TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2019

Deep Reinforcement Learning

Definition of Reinforcement Learning

RL is defined by the following properties:

- An environment with **state**.
- State changes are influenced by **sequential decisions**.
- Reward (or loss) depends on **making decisions** that lead to **desirable states**.

Reinforcement Learning Examples

- Board games (chess or go)
- Atari Games (pong)
- Robot control (driving)
- Dialog
- Life

Policies

A policy is a way of behaving.

Formally, a (nondeterministic) policy maps a state to a probability distribution over actions.

$$\pi(a_t|s_t) \quad \text{probability of action } a_t \text{ in state } s_t$$

Imitation Learning

Construct a training set of state-action pairs (s, a) from experts.

"video" driver behavior

Define stochastic policy $\pi_\Phi(s)$.

$$\Phi^* = \operatorname*{argmin}_\Phi E_{(s,a)\sim\mathrm{Train}} \; -\ln \pi_\Phi(a \mid s)$$

This is just cross-entropy loss where we think of a as a "label" for s.

action as "label"

Dangers of Imperfect Imitation Learning

you are not "expert"

Perfect imitation learning would reproduce expert behavior. Imitation learning is **off-policy** — the state distribution in the training data is different from that defined by the policy being learned. learned a policy => collect training data

is not for your policy

Imitating experts, such as expert fire eaters, can be dangersous. "Don't try this at home".

Also, it is difficult to exceed expert performance by imitating experts. But this can happen.

Markov Decision Processes (MDPs)

An MDP consists of a set $\mathcal{S}$ of states, a set $\mathcal{A}$ of allowed actions, a reward function R and a next-state probability function P_T. We will use the following notation.

$s_t \in \mathcal{S}$ is the state at time t

$a_t \in \mathcal{A}$ is the action taken at time t.

$r_t = R(s_t, a_t) \in \mathbb{R}$ is the reward at time t

$P_T(s_{t+1}|s_t, a_t)$ is the probability of s_{t+1} given s_t and a_t.

The function $R(s, a)$ can allow for a cost of the action a.

Optimizing Reward

In RL we maximize reward rather than minimize loss.

$$\pi^* = \underset{\pi}{\text{argmax}} \; R(\pi)$$

$R(\pi) \; = \; E_\pi \; \sum_{t=0}^{T} \; r_t$ episodic reward (go)

or $E_\pi \; \sum_{t=0}^{\infty} \; \gamma^t r_t$ discounted reward (financial planning)

or $\lim_{T\to\infty} \frac{1}{T} \; \sum_{t=0}^{T} \; r_t$ asymptotic average reward (driving) imediate but also future Reward

The Value Function

For discounted reward:

Starting at S

$$V^{\pi}(s) = E_{\pi} \sum_{t} \gamma^{t} r_{t} \mid \pi,\ s_0 = s$$

optimal behaviour

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

immediate Reward

discounted of next state

$$\pi^{*}(a|s) = \operatorname*{argmax}_{a} R(s,a) + \gamma E_{s' \sim P_T(s'|s,a)} V^{*}(s')$$

$$V^{*}(s) = \max_{a} R(s,a) + \gamma E_{s' \sim P_T(\cdot|s,a)} V^{*}(s')$$

Value achieve under optimal Policy

Value Iteration

Suppose the state space and action space are finite.

In that case we can do value iteration.

Cannot get worse V_{i+1}
compare to V_i

$$V_0(s) = 0$$

$$V_{i+1}(s) = \max_a R(s,a) + \gamma E_{s' \sim P_T(\cdot|s,a)} V_i(s')$$

If all rewards are non-negative then

$$V_{i+1}(s) \geq V_i(s) \quad V_i(s) \leq V^*(s) \quad \text{so } \lim_{i \to \infty} V_i(s) \text{ exists}$$

Value Iteration

Theorem: **For discounted reward**

$$V_\infty(s) \doteq \lim_{i\to\infty} V_i(s) = V^*(s)$$

Proof

$$
\begin{aligned}
\Delta &\doteq \max_s \;\; V^*(s) - V_\infty(s) \quad = 0 \\
&= \max_s \begin{pmatrix} \max_a R(s,a) + E_{s'|a}\gamma V^*(s') \\ -\max_a R(s,a) + E_{s'|a}\gamma V_\infty(s') \end{pmatrix} \\
&\leq \max_s \max_a \begin{pmatrix} R(s,a) + E_{s'|a}\gamma V^*(s') \\ -R(s,a) + E_{s'|a}\gamma V_\infty(s') \end{pmatrix} \\
&= \max_s \max_a \; E_{s'|a}\, \gamma(V^*(s') - V_\infty(s')) \\
&\leq \gamma\Delta
\end{aligned}
$$

