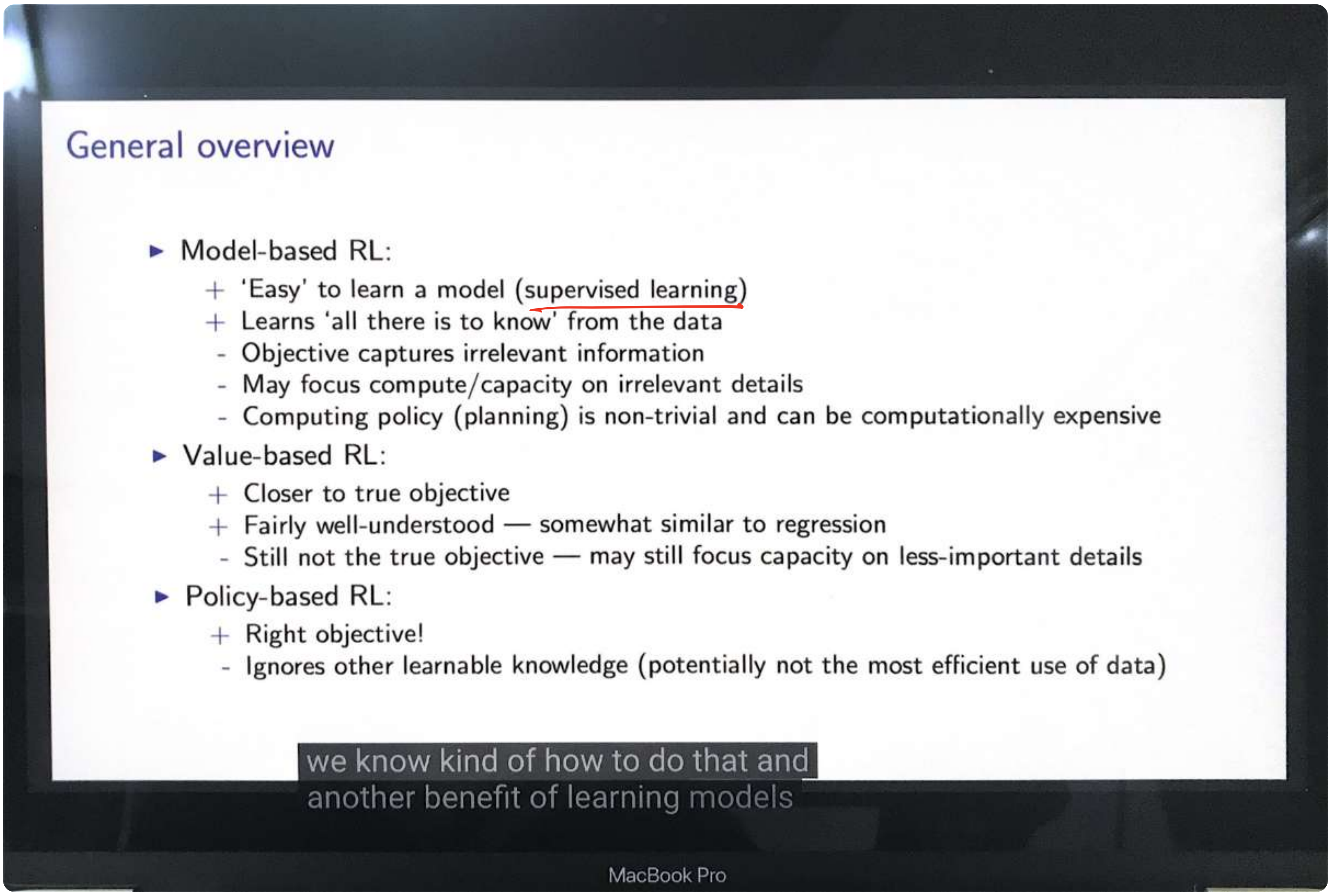
Vapnik's rule:

"Never solve a more general problem as an intermediate step." Vladimir Vapnik (1998)



Value-based

V(s) / Q(s, a) → 选择 policy 做 action

输入一个状态

Policy-based

输出对应的action / action 概率分布

Policy Network:

DNN: $a = \pi(s, \theta)$

$a = \pi(a|s, \theta)$

Policy Gradient: 通过对收益期望求梯度，从而对policy network参数进行更新

Policy gradients / Actor Critics

* Learn policy directly

Policy-based RL

Approximate paramteric value function (previously)

$$\begin{cases} V_w(s) \approx V_\pi(s) \\ q_w(s,a) \approx q_\pi(s,a) \end{cases}$$

