Deep Reinforcement Learning: An Introduction

Abdelrahman Khaled and Amr Khalil

Machine Learning Community
German University in Cairo

August 1, 2019

Outline

Reinforcement Learning

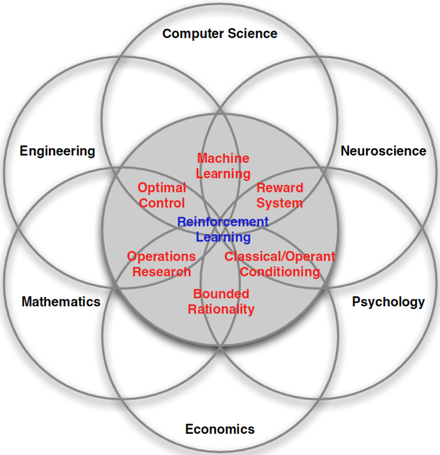
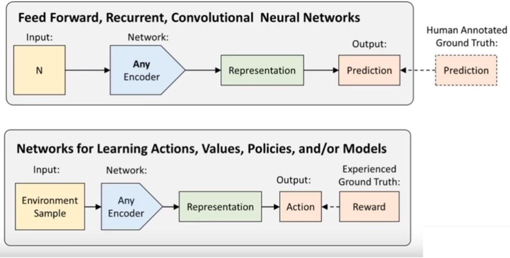


Figure: Venn Diagram from David Silver's UCL slides

Reinforcement Learning

Definition

Reinforcement learning is a training method based on rewarding desired behaviors and/or punishing undesired ones.
-*Search Enterprise AI*



Reinforcement Learning

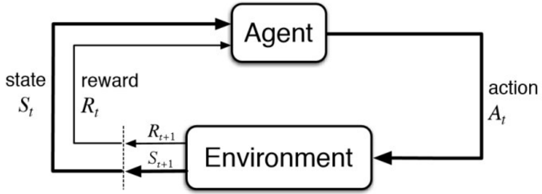


Figure: The agent-environment cycle. *Image source*

Reinforcement Learning

A typical reinforcement learning problem is different from a traditional machine learning problem in the sense that:

1. There is no predefined data set.
2. Sequence of inputs and time between inputs matters.
3. Agent's actions often affect the upcoming inputs.
4. Pre-processing of data is done online.

Reinforcement Learning

A typical reinforcement learning problem is different from a traditional machine learning problem in the sense that:

1. There is no predefined data set.
2. Sequence of inputs and time between inputs matters.
3. Agent's actions often affect the upcoming inputs.
4. Pre-processing of data is done online.

Offline RL

Reinforcemet Learning can be done offline by collecting data about the environment over a set period of time to create a dataset. However this loses the effect of the agent learning as it experiences the environment.

Frozen Lake

the game consists of a grid, some cells in the grid are terminal nodes that will reward you or punch you if you step in them. each time you can take action of moving up,down,left,right and depending on your action you will either move in the direction you choose or as the grid is a frozen lake you will move perpendicular on the direction you choose depending on some noise variable.

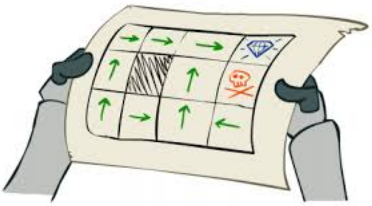


Figure: Frozen Lake

Environment

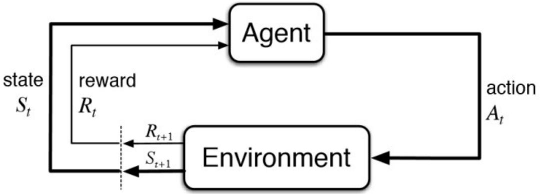


Figure: The agent-environment cycle. *Image source*

Environment

Definition

Markov Decision Processes a mathematical framework for modeling decision making in situations where outcomes are partly random.

Markov Decision Processes can be considered as a tuple

Environment

Definition

Markov Decision Processes a mathematical framework for modeling decision making in situations where outcomes are partly random.

Markov Decision Processes can be considered as a tuple

1. Set of states S

Environment

Definition

Markov Decision Processes a mathematical framework for modeling decision making in situations where outcomes are partly random.

Markov Decision Processes can be considered as a tuple

1. Set of states S
2. Set of actions A

Environment

Definition

Markov Decision Processes a mathematical framework for modeling decision making in situations where outcomes are partly random.

