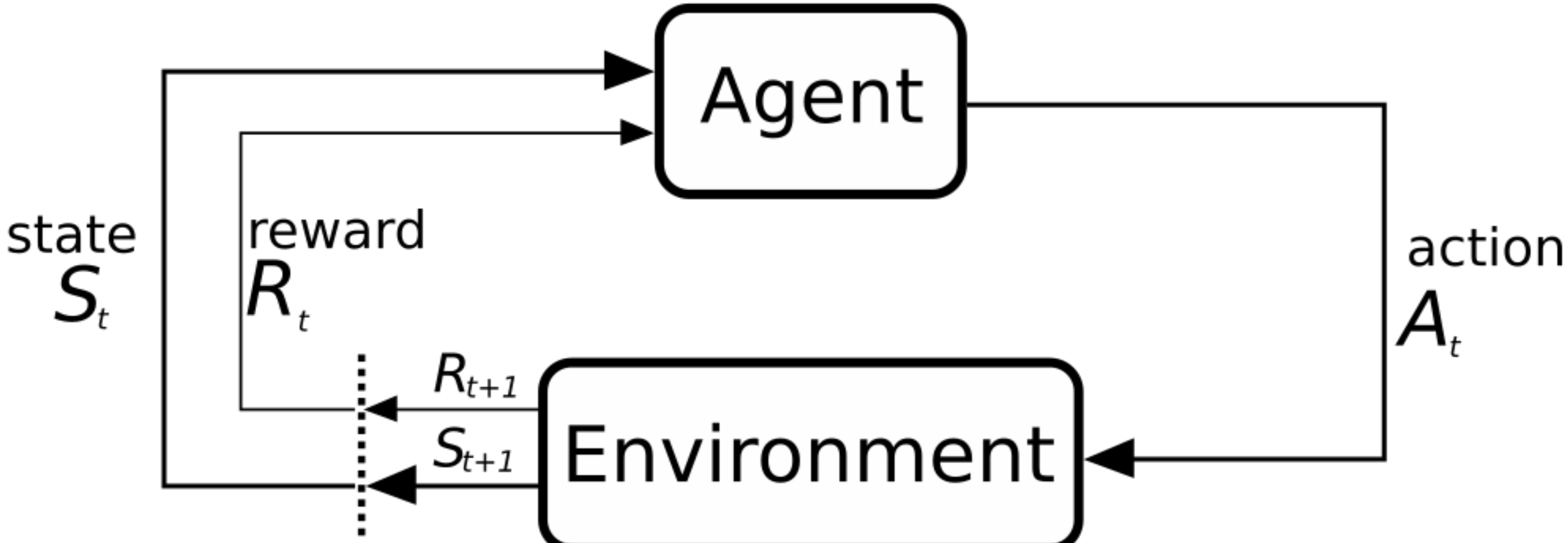
Exploration & Exploitation

Blink Sakulkueakulsuk

Recap

- A way to learn – a natural one
  - Learning through interactions
  - Active learning
  - Sequential interaction
  - Goal-direted
  - Learn even without the optimal sequence of actions

Recap



Rewards

It means, at time *t*, how good is the agent action

The ultimate goal of the agent is to maximize cumulative reward

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ldots$$

This cumulative reward is called **return**

Values

We define the expected cumulative reward of a state s, the **value.**

$$
\begin{aligned}
v(s) &= \mathbb{E}[G_t \mid St = s] \\
&= \mathbb{E}[R_{t+1} + R_{t+2} + R_{t+3} + \ldots \mid S_t = s]
\end{aligned}
$$

Action values

We can map value to both action and state

$$q(s, a) = \mathbb{E}[G_t \mid S_t = s, A_t = a]$$
$$= \mathbb{E}[R_{t+1} + R_{t+2} + R_{t+3} + \ldots | S_t = s, A_t = a]$$

Environment State

The internal state of the environment

Usually, it contains a lot of information. Those information may or may not be useful for the agent.

Sometimes, it is not even visible to the agent.

