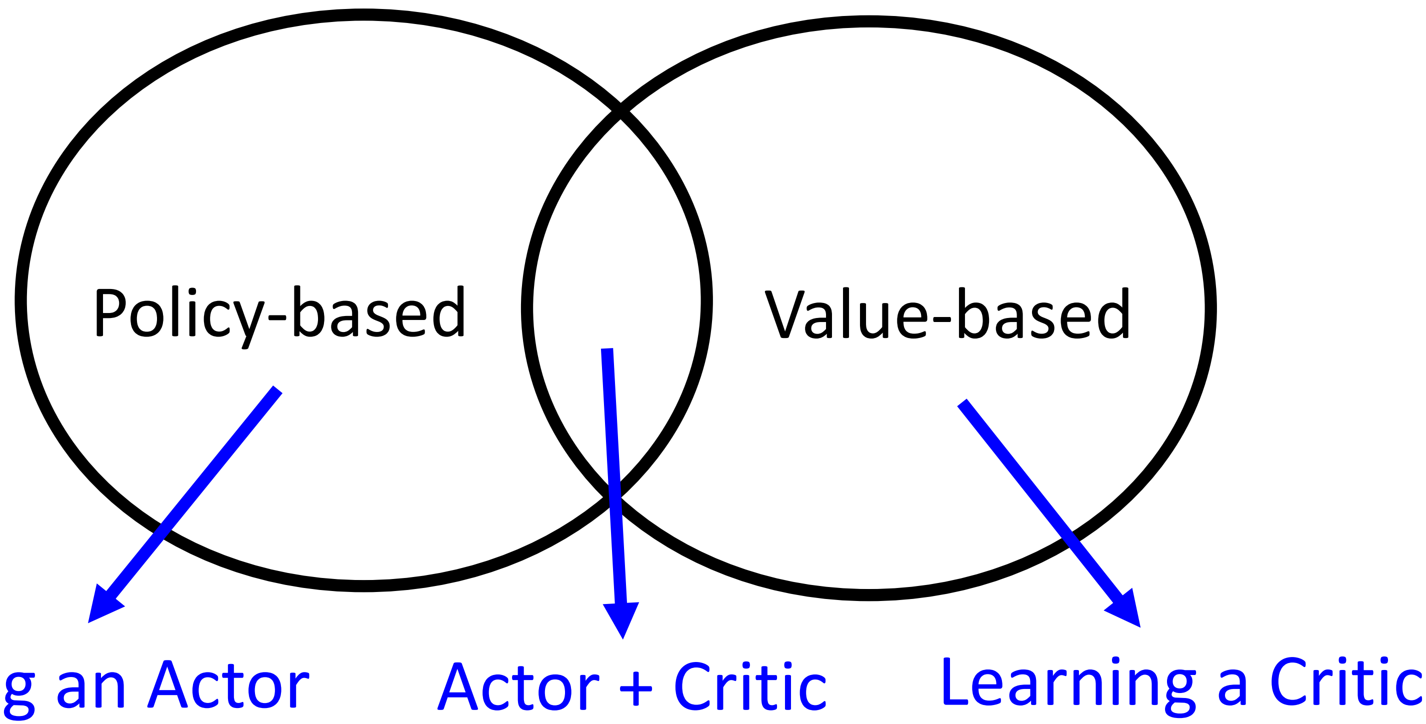
Outline

Alpha Go: policy-based + value-based + model-based



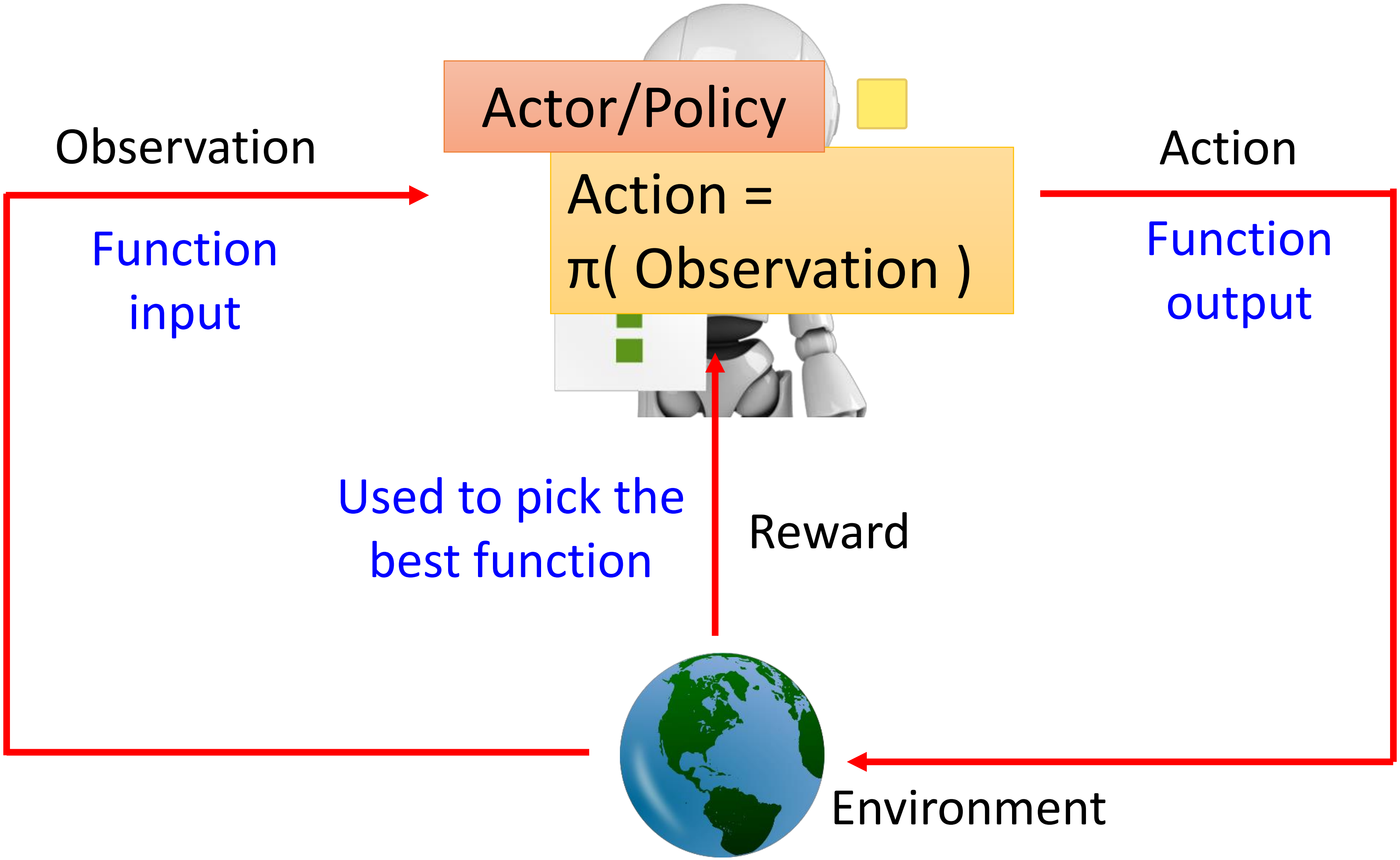
Asynchronous Advantage Actor-Critic (A3C)