Markov Decision Processes can be considered as a tuple

1. Set of states S
2. Set of actions A
3. Transition function T $P(s'|s,a)$

Environment

Definition

Markov Decision Processes a mathematical framework for modeling decision making in situations where outcomes are partly random.

Markov Decision Processes can be considered as a tuple

1. Set of states S
2. Set of actions A
3. Transition function T $P(s'|s,a)$
4. Reward function $R(s,a,s')$

Environment

Definition

Markov Decision Processes a mathematical framework for modeling decision making in situations where outcomes are partly random.

Markov Decision Processes can be considered as a tuple

1. Set of states S
2. Set of actions A
3. Transition function T $P(s'|s, a)$
4. Reward function $R(s, a, s')$
5. Start state $s0$

Environment

Definition

Markov Decision Processes a mathematical framework for modeling decision making in situations where outcomes are partly random.

Markov Decision Processes can be considered as a tuple

1. Set of states S
2. Set of actions A
3. Transition function T $P(s'|s,a)$
4. Reward function $R(s,a,s')$
5. Start state $s0$
6. Discount factor γ

Frozen Lake

Markov Decision Processes

1. Set of states S
2. Set of actions A
3. Transition function T $P(s'|s,a)$
4. Reward function $R(s,a,s')$
5. Start state $s0$
6. Discount factor γ

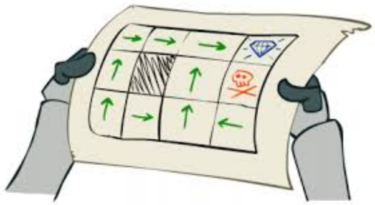
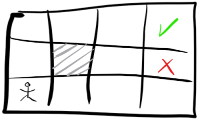


Figure: Frozen Lake

Frozen Lake

frozen lake



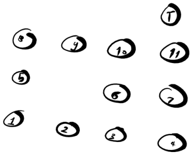
probability

$P(s' \mid s, a)$

$P(2 \mid 1, R) = 80\%$

if the noise $= 0,2$

States



actions

→ Right

← Left

↑ Up

↓ down

o Exit

Transition function

$$T(1,R) = \begin{cases} 2, & 80\% \\ 5, & 10\% \\ 1, & 10\% \end{cases}$$

$T(11, \text{Exit}) = T$

discount factor $0 < \gamma < 1$

Reward function

$R(11, \text{Exit}, T) = 1$

$R(10, R, 11) = 0$

$R(7, \text{Exit}, T) = -1$

Terminology

Terminology

Definition

A reward R_t is a number the environment provides to the agent at timestep t. It can either be positive (positive reward) or negative (punishment or negative reward). A reward of 0 is the same as no reward.

Terminology

Definition

A reward R_t is a number the environment provides to the agent at timestep t. It can either be positive (positive reward) or negative (punishment or negative reward). A reward of 0 is the same as no reward.

Definition

An episode E is everything involved in the agent-environment cycle from start until termination.

Value Iteration

Value Iteration

Definition

A policy π is a function that maps a state to an action. $(\pi : \mathcal{S} \longrightarrow \mathcal{A})$. An optimal policy is usually denoted as π^*.

Value Iteration

Definition

A policy π is a function that maps a state to an action. $(\pi : \mathcal{S} \longrightarrow \mathcal{A})$. An optimal policy is usually denoted as π^*.

Definition

A value function V maps a state to a real number that signifies the intrinsic goodness of that state. $(V : \mathcal{S} \longrightarrow \mathbb{R})$. An optimal value function is usually denoted as V^*.

$$V^*(s_t) = Max_\pi E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | \pi, s_t = s0]$$