Agent State

A state of the agent can be reached by its **history**

$$H_t = O_0, A_0, R_1, O_1, \ldots, O_{t-1}, A_{t-1}, R_t, O_t$$

* O = Observation

Once the agent reaches this state, the actions can be determined

Markov decision processes

$$p(r, s \mid S_t, A_t) = p(r, s \mid H_t, A_t)$$

The equation above essentially means "the future state is independent of the past state given the present state"

MDP provides baseline for reinforcement learning.

Agent State

A state of the agent in this case is **a function of history**

$$S_{t+1} = u(S_t, A_t, R_{t+1}, O_{t+1})$$

The agent state is typically much smaller than the environment state

Policy

Given a state, a policy $\pi(S)$ defines an agent's action

- Mapping from agent state to action

Deterministic policy: $\pi(S) = A$

Stochastic policy: $\pi(A|S) = p(A|S)$

Value function

state s อ้าวอิงจากคู่มือ π

$$v_\pi(s) = \mathbb{E}[G_t \mid S_t = s, A_t \sim \pi(s)]$$
$$= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t \sim \pi(s)]$$

- $A_t \sim \pi(s)$ means that we choose action A from policy π
- $\gamma \in [0,1]$ is a discount factor

This is called a **Bellman equation**

If we want to find the optimal value, we can modify the equation to

$$v_*(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

Value function approximations

In many cases, we cannot find, or it is not feasible to find, the exact value from the value function.

- The states are infinitely many.

We need to approximate the function.

bc state it's too big

use neural network

Model

A **model** predicts what will happen next in the environment

For example, a model P predicts the next state

$$P(s,a,s') \approx p(S_{t+1} = s' \mid S_t = s, A_t = a)$$

For example, a model R predicts the next reward

$$R(s,a) = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$

Exploration & Exploitation

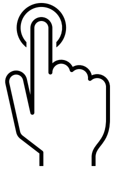
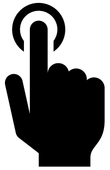
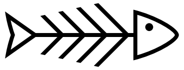
concept : ใช้ให้ agent เรียนรู้

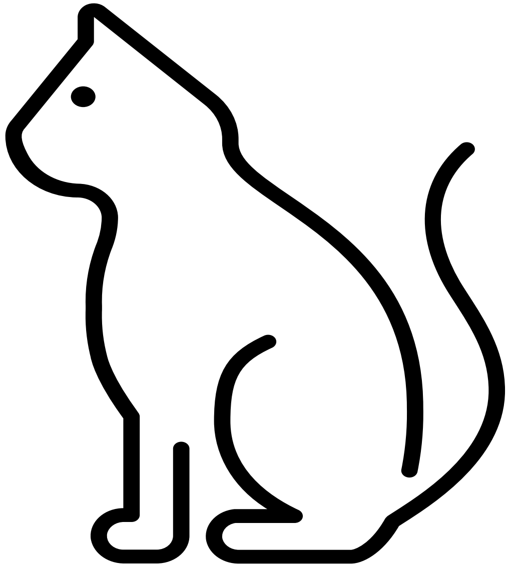
Key Question

How can we **learn** without compromising the **reward**?

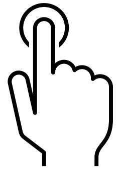
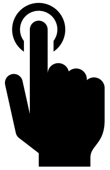
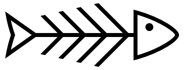
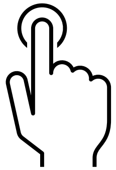
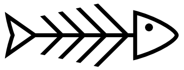
reward ↑ ↓ take ได้ทีละครั้ง

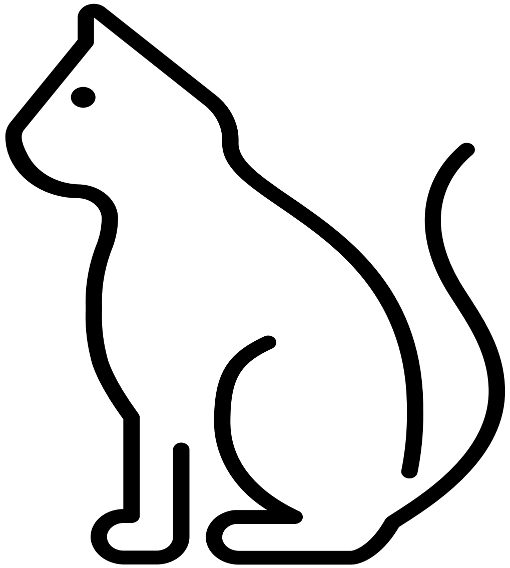
Example

Day	Action	Reward
1		
2		
3	?	



