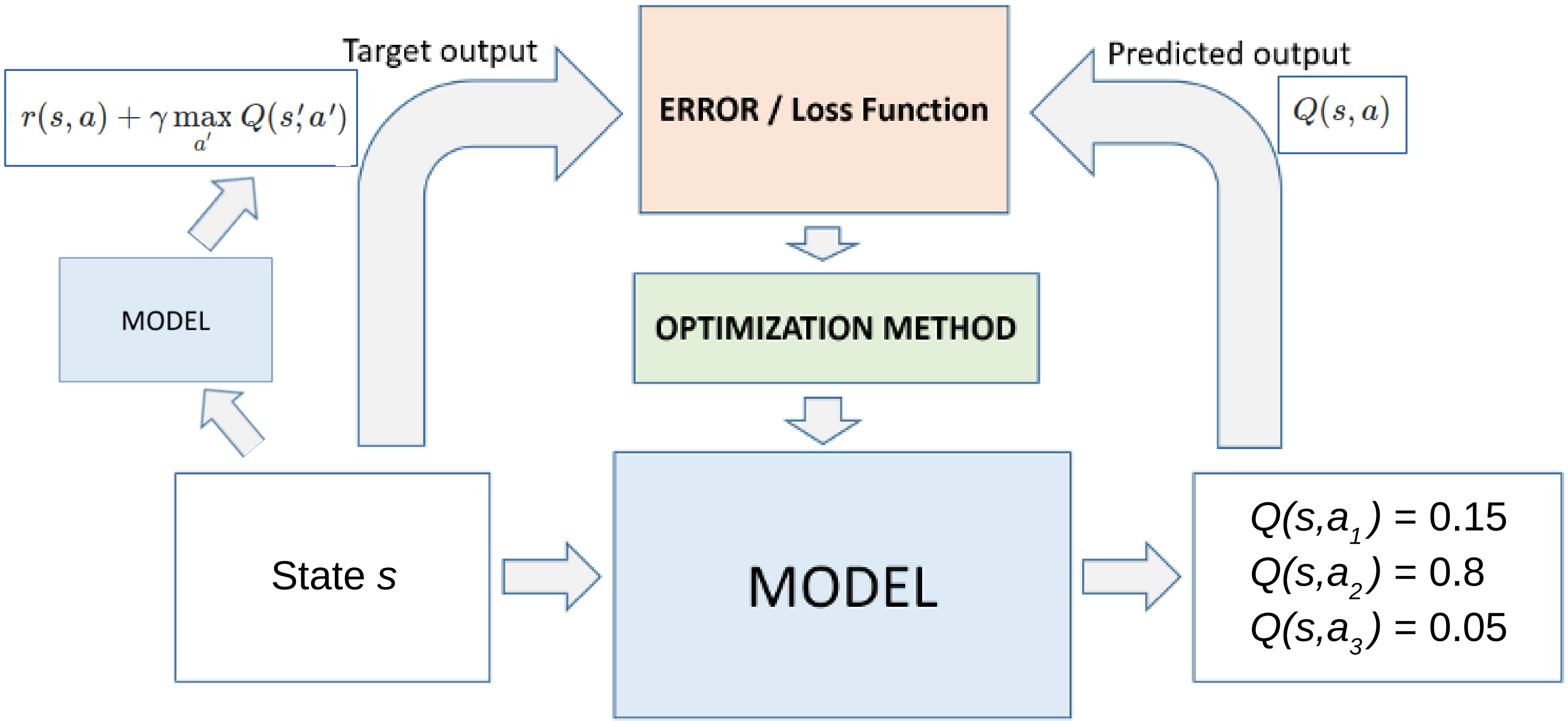
Deep Q-Learning



Deep Q-Learning

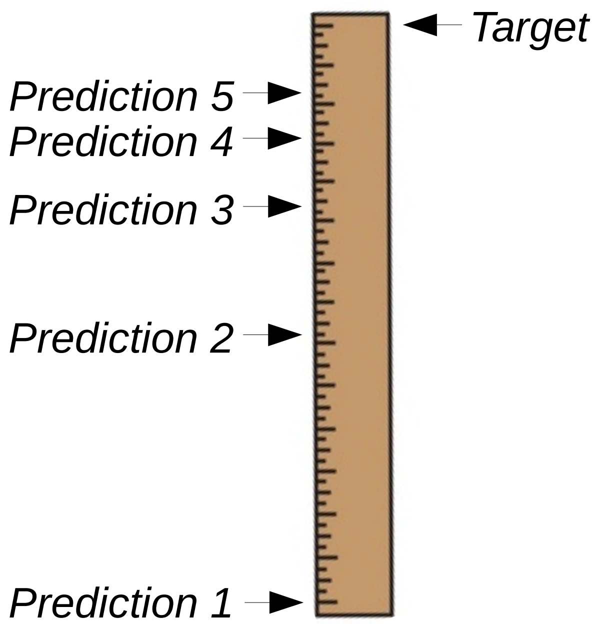


Deep Q-Learning