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning", ICML, 2016

Policy-based Approach

Learning an Actor

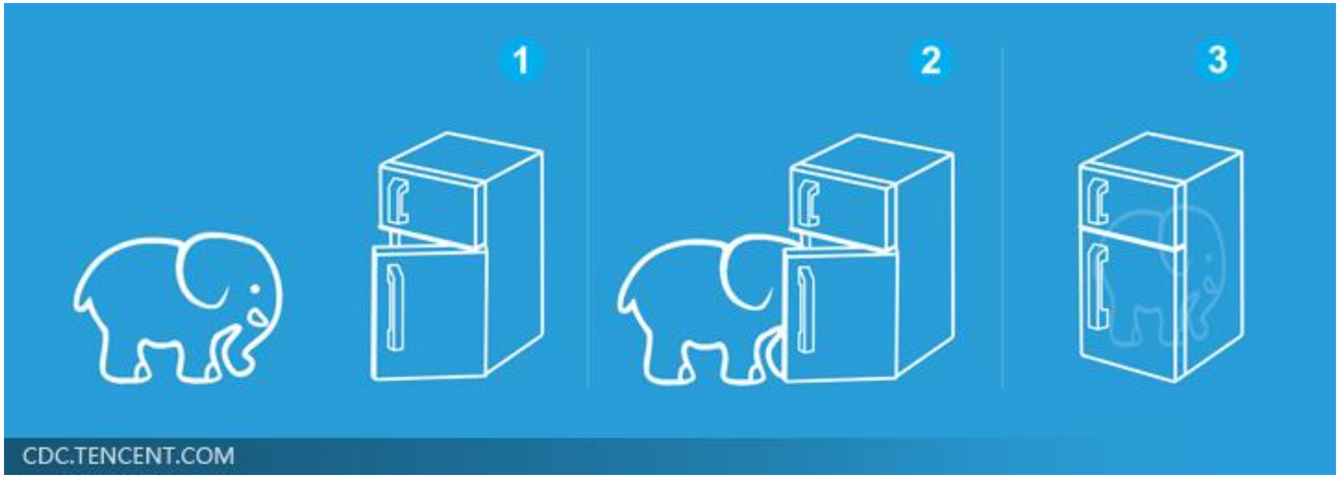
Machine Learning
≈ Looking for a Function