Example

Day	Action	Reward
1		
2		
3		 อ่าว ขาวทรยศ
4	?	กดเพื่อหา Pattern



Exploration vs Exploitation

Exploration: gain knowledge

exploitation

~~Exploration~~: gain maximum reward

๒

มั่นใจ รู้สูตร กดเพื่อรับ reward

But do we have enough knowledge to know the action to gain maximum reward?

- It is a trade-off. We need to learn, but not too much.

โจทย์

Introducing Multi-Armed Bandit

โจทย์พื้นฐาน so popular

+ Agent ไปตู้

แต่ละตู้จะมี reward disturbution ต่างกัน

AKA Slot Machine

เลือก 1 action

set of all action

At each step t the agent performs an action $A_t \in A$

เลือกมา 1 set ที่ timestep t ใน action ทั้งหมด

- The agent receives a reward R_t
- The agent doesn't know the reward rate for each action. (each machine has its own reward distribution $p(r|a)$)

ค.เป็นน่าจะเป็น ที่ได้รับ reward ใน 1 action

- The agent wants the maximum return over the lifetime.

Action values → เล่น 1000 ครั้ง ได้ avg reward เท่าไหร่

Since we have only 1 state, the action value is the following equation.

$$q(a) = \mathbb{E}[R_t | A_t = a]$$

t = 1000 | return

S1 = 600 | 1,200 baht → $q(A_1) = \frac{1200}{600} = 2$

S2 = 400 | 1,200 baht → $q(A_2) = \frac{1200}{400} = 3$

on average S1 = 2, S2 = 3

value มากกว่า

$$Q_t(a) = \frac{sum\ of\ reward}{numbers\ of\ action}$$

Expectation can be found as the average reward of the action.

$\frac{\text{sum of reward}}{\text{จน. action a}}$

identity

$$Q_t(a) = \frac{\sum_{n=1}^{t} R_n I(A_n = a)}{\sum_{n=1}^{t} I(A_n = a)}$$

if $A_n = a$, $I = 1$

$A_n \neq a$, $I = 0$

โดยที่ a คือ action ที่สนใจ

Given $I(True) = 1$ and $I(False) = 0$

Action values

* เก็บจริงไม่ได้ history เยอะเกิน memory ระเบิดตู้ม

Or it can be updated iteratively:

error ที่เก็บขึ้นควรจะ Adjust มากน้อย

หา different × αt

$$Q_t(A_t) = \begin{cases} Q_{t-1}(A_t) + \alpha_t \overbrace{\boxed{\left(R_t - Q_{t-1}(A_t)\right)}}^{\text{error}} & \text{for } a = A_t \\ Q_{t-1}(a) & \text{for } a \neq A_t \end{cases}$$

ไม่ทำไร เพราะไม่ใช่ A_t

เก็บแค่ $\bar{x}$ - state average, N - จน.ครั้ง

$$\frac{\bar{x}N + x}{N+1}$$ new state

Given

initial condition เริ่มต้นที่ 0

$$\alpha_t = \frac{1}{N_t(A_t)}, \qquad N_t(A_t) = N_{t-1}(A_t) + 1, \qquad N_t(a) = 0, \forall a$$

α_t คือ จน.ครั้งที่โยกตู้ที่ n เช่น โยก 10 ครั้ง avg reward = 2 แต่โยกได้ 3, extra = 1 → $\frac{1}{10}$

แรก $N_t(A_t)$ ค่าน้อย error จะทำให้ค่า swing และพอ $N_t(A_t)$ เยอะขึ้น ค่าจะลู่เข้า