The Q Function

For discounted reward: Considering (s, a) rather than (s)
Value function

$$Q^{\pi}(s,a) = E_{\pi} \sum_t \gamma^t r_t \mid \pi,\ s_0 = s,\ a_0 = a$$

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

$$\pi^{*}(a|s) = \operatorname*{argmax}_a Q^{*}(s,a)$$

$$Q^{*}(s,a) = R(s,a) + \gamma E_{s' \sim P_T(\cdot|s,a)} \max_{a'} Q^{*}(s',a')$$

we know what action to take directly once we have values

Q Function Iteration

It is possible to define Q-iteration by analogy with value iteration, but this is generally not discussed.

Value iteration is typically done for finite state spaces. Let S be the number of states and A be the number of actions.

One update of a Q table takes $O(S^2 A^2)$ time while one update of value iteration is $O(S^2 A)$.

Q-Learning

When learning by updating the Q function we typically assume a parameterized Q function $Q_\Phi(s,a)$.

Q is correct, then error = 0

Bellman Error:

$$\mathrm{Bell}_\Phi(s,a) \doteq \left(Q_\Phi(s,a) - \left(R(s,a) + \gamma\ E_{s' \sim P_T(s'|s,a)}\ \max_{a'}\ Q_\Phi(s',a')\right)\right)^2$$

Theorem: If $\mathrm{Bell}_\Phi(s,a) = 0$ for all (s,a) then the induced policy is optimal.

Algorithm: Generate pairs (s,a) from the policy $\mathrm{argmax}_a\ Q_\Phi(s_t,a)$ and repeat

$$\Phi \mathrel{-}= \eta \nabla_\Phi\ \mathrm{Bell}_\Phi(s,a)$$

Issues with Q-Learning

Problem 1: Nearby states in the same run are highly correlated. This increases the variance of the cumulative gradient updates.

Problem 2: SGD on Bellman error tends to be unstable. Failure of Q_Φ to model unused actions leads to policy change (exploration). But this causes Q_Φ to stop modeling the previous actions which causes the policy to change back ...

To address these problems we can use a **replay buffer**.

Using a Replay Buffer

We use a replay buffer of tuples (s_t, a_t, r_t, s_{t+1}).

Repeat:

1. Run the policy $\mathrm{argmax}_a\, Q_\Phi(s,a)$ to add tuples to the replay buffer. Remove oldest tuples to maintain a maximum buffer size.
2. $\Psi = \Phi$
3. for N times select a random element of the replay buffer and do

$$\Phi \mathrel{-}= \eta \nabla_\Phi \left(Q_\Phi(s_t, a_t) - (r_t + \gamma \max_a Q_\Psi(s_{t+1}, a)\right)^2$$

holding this fixed

use only buffer's tuple

Replay is Off-Policy

Note that the replay buffer is from a **mixture of policies** and is **off-policy** for $\mathrm{argmax}_a\ Q_\Phi(s,a)$. This seems to be important for stability.

This seems related to the issue of stochastic vs. deterministic policies. More on this later.