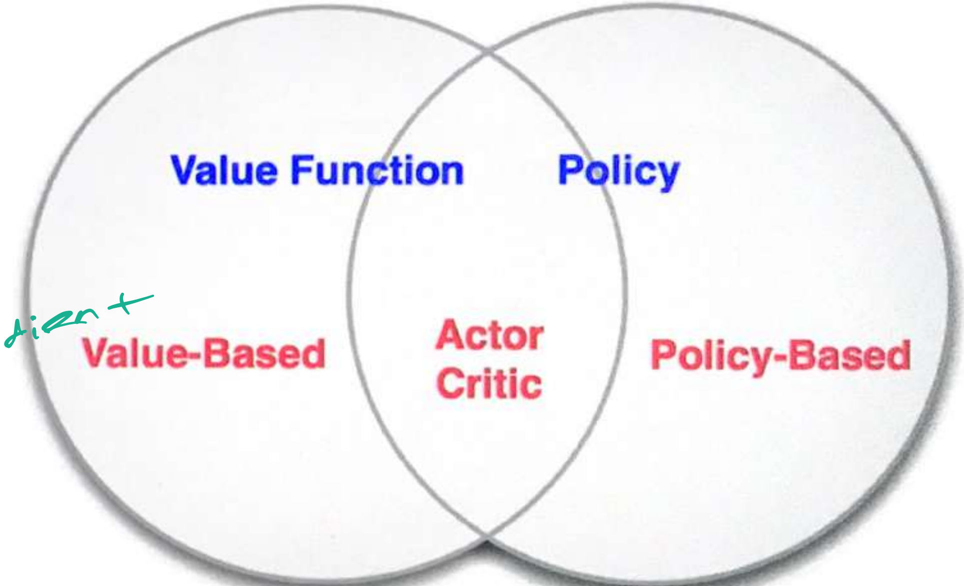
Directly parametrize the policy

$$\boxed{\pi_\theta(a|s) = p(a|s,\theta)}$$

Value-Based and Policy-Based RL

- Value Based
  - Learnt Value Function
  - Implicit policy (e.g. ϵ-greedy)
- Policy Based
  - No Value Function
  - Learnt Policy
- Actor-Critic
  - Learnt Value Function
  - Learnt Policy

policy gradient



Simultaneously learn

terminology there's also something
called actor critic systems and

MacBook Pro

Advantages of Policy-Based RL

learn the optimal stochastic policy

Advantages:

- Good convergence properties
- Easily extended to high-dimensional or continuous action spaces
- Can learn **stochastic** policies
- Sometimes policies are **simple** while values and models are **complex**
  - E.g., rich domain, but optimal is always **go left**

Disadvantages:

- Susceptible to local optima (especially with non-linear FA)
- Obtained knowledge is **specific**, does not always generalize well
- Ignores a lot of information in the data (when used in isolation)

have a nonlinear function you end up in
some local

Policy objective function:

policy $\pi_\theta(s,a)$ with parameters θ

↑

find best θ

start value : $J_1(\theta) = V_{\pi_\theta}(s_1)$

average value : $J_{avV}(\theta) = \sum_s \mu_{\pi_\theta}(s) V_{\pi_\theta}(s)$

$\mu_\pi(s) = p(S_t = s \mid \pi)$ is the probability of being in state s in the long run

Average reward per time-step :

$$J_{avR}(\theta) = \sum_s \mu_{\pi_\theta}(s) \sum_a \pi_\theta(s,a) \sum_r p(r \mid s,a)\, r$$

Policy Optimization

θ : $J(\theta)$ maximize

- Hill climbing
- Genetic algorithms

✓ <u>Gradient ascent</u> :

objective function $J(\theta)$

$$\Delta\theta = \underbrace{\alpha}_{\text{step-size}} \nabla_\theta J(\theta)$$

Policy gradient :

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$

Estimate of the Policy gradient:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_d [V_{\pi_\theta}(s)]$$

