Introduction to Deep Learning

Arijit Mondal
Dept. of Computer Science & Engineering
Indian Institute of Technology Patna
arijit@iitp.ac.in

Deep Reinforcement Learning

Introduction

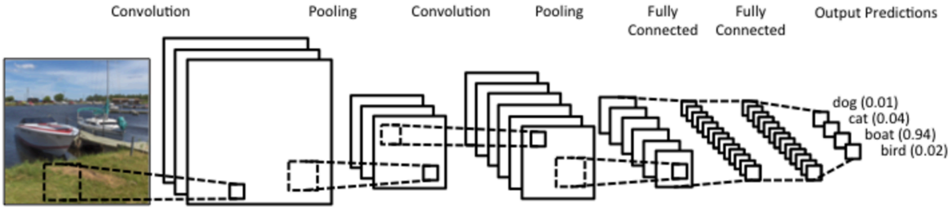
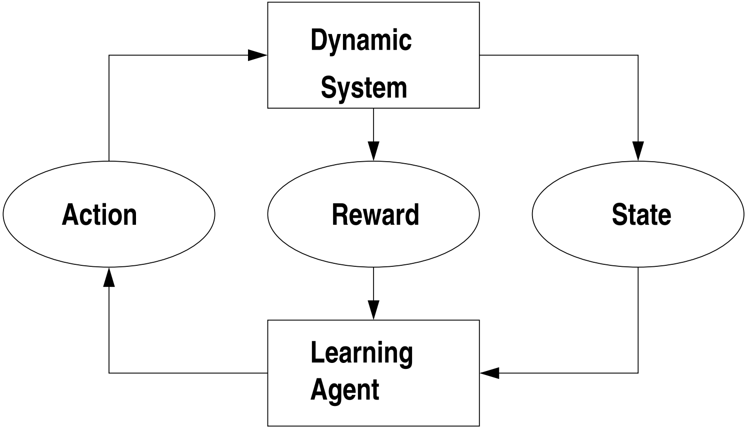


Image source: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

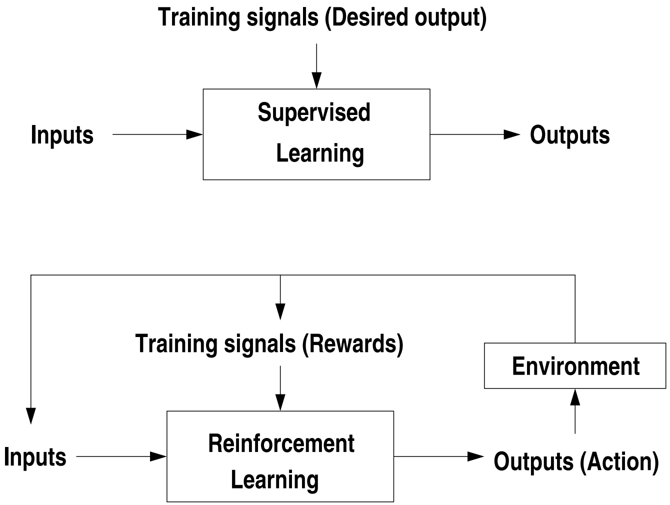
Interaction with environment



Reinforcement learning

- Set of actions that the learner will make in order to maximize its profit
- Action may not only affect the next situation but also subsequent situation
  - Trial and error search
  - Delayed reward
- A learning agent is interacting with environment to achieve a goal
- Agent needs to have idea of state so that it can take right action
- Three key aspects — **observation, action, goal**

Reinforcement vs supervised learning



Reinforcement learning

- It is different from supervised learning
  - Learning from examples provided by a knowledgeable external supervisor
  - Not adequate for learning from interaction
- In interaction problem it is often impractical to obtain examples of desired behavior that are correct and representative of all situations
- Trade-off between *exploration* and *exploitation*
  - To improve reward it must prefer effective action from the past (exploit)
  - To discover such action it has to try unselected actions (explore)
  - Exploit and exploration cannot be pursued exclusively
- Agent interacts with uncertain environment

When to use RL

- Data in the form of trajectories
- Need to make a sequence of decision
- Observe (partial, noisy) feedback to state or choice of action

Examples

- Chess player eg. games
- Robotics
- Adaptive controller
- All involve interaction between active decision making agent and its environment

Elements of RL

- Agent
- Environment
- Policy — The way agent behaves at a given time
  - Mapping of state-action pair to state
  - Can use look up table or search method
  - Core of reinforcement learning problem
- Reward function — Defines the goal in reinforcement learning problem
  - It maps state-action pair to a single number
  - Objective of RL agent is to maximize total reward
  - Defines bad or good events
  - Must be unalterable by agent, however policy can be changed

Elements of RL (contd.)