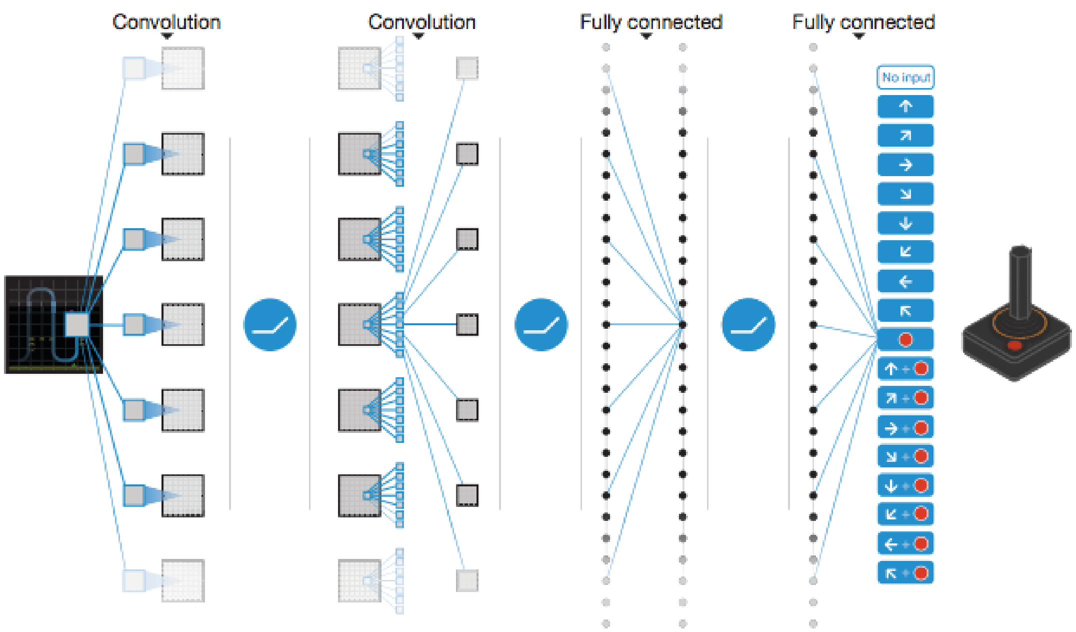
Multi-Step Q-learning

$$\Phi \mathrel{-}= \sum_t \nabla_\Phi \left(Q_\Phi(s_t, a_t) - \sum_{\delta=0}^{D} \gamma^\delta r_{(t+\delta)} \right)^2$$

Human-level control through deep RL (DQN)

Mnih et al., Nature, 2015. (Deep Mind)

We consider a CNN $Q_\Phi(s,a)$.



Watch The Video

https://www.youtube.com/watch?v=V1eYniJ0Rnk

Asynchronous Q-Learning (Simplified)

"agent"

No replay buffer. Many asynchronous threads each repeating:

$$\tilde{\Phi} = \Phi \text{ (retrieve } \Phi)$$

different agent get their parameters

using policy $\mathrm{argmax}_a\, Q_{\tilde{\Phi}}(s, a)$ compute

$$s_t, a_t, r_t, \ldots, s_{t+K}, a_{t+K}, r_{t+K}$$

runs a long game

$$\Phi \mathrel{-}= \eta \sum_{i=t}^{t+K-D} \nabla_{\tilde{\Phi}} \left(Q_{\tilde{\Phi}}(s_i, a_i) - \sum_{\delta=0}^{D} \gamma^{\delta} r_{i+\delta}\right)^2 \text{ (update } \Phi)$$

buffer → replaced by different agent play using $\bar{\Phi}$

The REINFORCE Algorithm

Williams, 1992

REINFORCE is a Policy Gradient Algorithm

We assume a parameterized policy $\pi_\Phi(a|s)$.

$\pi_\Phi(a|s)$ is normalized while $Q_\Phi(s,a)$ is not.

Policy Gradient Theorem (Episodic Case)

$$\Phi^* = \underset{\Phi}{\text{argmax}} \; E_{\pi_\Phi} \, R$$

episode

$$\nabla_\Phi \, E_{\pi_\Phi} \, R \; = \sum_{s_0,a_0,s_1,a_1,\ldots,s_T,a_T} \nabla_\Phi P(s_0, a_0, s_1, a_1, \ldots, s_T, a_T) \; R$$

prob of getting that episode

$$\begin{aligned} \nabla_\Phi \, P(\ldots)R \; &= \; P(S_0)\nabla_\Phi \, \pi(a_0)P(s_1)\pi(a_1) \cdots P(s_T)\pi(a_T) \; R \\ &\quad + P(S_0)\pi(a_0)P(s_1)\nabla_\Phi \, \pi(a_1) \cdots P(s_T)\pi(a_T) \; R \\ &\quad \vdots \\ &\quad + P(S_0)\pi(a_0)P(s_1)\pi(a_1) \cdots P(s_T)\nabla_\Phi \, \pi(a_T) \; R \end{aligned}$$

product Rule

$$= \; P(\ldots) \left(\sum_t \frac{\nabla_\Phi \; \pi_\Phi(a_t)}{\pi_\Phi(a_t)} \right) R$$