Frozen Lake

Reward Function as a Look up table

0	0	0	+1
0		0	-1
0	0	0	0

with noise = 0 and $\gamma = 1$

$V(3,4) = 1$

$V(3,3) = 1$

$V(2,3) = 1$

$V(1,1) = 1$

State-value as a look up table

	1	2	3	4
3	$V(3,1)$	$V(3,2)$	$V(3,3)$	$V(3,4)$
2	$V(2,1)$		$V(2,3)$	$V(2,4)$
1	$V(1,1)$	$V(1,2)$	$V(1,3)$	$V(1,4)$

with noise = 0 and $\gamma = 0{,}9$

$V(3,4) = 1$

$V(3,3) = 0{,}9$

$V(2,3) = 0{,}9 * 0{,}9$

$V(1,1) = 0{,}9 * 0{,}9 * 0{,}9 * 0{,}9 * 0{,}9$

with noise = 0,2 and $\gamma = 0{,}9$

$V(3,3) = 1 * {,}8 + {,}1 * V(2,3) + {,}1 * V(3,3)$

problem

Value Iteration

$\forall s, V_0^*(s) = 0;$

for $k = 1, 2.., H$ **do**

  for *Each s in S* **do**

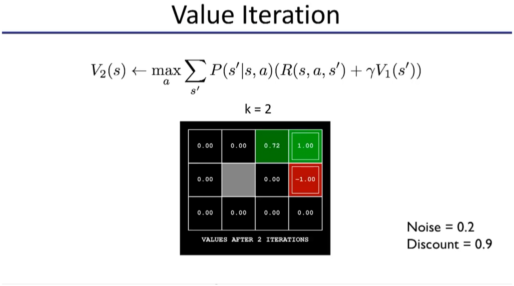
    $V_k^*(s) = max_a \sum_{s'} P(s`|s,a)(R(s,a,s`)) + \gamma V_{k-1}^*(s`);$

    $\pi_k^*(s) = argmax_a \sum_{s'} P(s`|s,a)(R(s,a,s`)) + \gamma V_{k-1}^*(s`);$

  end

end

Value Iteration



Video Time

`https://youtu.be/qaMdN6LS9rA?t=1517`

Q Learning

Definition

A quality function Q combines both the policy and value functions together. It maps a state and action pair to a real number that signifies how good it is to take the action in that state. $(Q : \mathcal{S} \times \mathcal{A} \longrightarrow \mathbb{R})$. An optimal quality function is usually denoted as Q^*.

Q Learning

Definition

Bellman Equation

$$Q^*(s,a) = \sum_{s'} P(s`|s,a)(R(s,a,s`)) + \gamma max_{a`} Q^*(s`,a))$$

Q Learning

Definition

Bellman Equation

$$Q^*(s,a) = \sum_{s'} P(s`|s,a)(R(s,a,s`)) + \gamma max_{a`} Q^*(s`,a))$$

Definition

Iteration Equation

$$Q^*_{k+1}(s,a) = \sum_{s'} P(s`|s,a)(R(s,a,s`)) + \gamma max_{a`} Q^*_k(s`,a))$$

Q Learning

In the end, all that really matters is the quality function. If we know Q, then we can get both V and π.

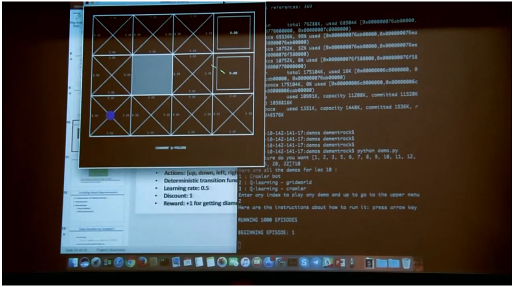
$$V(S_t) = \max_{A_t} Q(S_t, A_t)$$

$$\pi(S_t) = \arg\max_{a} Q(S_t, a)$$

Q Learning (sampling or tabular)

for *s in S* **do**
  for *a in A* **do**
    Q(s,a)=initialization;
  end
end
s=s_0;
for *k=1,2.....convergence* **do**
  a=sampleAction();
  s'=runAction(a);
  if $s' = TERMINAL$ **then**
    Q(s,a) = R(s,a,s');
    s=s_0;
  else
    $Q(s,a) = Q(s,a) + \alpha[R(s,a,s') + \gamma \max_a Q(s',a) - Q(s,a)]$;
  end

Q Learning (sampling or tabular)



Video Time

`https://youtu.be/q0-HUo0Ls04?t=1916`

Exploration vs Exploitation

inisilize
$\epsilon = 1$;
$\epsilon_{min} = 0.01$;
$\lambda = 0.001$;

Function `sampleAction()`**:**
    if $\epsilon < \epsilon_{min}$ **then**
        $\epsilon = \epsilon_{min}$;
    end
    r = random();
    if $r < \epsilon$ **then**
        ϵ-= λ;
        return randomAction();
    else
        return $\arg\max_a Q(S_t, a)$;
    end
End Function

Deep Q-Learning

Q-learning is a technique in reinforcement learning where the agent attempts to fill a table with the correct value for each state-action pair. It's usually initialized at 0s and filled using the following formula:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Deep Q-Learning

Sometimes regular Q-learning is just not feasible, but that's mainly because of the fact that the state space can become extremely large. For example:

Deep Q-Learning

Sometimes regular Q-learning is just not feasible, but that's mainly because of the fact that the state space can become extremely large. For example:

- Chess has about 10^{20} possible states. If each state only took 10 bytes of data, the table would be more than 800 exabytes large. All of the memory space in existence currently cannot hold that much data.