- Value function
  - Specifies what is good in long run
  - Value of a state is the total amount of reward an agent can expect to accumulate over future starting from the state
  - Indicates long term desirability of states
  - The action tries to move to a state of highest value (not highest reward)
  - Rewards are mostly given by the environment
  - Value must be estimated or reestimated from the sequence of observation
  - Need efficient method to find values
    - Evolutionary methods (genetic algorithm, simulated annealing) search directly in the space of policies without applying value function

Elements of RL (contd.)

- Model of environment
  - Mimics the behavior of environment
  - Given state and action, model might predict resultant next state and next reward
  - Every RL system uses trial and search methodology to learn

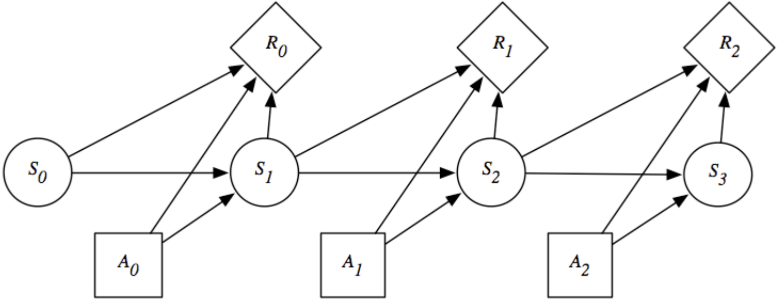
Reinforcement learning

- Learning agent tries a sequence of actions (a_t)
- Observes outcomes (state s_{t+1}, rewards r_t) of those actions
- Statistically estimated relationship between action choice and outcomes $Pr(s_t | s_{t-1}, a_{t-1})$
- Selection of policy $\pi(s)$ that optimizes selected outcome

$$\arg\max_{\pi} E_{\pi}[r_0 + r_1 + \ldots + r_T | s_0]$$

Markovian decision process

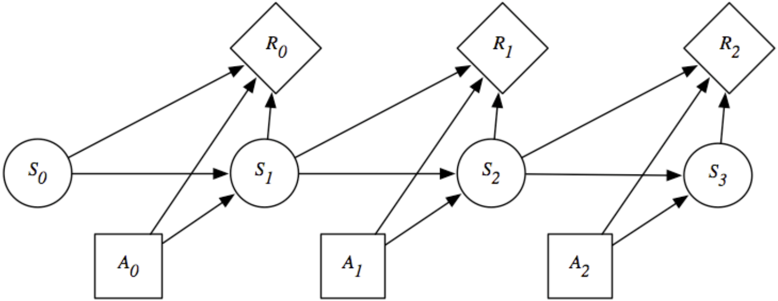
- S — set of states
- A — set of actions
- $Pr(s_t|s_{t-1}, a_{t-1})$ — Probabilistic effects
- r_t — reward function
- μ_t — initial state distribution



The Markov property

- The future state depends only on the current state

$$Pr(s_t | s_{t-1}, \ldots, s_0) = Pr(s_t | s_{t-1})$$



Utility maximization

- Let U_t be the utility for a trajectory starting from t
- Episodic tasks (eg. games)

$$U_t = r_t + r_{t+1} + r_{t+2} + \ldots + r_T$$

- Continuing tasks (eg. can run forever)

$$U_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = \sum_{k=0} \gamma^k r_{t+k}$$

- γ is known as discount factor and lies between 0 and 1
  - At each time step there is a chance of $(1 - \gamma)$ that agent dies and no reward after that
  - Inflation rate - receiving an amount of money today, the value of it tomorrow will be less by a factor of γ

Policy

- Policy defines the action selection strategy at every state

$$\pi(s, a) = P(a_t = a, s_t = s)$$

  - It can be stochastic or deterministic

- Goal is to maximize expected total reward

$$\arg\max_{\pi} E_{\pi}[r_0 + r_1 + \ldots + r_T | s_0]$$

  - There are many policies!