Three Steps for Deep Learning

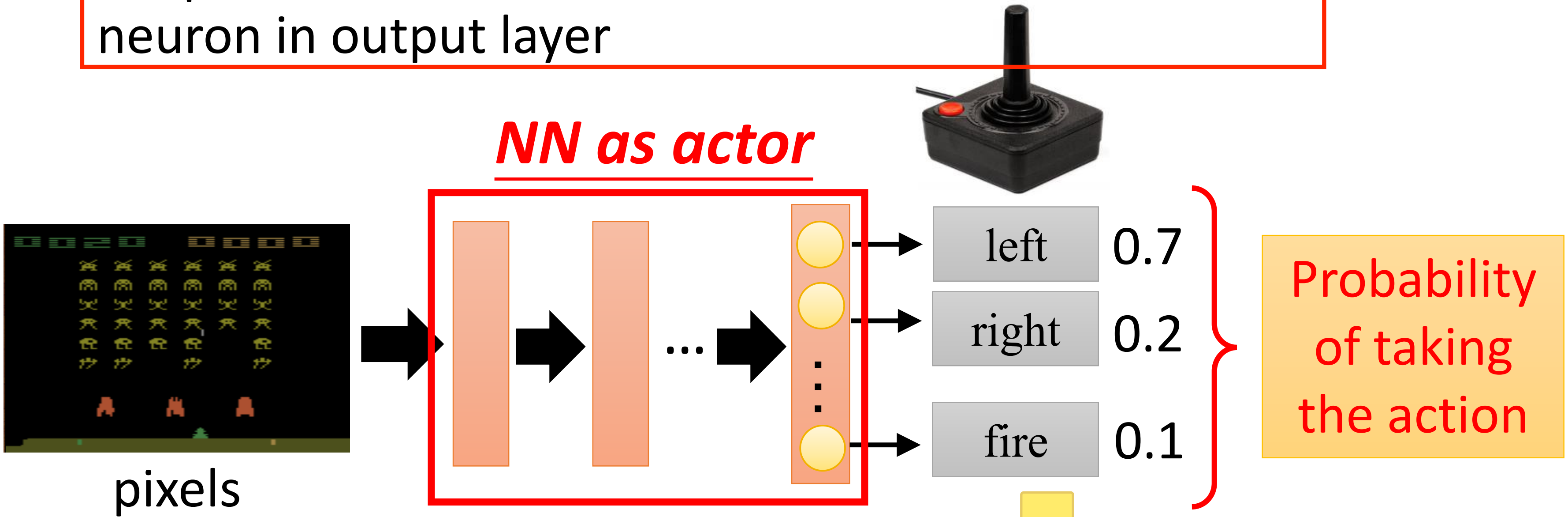
Step 1: Neural Network as Actor → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



Neural network as Actor

- Input of neural network: the observation of machine represented as a vector or a matrix
- Output neural network : each action corresponds to a neuron in output layer