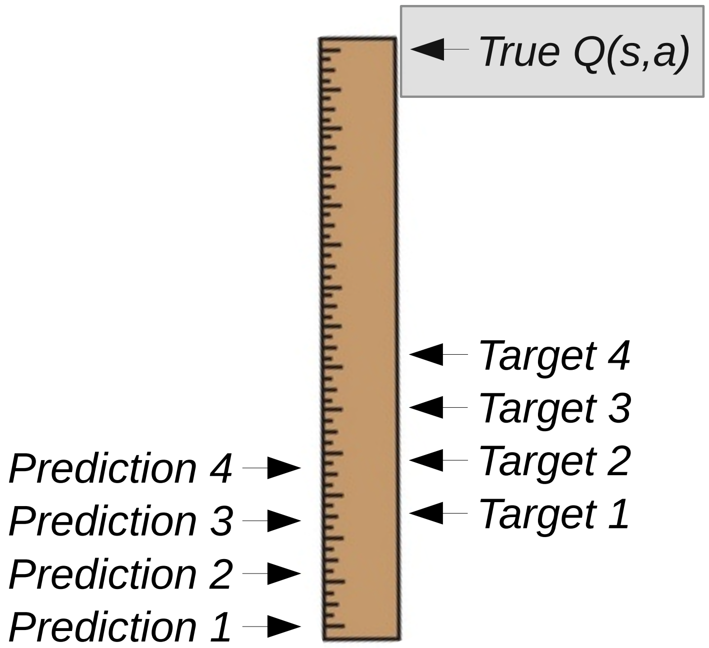


Deep Q-Learning

Traditional supervised learning

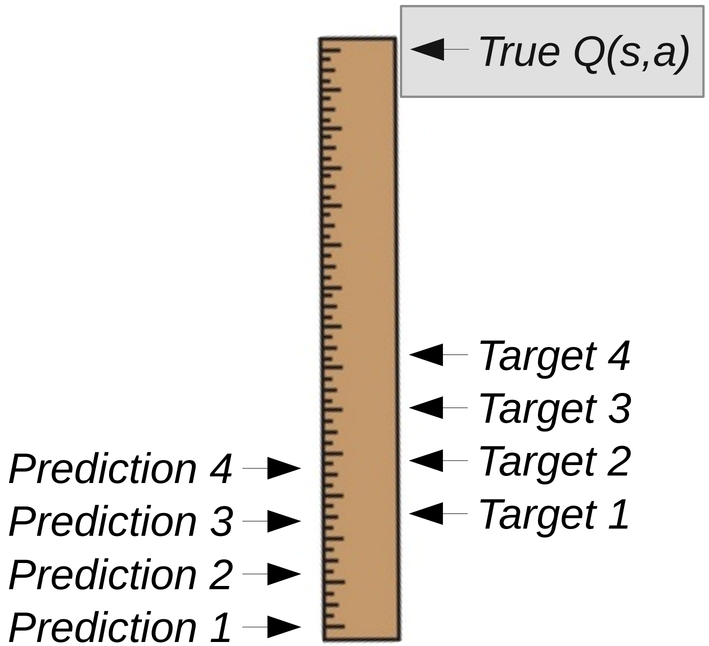


Q network (Reinforcement learning)