Example

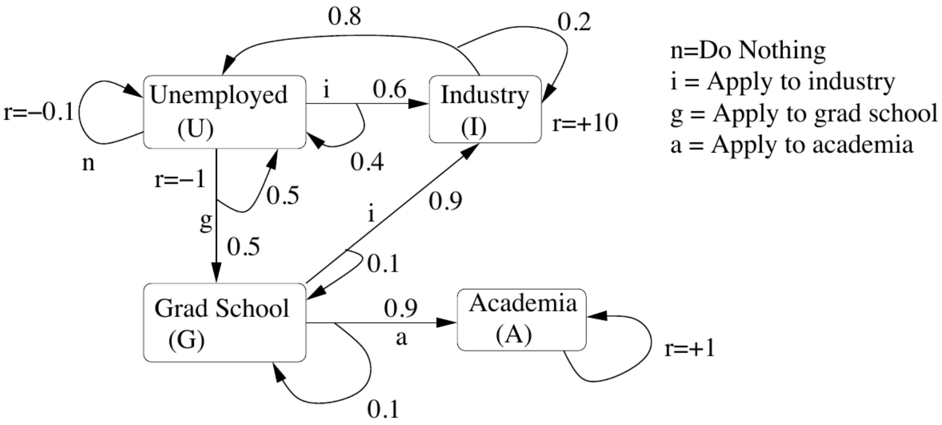


Image source: Reinforcement Learning by Joelle Pineau

Value functions

- As we are looking for best policy, it will be useful to estimate the expected return
- Good policy may be chosen by searching over the space of policies
- Value function at a state under a given policy is

$$V^{\pi}(s) \quad = \quad E_{\pi}[r_t + r_{t+1} + \ldots + r_T | s_t = s]$$

Value functions

- As we are looking for best policy, it will be useful to estimate the expected return
- Good policy may be chosen by searching over the space of policies
- Value function at a state under a given policy is

$$V^\pi(s) = E_\pi[r_t + r_{t+1} + \ldots + r_T | s_t = s]$$

$$V^\pi(s) = E_\pi[r_t | s_t = s] + E_\pi[r_{t+1} + \ldots + r_T | s_t = s]$$

Value functions

- As we are looking for best policy, it will be useful to estimate the expected return
- Good policy may be chosen by searching over the space of policies
- Value function at a state under a given policy is

$$
\begin{aligned}
V^{\pi}(s) &= E_{\pi}[r_t + r_{t+1} + \ldots + r_T | s_t = s] \\
V^{\pi}(s) &= E_{\pi}[r_t | s_t = s] + E_{\pi}[r_{t+1} + \ldots + r_T | s_t = s] \\
V^{\pi}(s) &= \sum_{a \in A} \pi(s, a) r(s, a) + E_{\pi}[r_{t+1} + \ldots + r_T | s_t = s]
\end{aligned}
$$

Value functions

- As we are looking for best policy, it will be useful to estimate the expected return
- Good policy may be chosen by searching over the space of policies
- Value function at a state under a given policy is

$$
\begin{aligned}
V^{\pi}(s) &= E_{\pi}[r_t + r_{t+1} + \ldots + r_T | s_t = s] \\
V^{\pi}(s) &= E_{\pi}[r_t | s_t = s] + E_{\pi}[r_{t+1} + \ldots + r_T | s_t = s] \\
V^{\pi}(s) &= \sum_{a \in A} \pi(s,a) r(s,a) + E_{\pi}[r_{t+1} + \ldots + r_T | s_t = s] \\
V^{\pi}(s) &= \sum_{a \in A} \pi(s,a) r(s,a) + \sum_{a \in A} \pi(s,a) \sum_{s' \in S} T(s,a,s') E_{\pi}[r_{t+1} + \ldots + r_T | s_{t+1} = s']
\end{aligned}
$$

Value functions

- As we are looking for best policy, it will be useful to estimate the expected return
- Good policy may be chosen by searching over the space of policies
- Value function at a state under a given policy is

$$
\begin{aligned}
V^{\pi}(s) &= E_{\pi}[r_t + r_{t+1} + \ldots + r_T | s_t = s] \\
V^{\pi}(s) &= E_{\pi}[r_t | s_t = s] + E_{\pi}[r_{t+1} + \ldots + r_T | s_t = s] \\
V^{\pi}(s) &= \sum_{a \in A} \pi(s,a) r(s,a) + E_{\pi}[r_{t+1} + \ldots + r_T | s_t = s] \\
V^{\pi}(s) &= \sum_{a \in A} \pi(s,a) r(s,a) + \sum_{a \in A} \pi(s,a) \sum_{s' \in S} T(s,a,s') E_{\pi}[r_{t+1} + \ldots + r_T | s_{t+1} = s'] \\
V^{\pi}(s) &= \sum_{a \in A} \pi(s,a) r(s,a) + \sum_{a \in A} \pi(s,a) \sum_{s' \in S} T(s,a,s') V^{\pi}(s')
\end{aligned}
$$