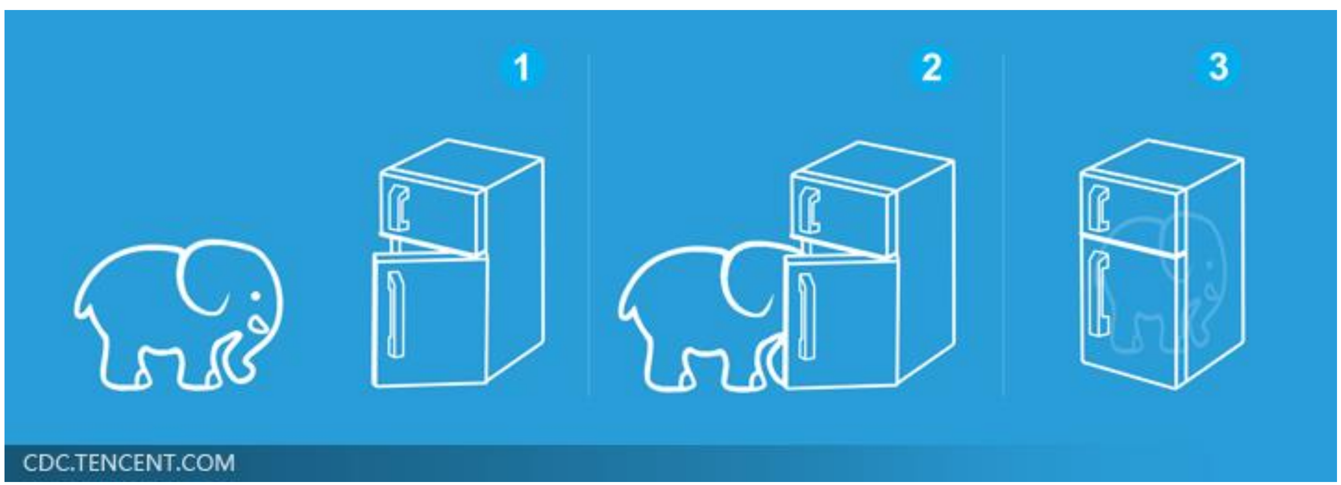
What is the benefit of using network instead of lookup table?

generalization

Three Steps for Deep Learning

Step 1: Neural Network as Actor → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



