L13

1) Recap : Unsupervised learning
2) Reinforcement Learning (RL)
    - Model free RL
    - Policy gradients

Supervised learning	Unsupervised learning
— Regression	— data visualization
— Classification	— Dimensionality reduction

— Data is static.

— learning is offline.

— prediction do not affect data samples

Reinforcement Learning (RL)

— Dataset is dynamically changing

— learning is online

— Explicit modeling of feedback loops

Application: → stock market prediction

→ Gaming [AlphaGo, AlphaZero, OpenAI. Dota

↓ Robotics

[RL ⟷ Control theory

Dynamic System, Optimal nonlinear control]

Setting

- Environment
- Agent
- Actions at time step
- Actions → rewards
- Goal: Maximum rewards.

$S_{t+1} = f(S_t, a_t)$

↳ state transition function / dynamical system

Reward: $r(S_t, a_t)$

Goal:

Agent observes

$\tau = S_0, a_0, S_1, a_1, \cdots, S_{T-1}$

↳ Horizon

↳ Trajectory / rollout 轨迹

$r(S_t, a_t)$ for $t = 0 \cdots (T-1)$

Predict a strategy / policy for for deciding the next step action

Policy Function: $a_t = \pi(\tau)$ — τ ← previous timestamps

Apply ML to solve this prediction problems!

→ Choose representation for π (policy)

→ Choose a loss function

$$R(\tau) = \sum_{t=0}^{T-1} -r(s_t, a_t)$$

Subject to $s_{t+1} = f(s_t, a_t)$

$a_t = \pi(\tau_t)$, so is given π — try to learn policy

→ linear models, logistic, neural nets

Deep RL

→ optimize this function

Need to figure out $\frac{\partial R}{\partial \pi}$

— Idea: Assume that π is not deterministic, instead it predicts a prob distribution of actions

↳ a_t random

s_t random — i.e. R is a random variable

Modified problem $\min E_{\pi(\tau)} R(\tau) = E\left[-\sum_{t=0}^{T-1} r(a_t, s_t)\right]$

s.t. $s_{t+1} = f(s_t, a_t)$

$a_t = \pi(\tau_t)$

Training Samples: → multiple trajectories / rollouts

Supervised $x \to y$ — only 1 y per rollout (π) — RL — Unsupervised x

— Q-learning
— Value iterations
— Actor Critic

Policy gradients:

assume π is a linear policy parametered by θ

$$\frac{\partial E_{z(\tau)}[R(\tau)]}{\partial \theta} = ?$$

log-derivative

$$\frac{\partial}{\partial \theta} \log z(\tau) = \frac{1}{z(\tau)} \cdot \frac{\partial}{\partial \theta} z(\tau)$$

i.e. $\frac{\partial z(\tau)}{\partial \theta} = z(\tau) \cdot \frac{\partial}{\partial \theta} \log z(\tau)$

$$\begin{aligned}
\frac{\partial E_{z(\tau)}[R(\tau)]}{\partial \theta} &= \frac{\partial}{\partial \theta} \sum_{\tau} z(\tau) R(\tau) \\
&= \sum_{\tau} R(\tau) \frac{\partial z(\tau)}{\partial \theta} \\
&= \sum_{\tau} \underbrace{z(\tau)}\ \underbrace{R(\tau) \frac{\partial}{\partial \theta} \log z(\tau)}
\end{aligned}$$

$$\nabla E_{z(\tau)}[R(\tau)] = E_{z(\tau)}\left[R(\tau) \frac{\partial}{\partial \theta} \log z(\tau)\right]$$

Gradient of expected rewards

$\doteq$ Expected value of Reward $\times$ log derivative

$\therefore$ Just like in SGD,

- sample different roll outs
- compute approximation to the gradient of expected rewards.
- update θ in that direction

Repeat

REINFORCE

(把前面的 a_t, s_t)

1) sample rollout $\tau = (s_0, a_0, s_1, \ldots, s_T)$
2) compute $R(\tau) = \sum_{t=0}^{T-1} -r(s_t, a_t)$
3) $\theta \leftarrow \theta - \eta R(\tau) \frac{\partial \log \pi_\theta(\tau)}{\partial \theta}$

reward 分数，越大越好，$R(\tau) \downarrow, r \uparrow$

↳ R appears but never its gradient

↳ Useful when reward are discrete

↳ Environment f do not appear, only the rollout $\tau \ldots R(\tau)$

"Model free reinforcement learning"

↳ "model_based_RL"

Derivative free optimizator

Gradient descent $\theta \leftarrow \theta - \eta \nabla R(\theta)$

Random search
- (a) pick a random vector ✓
- (b) compute step size (either negative or positive) that minimize $R(\theta + \eta v)$
- (c) $\theta \leftarrow \theta + \eta v$