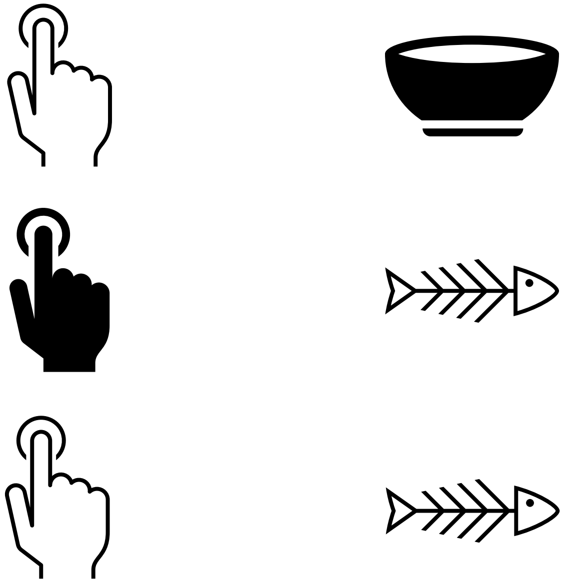
Example

Day Action Reward

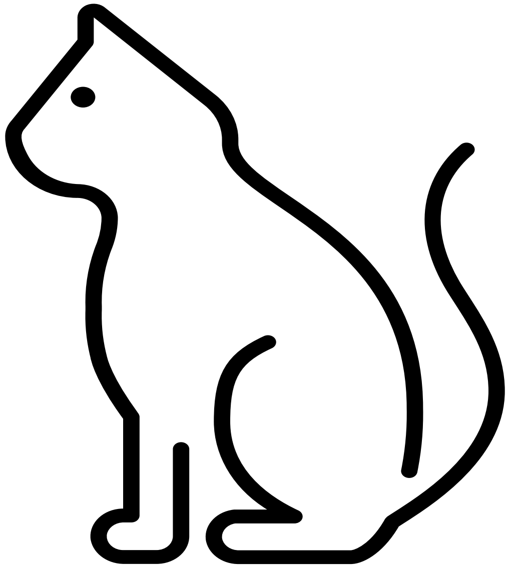
1

2

3



4 ?



reward

Food: R = 1

No food: R = -1

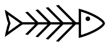
กดครั้งที่ 3

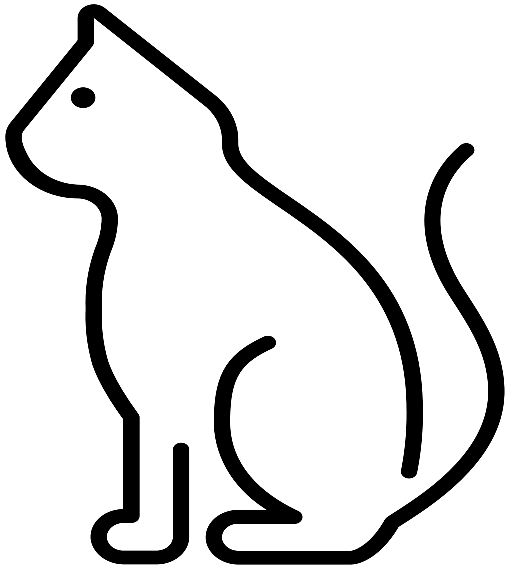
$Q_3(white) = 0$ → Reward 1 และ -1 ; $\frac{1+(-1)}{2}$

$Q_3(black) = -1$ → Reward -1 และ -1 ; $\frac{-1+(-1)}{2}$

$\frac{\bar{x}N + x}{N+1}$

Example

Day	Action	Reward
1		
2		
3		
4		
5		
6		



Food: R = 1

No food: R = -1

$Q_6(white) = -0.6 \rightarrow \frac{-0.6 + (-1)}{2} = -0.8$
$Q_6(black) = -1$

What choice to choose next?

exploration → พอเถอะ ไปกดดำ

exploitation → ขาวเพราะเคยได้ ดูจากผลลัพธ์ที่เห็นไปแล้ว

Introducing Regrets → for balancing exploration & exploitation

At the point, everyone knows that we need to balance between exploration and exploitation.