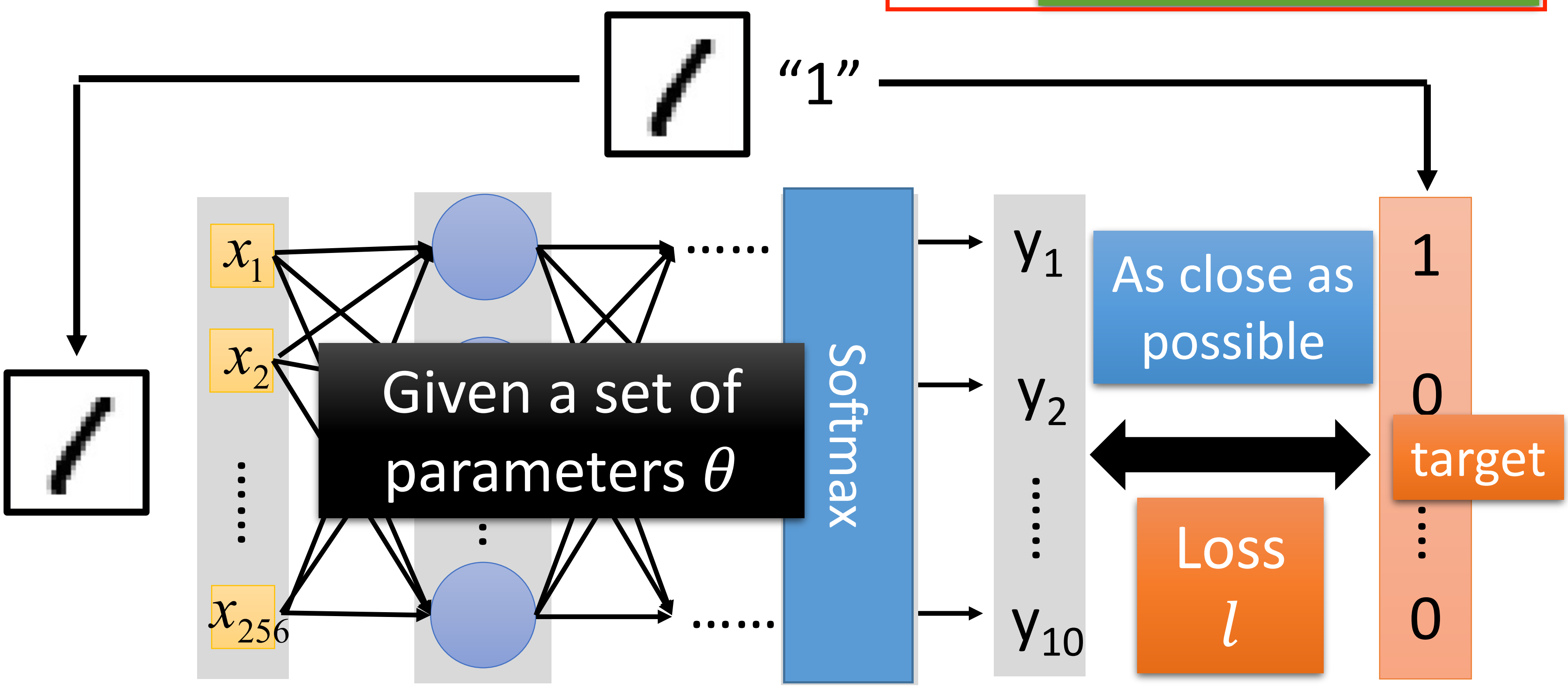
Goodness of Actor

- Review: Supervised learning

Total Loss: $$L = \sum_{n=1}^{N} l_n$$

Find ***the network parameters θ^**** that minimize total loss L

Training Example



Goodness of Actor

- Given an actor $\pi_\theta(s)$ with network parameter θ
- Use the actor $\pi_\theta(s)$ to play the video game
  - Start with observation s_1
  - Machine decides to take a_1
  - Machine obtains reward r_1
  - Machine sees observation s_2
  - Machine decides to take a_2
  - Machine obtains reward r_2
  - Machine sees observation s_3
  - ......
  - Machine decides to take a_T
  - Machine obtains reward r_T

END

Total reward: $R_\theta = \sum_{t=1}^{T} r_t$

Even with the same actor, R_θ is different each time

Randomness in the actor and the game

We define $\bar{R}_\theta$ as the *expected value* of R_θ

$\bar{R}_\theta$ evaluates the goodness of an actor $\pi_\theta(s)$

Goodness of Actor

- An episode is considered as a trajectory τ
  - $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T\}$
  - $R(\tau) = \sum_{t=1}^{T} r_t$
  - If you use an actor to play the game, each τ has a probability to be sampled
    - The probability depends on actor parameter θ: $P(\tau|\theta)$

$$\bar{R}_\theta = \boxed{\sum_{\tau}} R(\tau) \boxed{P(\tau|\theta)} \approx \boxed{\frac{1}{N}\sum_{n=1}^{N}} R(\tau^n)$$

Use π_θ to play the game N times, obtain $\{\tau^1, \tau^2, \cdots, \tau^N\}$

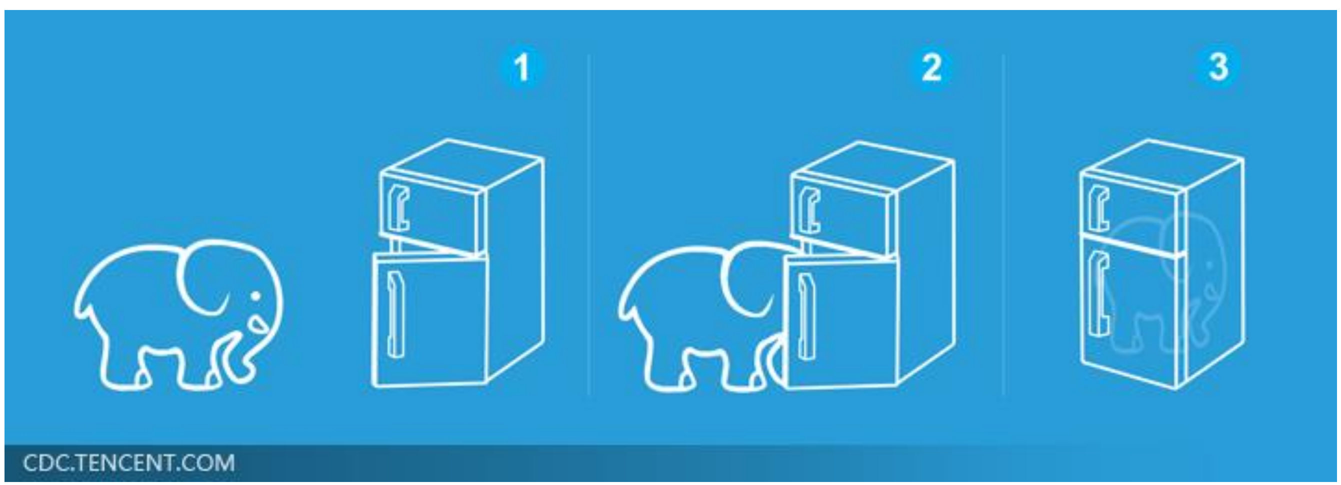
Sum over all possible trajectory

Sampling τ from $P(\tau|\theta)$ N times

Three Steps for Deep Learning

Step 1: Neural Network as Actor → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