Value of policy

- Reorganizing the last expression

$$V^{\pi}(s) = \sum_{a \in A} \pi(s, a) \left(r(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^{\pi}(s') \right)$$

Value of policy

- Reorganizing the last expression

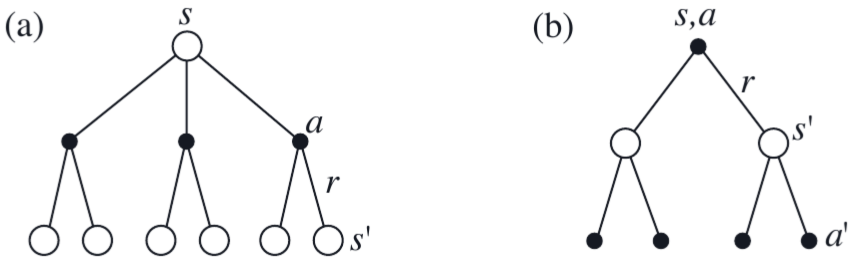
$$V^{\pi}(s) \quad = \quad \sum_{a \in A} \pi(s,a) \left(r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^{\pi}(s') \right)$$

- If we have state-action value functions

$$Q^{\pi}(s,a) \quad = \quad r(s,a) + \gamma \sum_{s' \in S} P(s' | a, s) \sum_{a'} \pi(s', a') Q^{\pi}(s', a')$$

- Known as **Bellman's equation**

Value computation



Value of policy

- State value function

$$V^{\pi}(s) \;=\; \sum_{a \in A} \pi(s,a)\left(r(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^{\pi}(s')\right)$$

- In case of finite number of states, we have a system of linear equations with unique solution to V^{π}
- Above equation can be written in matrix form as

$$\mathsf{V}^{\pi} = \mathsf{R}^{\pi} + \gamma \mathsf{T}^{\pi}\mathsf{V}^{\pi}$$

- Solution will be

$$\mathsf{V}^{\pi} = (1 - \gamma \mathsf{T}^{\pi})^{-1}\mathsf{R}^{\pi}$$

Iterative policy evaluation

- Guess initial values for $V_0(s)$
  - It can be 0
- In every iteration say k, the value function for every state will be updated as

$$V_{k+1} = R(s, \pi(s)) + \gamma \sum_{s'} T(s, \pi(s), s') V_k(s')$$

- Iteration will stop when the difference between two consecutive iteration is within a given threshold

Convergence of iterative policy evaluation

- Absolute error in after $(k+1)$th iteration

$$
\begin{aligned}
V_{k+1}(s) - V^{\pi}(s) &= |\sum_a \pi(s,a)(R(s,a) + \gamma \sum_{s'} T(s,a,s')V_k(s') \\
&\quad - \sum_a \pi(s,a)(R(s,a) - \gamma \sum_{s'} T(s,a,s')V^{\pi}(s')| \\
&\leq \gamma \sum_a \pi(s,a) \sum_{s'} T(s,a,s')|V_k(s') - V^{\pi}(s')|
\end{aligned}
$$

- If $\gamma \leq 1$, then error reduces to 0 gradually

Optimal value function

- Optimal value function may be defined as

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

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

- Any policy that achieves the optimal value function is known as optimal policy
  - Usually denoted as π^*
- Optimal value is unique
- Optimal policy is not necessarily unique

Optimal value function

- Suppose V^*, R, T, γ are known, then π^* can be determined as

$$\pi^*(s) = \arg\max_{a \in A} \left(r(s,a) + \gamma \sum_{s'} T(s,a,s') V^*(s) \right)$$

Optimal value function

- Suppose V^*, R, T, γ are known, then π^* can be determined as

$$\pi^*(s) = \arg\max_{a \in A} \left(r(s,a) + \gamma \sum_{s'} T(s,a,s') V^*(s) \right)$$

- Suppose π^*, R, T, γ are known, then V^* can be determined as

$$V^*(s) = \sum_{a \in A} \pi^*(s,a) \left(r(s,a) + \gamma \sum_{s'} T(s,a,s') V^*(s) \right)$$