Policy Gradient Theorem (Episodic Case)

$$\nabla_\Phi \; P(\ldots)R \;=\; P(\ldots)\left(\sum_t \; \frac{\nabla_\Phi \; \pi_\Phi(a_t|s_t)}{\pi_\Phi(a_t|s_t)}\right) R$$

$$\nabla_\Phi \; E_{\pi_\Phi} \; R \;=\; E_{\pi_\Phi} \; \left(\sum_t \; \nabla_\Phi \; \ln \pi_\Phi(a_t|s_t)\right) R$$

Policy Gradient Theorem

$$\nabla_\Phi \, E_{\pi_\Phi} \, R$$

$$= E_{\pi_\Phi} \left(\sum_t \nabla_\Phi \, \ln \pi_\Phi(a_t|s_t) \right) R$$

$$= E_{\pi_\Phi} \left(\sum_t \nabla_\Phi \, \ln \pi_\Phi(a_t|s_t) \right) \left(\sum_t r_t \right)$$

$$= E_{\pi_\Phi} \sum_{t,t'} \nabla_\Phi \, \ln \pi_\Phi(a_t|s_t) \;\; r_{t'}$$

Remove case

t' before t

X want to blame policy that

don't happened yet

Proof

Policy Gradient Theorem

$$\nabla_\Phi \, E_{\pi_\Phi} \, R = \sum_{t,t'} E_{s_t,a_t,r_{t'}} \;\; \nabla_\Phi \;\; \ln \pi_\Phi(a_t|s_t) \;\; r_{t'}$$

For $t' < t$ we have

$$\begin{aligned}
E_{r_{t'},s_t,a_t} \;\; r_{t'} \nabla_\Phi \;\; \ln \pi_\Phi(a_t|s_t) &= E_{r_{t'},s_t} \, r_{t'} \sum_{a_t} \pi_\Phi(a_t|s_t) \; \nabla_\Phi \;\; \ln \pi_\Phi(a_t|s_t) \\
&= E_{r_{t'},s_t} \, r_{t'} \sum_{a_t} \nabla_\Phi \; \pi_\Phi(a_t|s_t) \\
&= E_{r_{t'},s_t} \, r_{t'} \; \nabla_\Phi \sum_{a_t} \pi_\Phi(a_t|s_t) \\
&= 0
\end{aligned}$$

$\frac{\nabla \ln \pi_\phi}{\pi_\phi(a|s)}$

REINFORCE

problem 1

$$\nabla_\Phi \; E_{\pi_\Phi} \; R \quad = \quad E_{\pi_\Phi} \sum_{t,\; t' \geq t} \left(\nabla_\Phi \; \ln \pi_\Phi(a_t|s_t)\right) \; r_{t'}$$

Sampling runs and computing the above sum over t is Williams' REINFORCE algorithm.

Optimizing Discrete Decisions with Non-Differentiable Loss

The REINFORCE algorithm is used generally for non-differentiable loss functions.

For example error rate and BLEU score are non-differentiable — they are defined on the result of discrete decisions.

$$\Phi^* = \underset{\Phi}{\text{argmax}}\; E_{w_1,\ldots,w_n \sim P_\Phi}\; \text{BLEU}$$

The Variance Issue

REINFORCE typically suffers from high variance of the gradient samples requiring very small learning rates and very long convergence times.

$$\nabla_\Phi \; E_{\pi_\Phi} \, R \; = \; E_{\pi_\Phi} \; \sum_{t,\; t' \geq t} \; (\nabla_\Phi \; \ln \pi_\Phi(a_t | s_t)) \;\; r_{t'}$$

We have to consider

- the variation over the choice of a_t
- the variation over $r_{t'}$ given a_t

The Variance Issue

$$\nabla_\Phi \; E_{\pi_\Phi} \; R \; = \; E_{\pi_\Phi} \; \sum_{t,\; t' \geq t} \; \nabla_\Phi \; \ln \pi_\Phi(a_t | s_t) \;\; r_{t'}$$

We can reduce the variation over $r_{t'}$ given s_t and a_t by shifting to the following.

$$\nabla_\Phi \; E_{\pi_\Phi} \; R \; = \; E_{\pi_\Phi} \; \sum_{t,t'} \; \nabla_\Phi \; \ln \pi_\Phi(a_t | s_t) \;\; E_{r_{t'} \,|\, s_t, a_t} \; r_{t'}$$

Before saying how this can be computationally approximated, we state the above expression somewhat differently.