Gradient Ascent

- Problem statement

$$\boxed{\theta^* = arg \max_\theta \bar{R}_\theta \qquad \bar{R}_\theta = \sum_\tau R(\tau)P(\tau|\theta)}$$

- Gradient ascent
  - Start with θ^0
  - $\theta^1 \leftarrow \theta^0 + \eta \nabla \bar{R}_{\theta^0}$
  - $\theta^2 \leftarrow \theta^1 + \eta \nabla \bar{R}_{\theta^1}$
  - ……

$$\theta = \{w_1, w_2, \cdots, b_1, \cdots\}$$

$$\nabla \bar{R}_\theta = \begin{bmatrix} \partial \bar{R}_\theta / \partial w_1 \\ \partial \bar{R}_\theta / \partial w_2 \\ \vdots \\ \partial \bar{R}_\theta / \partial b_1 \\ \vdots \end{bmatrix}$$

Gradient Ascent

$$\bar{R}_\theta = \sum_\tau R(\tau) P(\tau|\theta) \qquad \nabla \bar{R}_\theta = ?$$

$$\nabla \bar{R}_\theta = \sum_\tau R(\tau) \nabla P(\tau|\theta) = \sum_\tau R(\tau) P(\tau|\theta) \frac{\nabla P(\tau|\theta)}{P(\tau|\theta)}$$

$R(\tau)$ do not have to be differentiable

It can even be a black box.

$$= \boxed{\sum_\tau} R(\tau) \boxed{P(\tau|\theta)} \nabla log P(\tau|\theta)$$

$$\boxed{\frac{dlog\left(f(x)\right)}{dx} = \frac{1}{f(x)} \frac{df(x)}{dx}}$$

$$\approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n) \underline{\nabla log P(\tau^n|\theta)}$$

Use π_θ to play the game N times,
Obtain $\{\tau^1, \tau^2, \cdots, \tau^N\}$

Gradient Ascent

$\nabla log P(\tau|\theta) = ?$

- $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T\}$

$$P(\tau|\theta) =$$

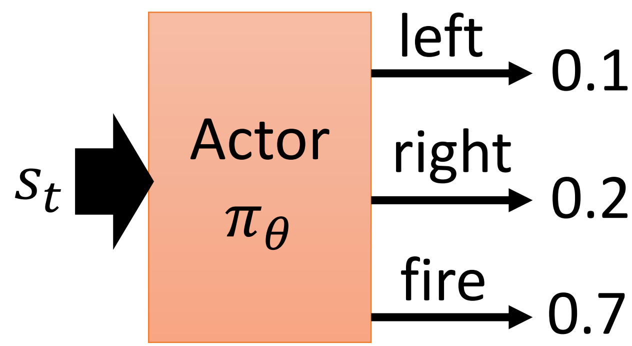
$$p(s_1)p(a_1|s_1,\theta)p(r_1,s_2|s_1,a_1)p(a_2|s_2,\theta)p(r_2,s_3|s_2,a_2)\cdots$$

$$= p(s_1)\prod_{t=1}^{T} p(a_t|s_t,\theta)p(r_t,s_{t+1}|s_t,a_t)$$

not related to your actor

Control by your actor π_θ

$p(a_t = "fire"|s_t,\theta)$
$= 0.7$



Gradient Ascent

$\nabla logP(\tau|\theta) = ?$

- $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T\}$

$$P(\tau|\theta) = p(s_1)\prod_{t=1}^{T} p(a_t|s_t,\theta)p(r_t,s_{t+1}|s_t,a_t)$$

$$logP(\tau|\theta)$$

$$= logp(s_1) + \sum_{t=1}^{T} logp(a_t|s_t,\theta) + logp(r_t,s_{t+1}|s_t,a_t)$$