- How can we balance it?
- Can we set some assumptions and follow them?
- Can we even do it optimally?

Regrets

The optimal value

$$v_* = \max_{a \in \boldsymbol{A}} q(a) = \max_a \mathbb{E}[R_t | A_t = a]$$

Regret is the loss occurred when taking sub-optimal action

$$v_* - q(A_t)$$

Noted that the agent doesn't know the regret, because it doesn't know the optimal value

Regrets

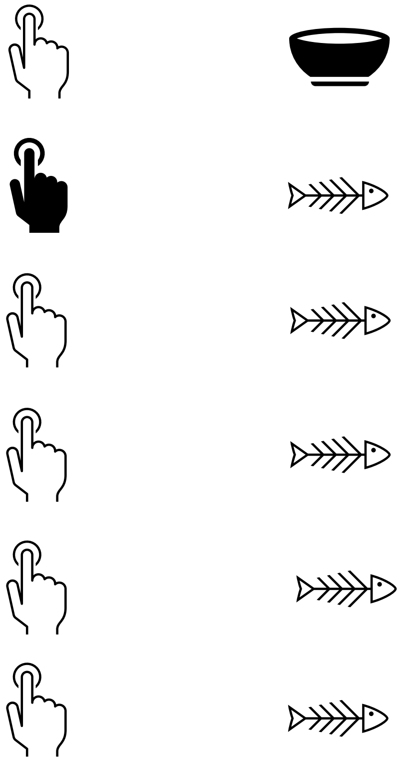
We can sum up all the regrets in the agent's lifetime

$$L_t = \sum_{i=1}^{t} (v_* - q(a_i))$$

By minimizing the cumulative regret, we maximize the cumulative reward

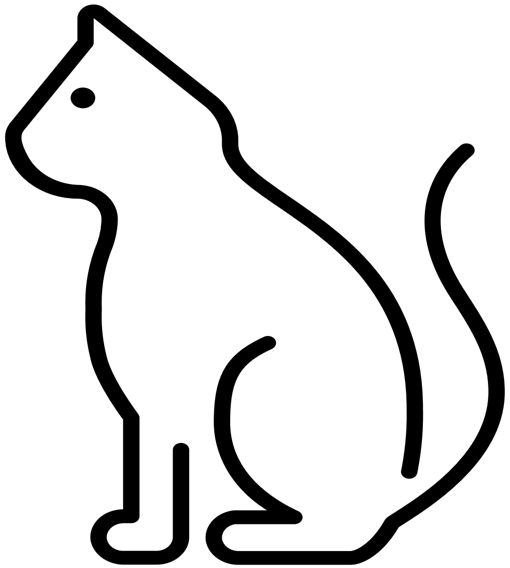
Regret Example

Day	Action	Reward
1		
2		
3		
4		
5		
6		



behavior : ทำซ้ำเพราะเคยได้

greedy



กดขาวได้ >>>> regret จะเพิ่มขึ้น

เพราะไม่เลือกจะเรียนรู้ค่า

Food: R = 1 / No food: R = -1

$p(food|white) = 0.1$

$p(food|black) = 0.9$

food   value ของการได้อาหาร

$v_* = q(black) = 0.9 * 1 + 0.1 * (-1) = 0.8$

$q(white) = 0.9 * (-1) + 0.1 * 1 = -0.8$

No food   ค.น่าจะเป็น

If the cat continues to choose white (because of the earlier episodes), it will get regrets of $1.6t$

Regrets

action a
กดกี่ครั้ง
regret เท่าไหร่

+

action b
กดกี่ครั้ง
regret เท่าไหร่

+

action c
กดกี่ครั้ง
regret เท่าไหร่

$$L_t = \sum_{i=1}^{t} \left(v_* - q(a_i)\right) = \sum_{a \in A} N_t(a)\left(v_* - q(a)\right) = \sum_{a \in A} N_t(a)\Delta_a$$

exploitation

exploration

Now, minimizing the equation above shows the balance between exploration and exploitation.