Deep Q-Learning

Sometimes regular Q-learning is just not feasible, but that's mainly because of the fact that the state space can become extremely large. For example:

- Chess has about 10^{20} possible states. If each state only took 10 bytes of data, the table would be more than 800 exabytes large. All of the memory space in existence currently cannot hold that much data.
- Tetris has about 10^{64} possible states. (These can be reduced down to around 10^{22} states, but that is still a large number).

Deep Q-Learning

Sometimes regular Q-learning is just not feasible, but that's mainly because of the fact that the state space can become extremely large. For example:

- Chess has about 10^{20} possible states. If each state only took 10 bytes of data, the table would be more than 800 exabytes large. All of the memory space in existence currently cannot hold that much data.
- Tetris has about 10^{64} possible states. (These can be reduced down to around 10^{22} states, but that is still a large number).
- A self-driving car has a *continuous* state space, and thus the number of states is possibly infinite.

Deep Q-Learning

Deep Q-learning uses a neural network as a function approximator for learning the Q function.

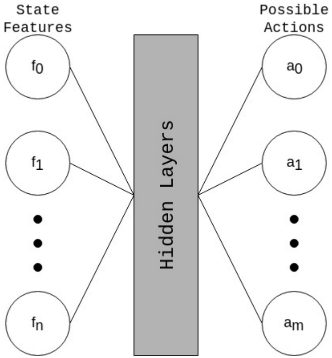


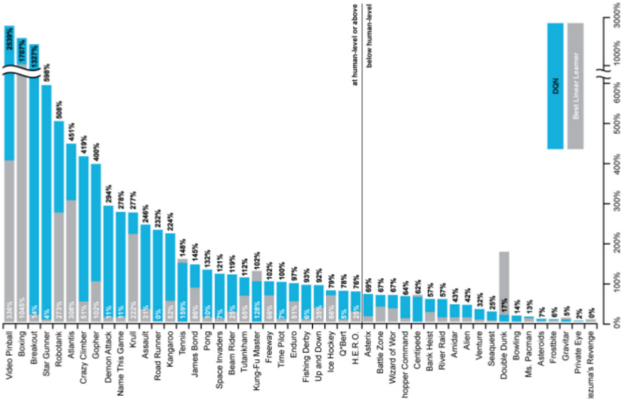
Figure: A deep Q-network

Success Stories in RL

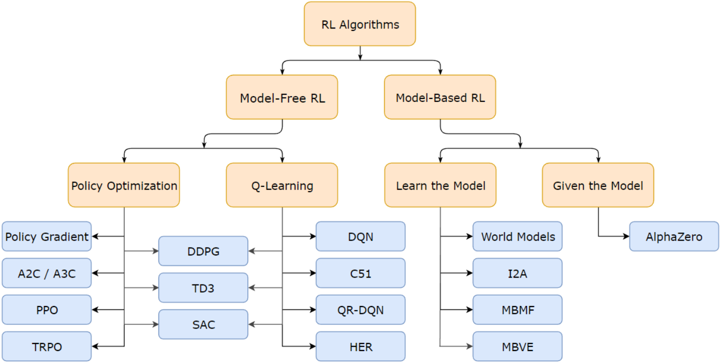
Probably the best known success story in RL is Google's deepmind team.
In 2013 they published a paper detailing their experiment where they used a DQN along with a few convolution layers to read input directly from the frames of a few different Atari games.

Success Stories in RL

In 2015, they published their improved results where their agent either achieved or surpassed human-level control on upwards of 20 different games.



RL Algorithms



References

- UCL. Advanced Topics 2015. (COMPM050/COMPGI13) *Reinforcement Learning*. David Silver. link
- Mnih et al. (2013). *Playing Atari with Deep Reinforcement Learning*.
- Mnih et al. (2015). *Human-level control through deep reinforcement learning*.
- AI Prism's Deep Reinforcement Learning Bootcamp