$$\nabla logP(\tau|\theta) = \sum_{t=1}^{T} \nabla logp(a_t|s_t,\theta)$$

Ignore the terms not related to θ

Gradient Ascent

$$\nabla log P(\tau|\theta)$$
$$= \sum_{t=1}^{T} \nabla log p(a_t|s_t, \theta)$$

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

$$\boxed{\nabla \bar{R}_\theta} \approx \frac{1}{N}\sum_{n=1}^{N} R(\tau^n) \nabla log P(\tau^n|\theta) = \boxed{\frac{1}{N}\sum_{n=1}^{N} \overset{1.}{R(\tau^n)} \sum_{t=1}^{T_n} \nabla log \overset{2.}{p(a_t^n|s_t^n, \theta)}}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n) \nabla log p(a_t^n|s_t^n, \theta)$$

What if we replace $R(\tau^n)$ with r_t^n ......

If in τ^n machine takes a_t^n when seeing s_t^n in

$R(\tau^n)$ is positive → Tuning θ to increase $p(a_t^n|s_t^n)$

$R(\tau^n)$ is negative → Tuning θ to decrease $p(a_t^n|s_t^n)$

It is very important to consider the cumulative reward $R(\tau^n)$ of the whole trajectory τ^n instead of immediate reward r_t^n

Gradient Ascent

$$\nabla \log P(\tau|\theta) = \sum_{t=1}^{T} \nabla \log p(a_t|s_t,\theta)$$

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N} R(\tau^n) \nabla \log P(\tau^n|\theta) = \boxed{\frac{1}{N}\sum_{n=1}^{N} R(\overset{1.}{\tau^n}) \sum_{t=1}^{T_n} \nabla \log p(\overset{2.}{a_t^n}|s_t^n,\theta)}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n) \boxed{\boxed{\nabla \underline{\log} p(a_t^n|s_t^n,\theta)} \quad \frac{\nabla p(a_t^n|s_t^n,\theta)}{p(a_t^n|s_t^n,\theta)}}$$

Why divided by $p(a_t^n|s_t^n,\theta)$?

e.g. in the sampling data … s has been seen in $\tau^{13}, \tau^{15}, \tau^{17}, \tau^{33}$

In τ^{13}, take action a	$R(\tau^{13}) = 2$	In τ^{15}, take action b	$R(\tau^{15}) = 1$
In τ^{17}, take action b	$R(\tau^{17}) = 1$	In τ^{33}, take action b	$R(\tau^{33}) = 1$

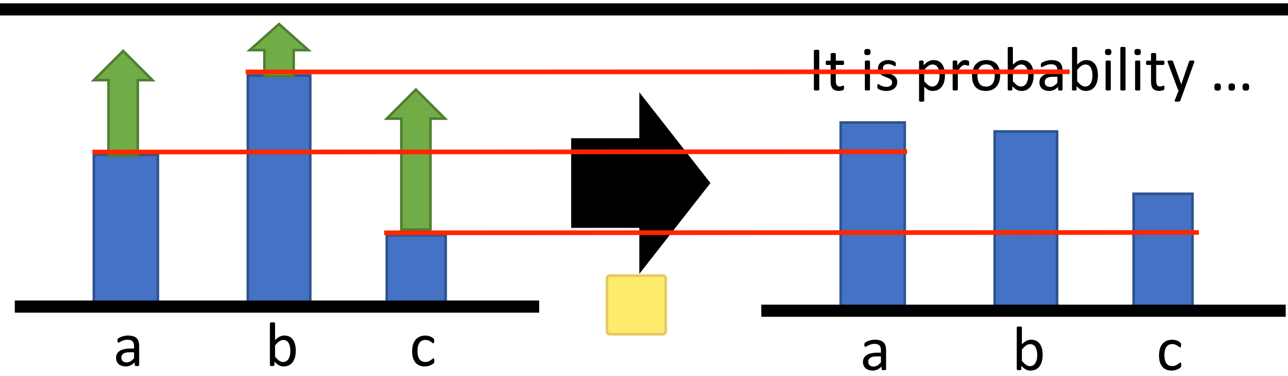
Add a Baseline

It is possible that $R(\tau^n)$ is always positive.

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

$$\boxed{\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (R(\tau^n) \underline{-\ b}) \nabla log p(a_t^n | s_t^n, \theta)}$$

Ideal case



Sampling

......