Use Monte Carlo samples

to compute this gradient

One-step Case (a contextual bandit)

$$J(\theta) = \mathbb{E}[R(S,A)]$$

Use identity:

$$\nabla_\theta \mathbb{E}[R(S,A)] = \mathbb{E}[\nabla_\theta \log \pi(A|S) R(S,A)]$$

then use the score function trick:

$$\nabla_\theta \mathbb{E}[R(S,A)]$$

$$= \nabla_\theta \sum_s d(s) \sum_a \pi_\theta(a|s) R(s,a)$$

$$= \sum_s d(s) \sum_a \nabla_\theta \pi_\theta(a|s) R(s,a)$$

$$= \sum_s d(s) \sum_a \pi_\theta(a|s) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} R(s,a)$$

$$= \sum_s d(s) \sum_a \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(a|s) R(s,a)$$

$$= \mathbb{E}[\nabla_\theta \log \pi_\theta(A|S) R(S,A)]$$

$$\Downarrow$$

$$\nabla_\theta \mathbb{E}[R(S,A)] = \mathbb{E}[\nabla_\theta \log_{\pi_\theta}(A|S) R(S,A)]$$

Use stochastic policy-gradient update:

$$\theta_{t+1} = \theta_t + \alpha R_{t+1} \nabla_\theta \log \pi_{\theta_t}(A_t|S_t)$$

Softmax policy:

Probability of action is proportional to exponentiated weight.

$$\pi_\theta(a|s) = \frac{e^{h(s,a)}}{\sum_b e^{h(s,b)}}$$

Gradient:

$$\nabla_\theta \log \pi_\theta(a|s)$$

$$= \nabla_\theta h(s,a) - \sum_b \pi_\theta(b|s) \nabla_\theta h(s,b)$$

Policy Gradient Theorem

For any differentiable policy $\pi_\theta(s,a)$

for any of the policy objective functions

$$\begin{cases} J = J_1, \\ J_{avR} \\ \frac{1}{1-\gamma} J_{avV} \end{cases}$$

The policy gradient:

$$\boxed{\nabla_\theta J(\theta) = \mathbb{E}\left[q_{\pi_\theta}(S,A)\,\nabla_\theta \log \pi_\theta(A|S)\right]}$$

Expectation is over both states and actions

Policy gradients on trajectories

derivation

Consider trajectory $\delta = S_0, A_0, R_1, S_1, A_1, R_1, S_2, \dots$ with return $G(\delta)$

$$\nabla_\theta J_\theta(\pi) = \nabla_\theta \mathbb{E}[G(\delta)]$$

$$= \mathbb{E}[G(\delta)\,\nabla_\theta \log p(\delta)]$$

$\nabla_\theta \log p(\delta)$

$$= \nabla_\theta \log [p(S_0)\,\pi(A_0|S_0)\,p(S_1|S_0,A_0)\,\pi(A_1|S_1)\cdots]$$

$$= \nabla_\theta [\log p(S_0) + \log \pi(A_0|S_0) + \log P(S_1|S_0,A_0$$

$$+ \log \pi(A_1|S_1) + \cdots]$$

$$= \nabla_\theta [\log \pi(A_0|S_0) + \log \pi(A_1|S_1) + \cdots]$$

So:

$$\nabla_\theta J_\theta(\pi) = \mathbb{E}\left[G(\delta) \nabla_\theta \sum_{t=0} \log \pi(A_t|S_t) \right]$$

$$= \mathbb{E}\left[\left(\sum_{t=0} R_{t+1} \right) \left(\nabla_\theta \sum_{t=0} \log \pi(A_t|S_t) \right) \right]$$

reduce variance:

$$\mathbb{E}\left[b \nabla_\theta \log \pi(A_t|S_t) \right]$$

$$= \mathbb{E}\left[\sum_a \pi(a|S_t)\, b \nabla_\theta \log \pi(a|S_t) \right]$$

$$= \mathbb{E}\left[b \nabla_\theta \sum_a \pi(a|S_t) \right]$$

$$= \mathbb{E}\left[b \nabla_\theta 1 \right] = 0$$

"subtract a <u>baseline</u> to reduce variance"

$\sum_{t=0}^{k} R_{t+1}$ does NOT depend on

$A_{k+1}, A_{k+2}, \ldots$

$$= \mathbb{E}\left[\sum_{t=0} \nabla_\theta \log \pi(A_t|S_t) \sum_{i=0} R_{i+1} \right]$$

$$= \mathbb{E}\left[\sum_{t=0} \nabla_\theta \log \pi(A_t|S_t) \sum_{i=t} R_{i+1} \right]$$

$$= \mathbb{E}\left[\sum_{t=0} \nabla_\theta \log \pi(A_t|S_t)\, q_\pi(S_t, A_t) \right]$$