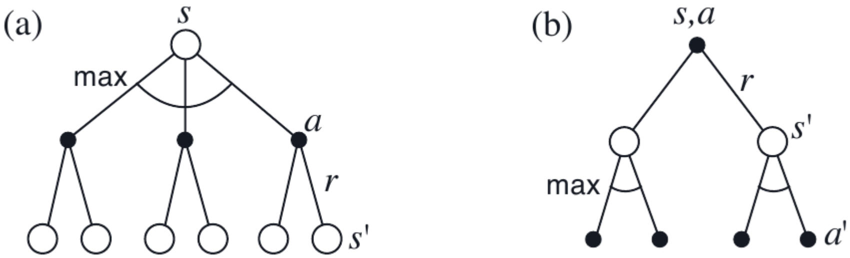
$$V^*(s) = r(s,\pi(s)) + \gamma \sum_{s'} T(s,\pi(s),s') V^*(s')$$

Optimal value function

- For state-action pair

$$Q^*(s,a) \quad = \quad r(s,a) + \gamma \sum_{s' \in S} P(s'|a,s) \max_{a'} Q^*(s',a')$$

Optimal value computation



Recycling Robot

- A robot does one of the following at each time step
  - Actively search for a can
  - Remain stationary and wait for someone to bring a can
  - Go back to home base to recharge battery

Recycling Robot: Transition relation

s	s'	a	$p(s' \mid s, a)$	$r(s, a, s')$
high	high	search	α	r_{search}
high	low	search	$1-\alpha$	r_{search}
low	high	search	$1-\beta$	-3
low	low	search	β	r_{search}
high	high	wait	1	r_{wait}
high	low	wait	0	r_{wait}
low	high	wait	0	r_{wait}
low	low	wait	1	r_{wait}
low	high	recharge	1	0
low	low	recharge	0	0

Example

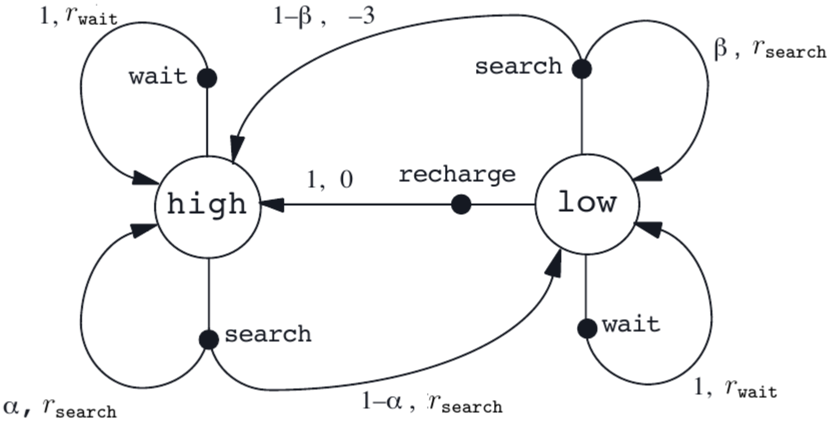


Image source: Reinforcement Learning by Andrew Barto and Richard S. Sutton

Optimal value computation

- For recycling robot

$$V^*(h) = \max \left\{ \begin{array}{l} p(h|h,s)[r(h,s,h) + \gamma V^*(h)] + p(l|h,s)[r(h,s,l) + \gamma V^*(l)], \\ p(h|h,w)[r(h,w,h) + \gamma V^*(h)] + p(l|h,w)[r(h,w,l) + \gamma V^*(l)] \end{array} \right\}$$

$$V^*(h) = \max\{r_s + \gamma[\alpha V^*(h) + (1-\alpha)V^*(l)],\ r_w + \gamma V^*(h)\}$$

$$V^*(l) = \max \left\{ \begin{array}{l} \beta r_s - 3(1-\beta) + \gamma[(1-\beta)V^*(h) + \beta V^*(l)] \\ r_w + \gamma V^*(l), \\ \gamma V^*(h) \end{array} \right\}$$

Finding a good policy (iterative approach)

- Start with an initial policy π_0
- Repeat the following
  - Determine the V^π using policy evaluation
  - Determine a new policy π' which is greedy with respect to V^π
- Terminate when $\pi = \pi'$

Finding a good policy (iterative approach)