$E_{s_{t'}, a_{t'} | s_t, a_t} r_{t'}$

Policy Gradient Theorem

$$\nabla_\Phi \, E_{\pi_\Phi} \, R = E_{\pi_\Phi} \sum_{t,\, t' \geq t} (\nabla_\Phi \, \ln \pi_\Phi(a_t|s_t)) \; E_{s_{t'}, a_{t'} \mid s_t, a_t} \, r_{t'}$$

$$= E_{\pi_\Phi} \sum_{t} (\nabla_\Phi \, \ln \pi_\Phi(a_t|s_t)) \;\; Q^{\pi_\Phi}(s_t, a_t)$$

$$Q^\pi(s,a) = E_\pi \; \textstyle\sum_t \, r_t \mid s_0 = s, \; a_0 = a$$

Policy Gradient Theorem

$$\nabla_\Phi \, E_{\pi_\Phi} \, R \; = \; E_{\pi_\Phi} \sum_t \; (\nabla_\Phi \; \ln \pi_\Phi(a_t|s_t)) \, Q^{\pi_\Phi}(s_t, a_t)$$

The point is that we can now approximate Q^{π_Φ} with neural network Q_Φ where the networks π_Φ and Q_Φ can use different, perhaps overlapping, parts of Φ.

We reduced the variance at the cost of approximating the expected future reward.

The Actor-Critic Algorithm

$$\nabla_\Phi \, E_{\pi_\Phi} \, R \approx E_{\pi_\Phi} \sum_t \left(\nabla_\Phi \; \ln \pi_\Phi(a_t|s_t)\right) \; Q_\Phi(s_t, a_t)$$

π_Φ is the "actor" and Q_Φ is the "critic"

The Actor-Critic Algorithm

$$\nabla_\Phi \, E_{\pi_\Phi} \, R \approx E_{\pi_\Phi} \sum_t \left(\nabla_\Phi \, \ln \pi_\Phi(a_t|s_t)\right) \, Q_\Phi(s_t, a_t)$$

max the rewards

We can sample an episode and then do

$$\Phi \mathrel{+}= \sum_t \eta_1 \left(\nabla_\Phi \, \ln \pi_\Phi(a_i|s_i)\right) Q_\Phi(s_t, a_t)$$

Reinforce algorithm

$$\Phi \mathrel{-}= \sum_t \eta_2 \, \nabla_\Phi \, \left(Q_\Phi(s_t, a_t) - \sum_{t' \geq t} r_{t'}\right)^2$$

Q learning

The two updates typically apply to different (but perhaps overlapping) subsets of the parameters Φ.

also want Q_Φ accurate

Variance from the Choice of a_t

To address the variance due to the choice of a_t we first make the following observation for any function $V(s)$ of states.

$\sum$ probability * Value

$$
\begin{aligned}
& E_{s_t, a_t} \; (\nabla_\Phi \; \ln \pi_\Phi(a_t|s_t))\, V(s_t) \\
= \; & E_{s_t} \sum_{a_t} (\pi_\Phi(a_t|s_t) \; \nabla_\Phi \; \ln \pi_\Phi(a_t|s_t))\, V(s_t) \\
= \; & E_{s_t} \sum_{a_t} (\nabla_\Phi \pi_\Phi(a_t|s_t))\, V(s_t) \\
= \; & E_{s_t} \, V(s_t) \sum_{a_t} (\nabla_\Phi \pi_\Phi(a_t|s_t))
\end{aligned}
$$