If we want to take an action $\boldsymbol{a}$ many times, we need to reduce the action regret Δ_a.

Option 1: epsilon-greedy

exploration ผ่านใต้ค่า epsilon

ไม่รู้ว่าตัวค่านี้เท่าไหร่ แต่มีวิธีแก้

A simple, yet powerful, algorithm

- $\epsilon \in [0,1]$ is a variable for exploration
- We select random action with probability ϵ
  - $a = random(A)$

random มาเทียบค่า epsilon

เป็น threshold of environment

if epsilon > ทอยเต๋า ; random เก็บใน state action value

why

epsilon < ทอยเต๋า ; เช็ค state action value and select best action

- We select the best action with probability $1 - \epsilon$
  - $a = argmax_{a \in A} Q_t(a)$

เลือก epsilon ยังไง ?

- ไม่มีค่าตอบตายตัว
-

Option 2: Upper Confidence Bounds

idea มาจาก theory ในฝั่ง Propobility

Add an upper confidence term $U_t(a)$ to the action value and adjust it through learning

Then we select an action that minimizing the adjusted action value

max value ที่เกิดขึ้น

ร action ไหนทำให้เกิด

$$a_t = \underbrace{argmax_{a \in A} Q_t(a) + U_t(a)}_{\text{action ไหนที่ทำให้ ค่า max}}$$

และ return อะไร

ตัวที่ออกมา ← a_t

Option 2: Upper Confidence Bounds

$$a_t = argmax_{a \in A} Q_t(a) + U_t(a)$$

เยอะ เพื่อจะได้ดัง argmax มา bc ไม่มั่นใจใน a

When we use action a a few times, $U_t(a)$ is large. (not sure if the value is correct) → ถ้าจุดนั้ง a_t จะ dominate U_t

When we use action a many times, $U_t(a)$ is small. (pretty sure that the value is correct)

* What is $U_t(a)$

Option 2: Upper Confidence Bounds

Assumption 1: we want to minimize this:

$$L_t = \sum_{i=1}^{t}\left(v_* - q(a_i)\right) = \sum_{a\in A} N_t(a)\left(v_* - q(a)\right) = \sum_{a\in A} N_t(a)\Delta_a$$

Assumption 2: we want to implement this:

When we use action a a few times, $U_t(a)$ is large. (not sure if the value is correct)

When we use action a many times, $U_t(a)$ is small. (pretty sure that the value is correct)

Option 2: Upper Confidence Bounds

represent regret ไปแล้ว no care

$$U_t(a) = \sqrt{\frac{\log t}{2N_t(a)}}$$

จน.ครั้งที่ทำ action a

Proved by using Hoeffding's Inequality

If we use the term above, we can implement an algorithm to both explore and exploit at the same time.

adjust แบบ dynamic to threshold

like epsilon

ดูว่า state ไหน return max

$$a_t = argmax_{a \in A} Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}$$

เพิ่มแบบ linear

เพิ่ม

ยิ่ง $N_t(a)$ มากเท่าไหร่ พจน์นี้จะน้อยลงเรื่อยๆ

เป็นตัว compensate Q_t ว่าควร explore, exploit

Option 2: Upper Confidence Bounds

any constant

$$a_t = argmax_{a \in A} Q_t(a) + c\sqrt{\frac{\log t}{\underbrace{N_t(a)}}}$$

train 100%

มากสุดใจ พจน์

นี้ต่ำมาก จะสนแต่ค่า Q_t

หมายความว่า

ค่าลู่เข้าสู่ true value

Basically, when we haven't explored some actions much, the value is high, and that action is selected and updated. Eventually, the value will converge to the true value for each action. Thus we can find an optimal action.

โจทย์ น้อง ตู้

```
n0 = 0
n1 = 0
sum = 0
sum_n0 = 0
sum_n1 = 0

for i in range (100000):
    x = np.random.uniform()
    if x < 0.9:   epsilon
        sum_n0 += -1   } state action value
        n0 += 1
    else:
        sum_n1 += 1
        n1 += 1


print(sum_n0)
print(sum_n1)
print(n0)
print(n1)
```