- Start with an initial value $V_0(s)$
- In every iteration, update the value function

$$V_k(s) = \max_{a \in A} \left(R(s,a) + \gamma \sum_{s'} T(s,a,s') V_{k-1}(s') \right)$$

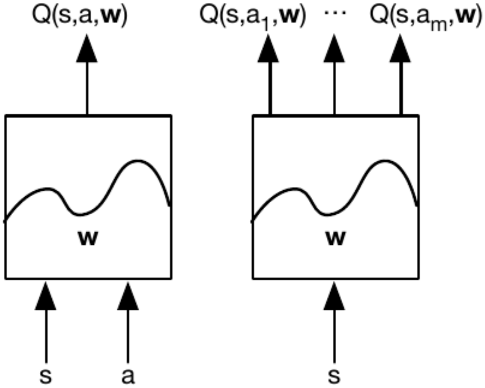
- Stop when maximum value change between iterations is below threshold
- The algorithm converges to the true value of V^*

Approaches to RL

- Value based RL
  - Estimate the optimal value function $Q^*(s, a)$
  - The maximum value that can be achieved under any policy
- Policy based RL
  - Look for optimal policy π^*
  - Policy achieving maximum future reward
- Model based RL
  - A model of the environment is developed
  - Plan is made using the model

Q-Networks

- Represent value function by Q-network with weights w, $Q(s, a, w) \approx Q^*(s, a)$



Q learning

- Optimal Q-values should obey Bellman equation

$$Q^*(s,a) = E_{s'}\left[r + \gamma \max_{a'} Q^*(s',a')|s,a\right]$$

- Right hand side may be treated as target
- Minimize MSE loss by SGD

$$l = \left(r + \gamma \max_{a'} Q(s',a',\mathrm{w}) - Q(s,a,\mathrm{w})\right)^2$$

- Can diverge using neural networks because of
  - Correlations between samples
  - Non-stationary targets

Deep Q network

- Data set are generated from agents own experience

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
$\ldots$
$s_t, a_t, r_{t+1}, s_{t+1}$

- Sample experience from data set and apply update

$$l = \left(r + \gamma \max_{a'} Q(s', a', \mathrm{w}^-) - Q(s, a, \mathrm{w})\right)^2$$

Deep reinforcement learning

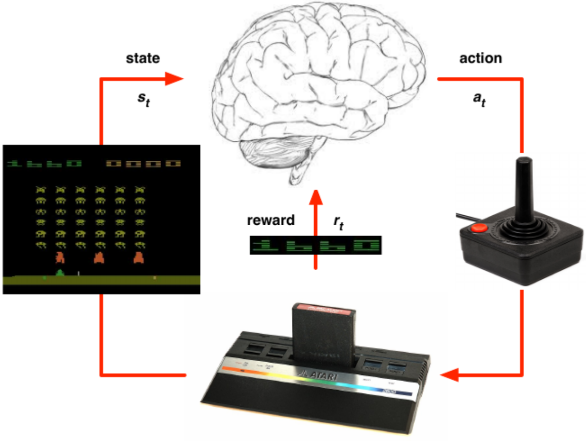


Image source:Deep reinforcement learning by David Silver

DQN

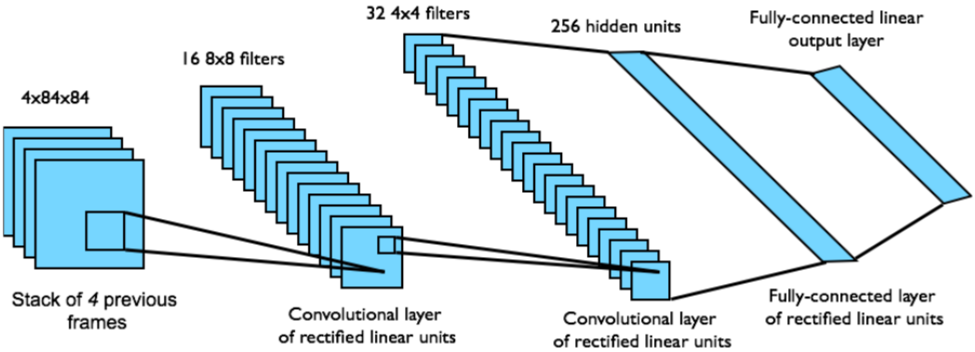


Image source:Deep reinforcement learning by David Silver

References

- *Reinforcement Learning: An Introduction* by Andrew Barto and Richard S. Sutton
- *Human-level control through deep reinforcement learning* by Deep Mind, Google