Variance from the Choice of a_t

$$E_{s_t,a_t} \;\; (\nabla_\Phi \; \ln \pi_\Phi(a_t|s_t))\, V(s_t)$$

$$= \; E_{s_t}\, V(s_t) \sum_{a_t} (\nabla_\Phi \pi_\Phi(a_t|s_t))$$

$$= \; E_{s_t}\, V(s_t)\; \nabla_\Phi \; \sum_{a_t} \pi_\Phi(a_t|s_t)$$

$$= \; 0$$

Variance from the Choice of a_t

$$E_{s_t,a_t} \sum_t \; (\nabla_\Phi \; \ln \pi_\Phi(a_t|s_t)) \, (Q^{\pi_\Phi}(s_t, a_t) - V(s_t))$$

$$= E_{s_t,a_t} \sum_t \; (\nabla_\Phi \; \ln \pi_\Phi(a_t|s_t)) \, Q^{\pi_\Phi}(s_t, a_t)$$

$$= \nabla_\Phi \; E_{\pi_\Phi} \; R$$

Variance from the Choice of a_t

replace Q by advantage

In particular we have

$$\nabla_\Phi \; E_{\pi_\Phi} \; R = E_{\pi_\Phi} \sum_t \; (\nabla_\Phi \; \ln \pi_\Phi(a_t|s_t)) \, (Q^{\pi_\Phi}(s_t, a_t) - V^{\pi_\Phi}(s_t))$$

$$V^{\pi_\Phi}(s) = E_{a \sim \pi_\Phi(a|s)} \; Q^{\pi_\Phi}(s, a)$$

$Q^{\pi_\Phi}(s, a) - V^{\pi_\Phi}(s)$ is the "advantage" of deterministically using a rather than sampling an action. → this reduce the variance

Nondeterminism of the policy π_Φ provides for exploration.

$Q(s,a) \to$ using a $\longrightarrow$ $V(s)$ = sampling an action

Advantage-Actor-Critic Algorithm

$$\nabla_\Phi \, E_{\pi_\Phi} \, R \approx \, E_{\pi_\Phi} \, \sum_t \, (\nabla_\Phi \, \ln \pi_\Phi(a_t|s_t)) \, \, (Q_\Phi(s_t, a_t) - V_\Phi(s_t))$$

We can sample an episode and then do

$$\Phi \mathrel{+}= \sum_t \eta_1 \, (\nabla_\Phi \, \ln \pi_\Phi(a_i|s_i)) \, (Q_\Phi(s_t, a_t) - V_\Phi(s_t))$$

$$\Phi \mathrel{-}= \sum_t \eta_2 \, \nabla_\Phi \, \left(Q_\Phi(s_t, a_t) - \sum\nolimits_{t' \geq t} r_{t'}\right)^2$$

$$\Phi \mathrel{-}= \sum_t \eta_3 \, \nabla_\Phi \, (V_\Phi(s_t) - Q_\Phi(s_t, a))^2$$

Asynchronous Methods for Deep RL (A3C)

Mnih et al., Arxiv, 2016 (Deep Mind)

$\tilde{\Phi} = \Phi$ (retrieve global Φ)

using policy $\pi_{\tilde{\Phi}}$ compute $s_t, a_t, r_t, \ldots, s_{t+K}, a_{t+K}, r_{t+K}$

$$R_i = \sum_{\delta=0}^{D} \gamma^{i+\delta} r_{(i+\delta)}$$

why not Q?

$$\Phi \mathrel{+}= \eta \sum_{i=t}^{t+K-D} \left(\nabla_{\tilde{\Phi}} \ln \pi_{\tilde{\Phi}}(a_i|s_i)\right) \left(R_i - V_{\tilde{\Phi}}(s_i)\right)$$

$$\Phi \mathrel{-}= \eta \sum_{i=t}^{t+K-D} \nabla_{\tilde{\Phi}} \left(V_{\tilde{\Phi}}(s_i) - R_i\right)^2$$

Issue: Policies must be Exploratory

The optimal policy is deterministic — $a(s) = \text{argmax}_a\, Q(s, a)$.

However, a deterministic policy never samples alternative actions.

Typically one forces a random action some small fraction of the time.

Issue: Discounted Reward

DQN and A3C use discounted reward on episodic or long term problems.

Presumably this is because actions have near term consequences.

This should be properly handled in the mathematics.

Observation: Continuous Actions are Differentiable

In problems like controlling an inverted pendulum, or robot control generally, a continuous loss can be defined and the gradient of loss of with respect to a deterministic policy exists.

More Videos

https://www.youtube.com/watch?v=g59nSURxYgk
https://www.youtube.com/watch?v=rAai4QzcYbs

END