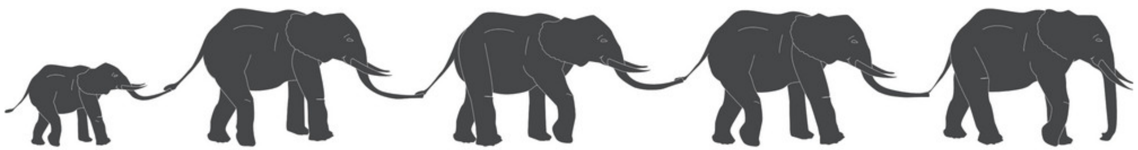


Deep Q-Learning

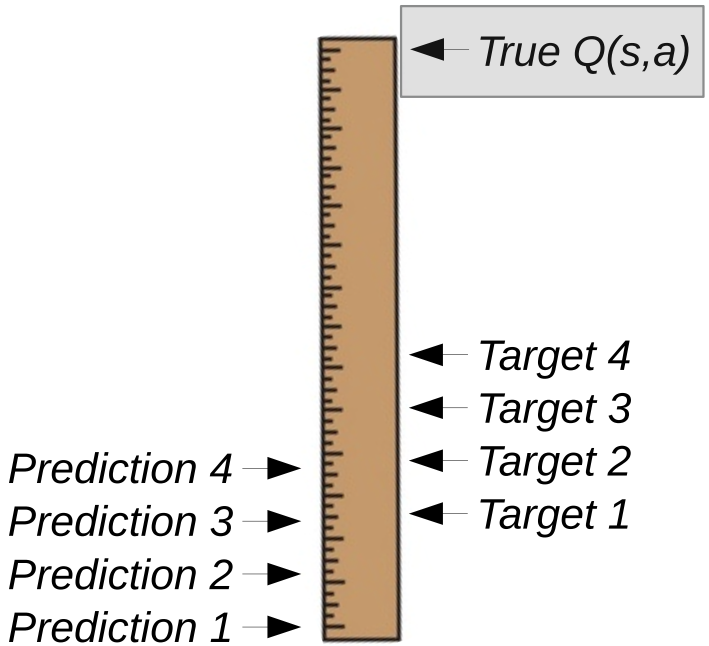
Q network (Reinforcement learning)



Deep Q-Learning

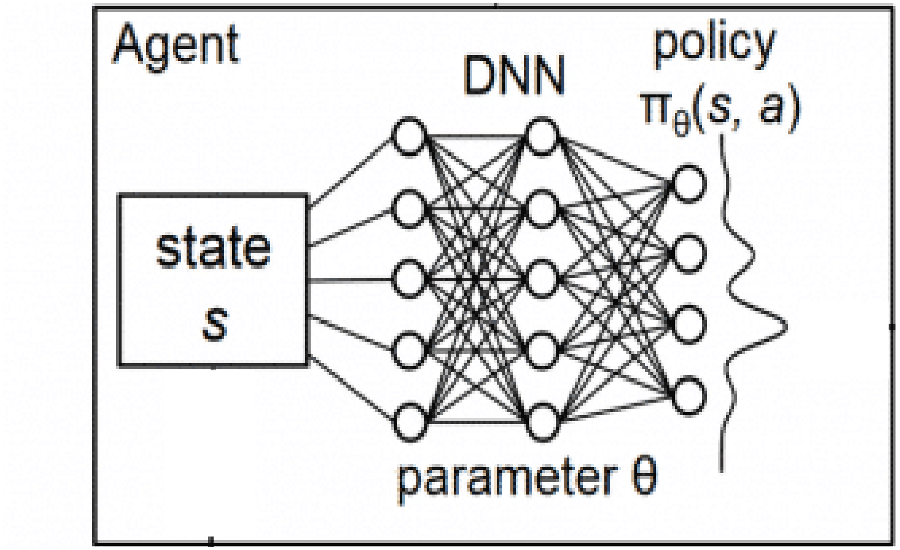


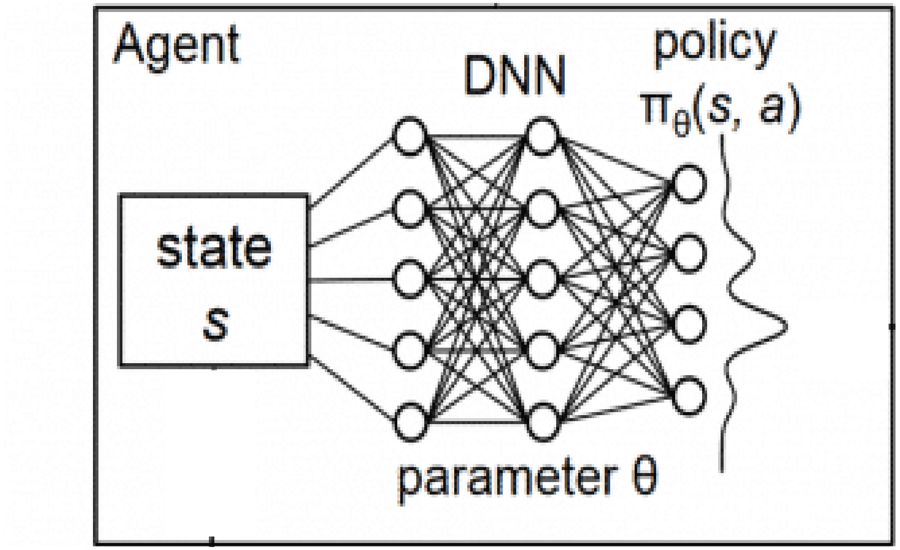
Q network (Reinforcement learning)



Challenges

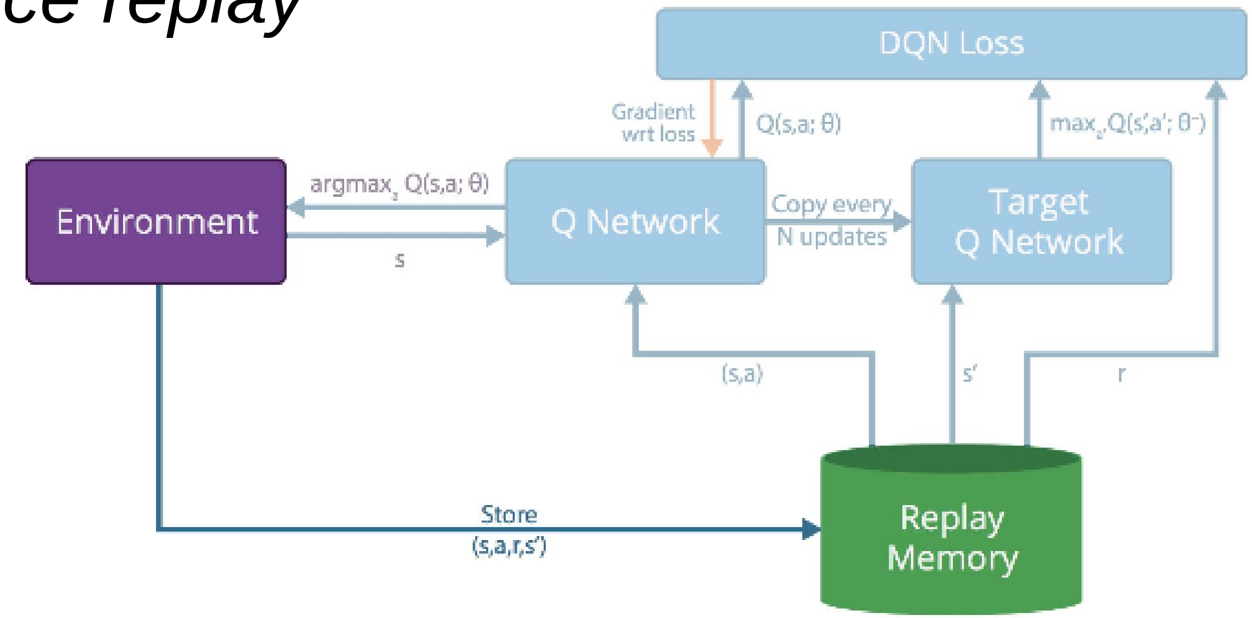
- Moving targets
  - *Two networks: behaviour and target*





Challenges

- Moving targets
  - *Two networks: behaviour and target*
- Correlated inputs (due to sequential states)
  - *Experience replay*



Challenges

- Moving targets
  - *Two networks: behaviour and target*
- Correlated inputs (due to sequential states)
  - *Experience replay*
- Markovian
  - *Add recurrency to network*