A good baseline is $V_\pi(S_t)$

$\nabla_\theta J_\theta(\pi) =$

$$\mathbb{E}\left[\sum_{t=0} \nabla_\theta \log \pi(A_t|S_t) \left(q_\pi(S_t, A_t) - V_\pi(S_t) \right) \right]$$

We estimate $V_w(s)$ explicity and sample

$$\boxed{q_\pi(S_t, A_t) \approx G_t^{(n)}}$$

e.g. $G_t^{(1)} = R_{t+1} + \gamma V_w(S_{t+1})$

Bias in Actor-Critic Algorithms

n-step TD-error:

$$\delta_t^{(n)} = G_t^{(n)} - V_w(S_t)$$

$$= \underbrace{R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_w(S_{t+n})}_{G_t^{(n)}} - V_w(S_t)$$

Estimating the Action-value Function

Critic $\xrightarrow{\text{solving}}$ policy evaluation

Actor Critic

最基础的policy gradient算法只能实现每轮次更新，很难准确地把reward反馈回去，训练效率很大，并且很容易不收敛

Policy Gradient:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)} [r(\tau)]$$

$$= \int_{\tau \sim \pi(\tau)} r(\tau)\, \pi_\theta(\tau)\, d\tau$$

$$\theta^* = \arg\max_\theta (J(\theta))$$

当前路径总的回报

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\boxed{\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t | s_t)} \boxed{\sum_{t=1}^{T} r(s_t, a_t)} \right]$$

获得当前Trace的概率的梯度

1. sample $\{\tau^i\}$ from $\pi_\theta(a_t | s_t)$

2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \sum_{t=1}^{T} r(s_t^i, a_t^i) \right)$

3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Based Actor-Critic:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, Q(s_t, a_t) \right]$$

Advantage Actor Critic (A2C)

$Q(s_t, a_t)$ 有很大的方差

$\Big\downarrow$ 减小方差

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) - V(s_t)$$

$\Big\downarrow$ 结合 $Q(s,a)$ 与 $V(s)$

$$A^\pi(s_t, a_t) = r(s_t, a_t) + V^\pi(s_{t+1}) - V^\pi(s_t)$$

Actor-Critic

Critic Update parameters $\boldsymbol{w}$ of $v_{\boldsymbol{w}}$ by n-step TD (e.g., $n = 1$)

Actor Update θ by policy gradient

function ADVANTAGE ACTOR CRITIC
    Initialise s, θ
    for t = 0, 1, 2, ... **do**
        Sample $A_t \sim \pi_\theta(S_t)$
        Sample R_{t+1} and S_{t+1}
        $\delta_t = R_{t+1} + \gamma v_{\boldsymbol{w}}(S_{t+1}) - v_{\boldsymbol{w}}(S_t)$ [one-step TD-error, or advantage]
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \beta\, \delta_t\, \nabla_{\boldsymbol{w}} v_{\boldsymbol{w}}(S_t)$ [TD(0)]
        $\theta \leftarrow \theta + \alpha\, \delta_t\, \nabla_\theta \log \pi_\theta(A_t \mid S_t)$ [Policy gradient update]
    end for
end function

step but it won't actually do multi it
will do one step temporal

MacBook Pro

Full advantage actor critic agent (A two c)

- Advantage actor critic includes:
  - A representation (e.g., LSTM): $(S_{t-1}, O_t) \mapsto S_t$
  - A network $v_{\mathbf{w}}$: $S \mapsto v$
  - A network π_θ: $S \mapsto \pi$
  - Copies/variants π^m of π_θ to use as policies: $S_t^m \mapsto A_t^m$
  - A n-step TD loss on $v_{\mathbf{w}}$

$$I(\mathbf{w}) = \frac{1}{2}\left(G_t^{(n)} - v_{\mathbf{w}}(S_t)\right)^2$$

where $G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} v_{\mathbf{w}}(S_{t+n})$

  - A n-step REINFORCE 'loss' on π_θ

$$I(\theta) = \left[G_t^{(n)} - v_{\mathbf{w}}(S_t)\right] \log \pi_\theta(A_t|S_t)$$

  - Optimizers to minimize the losses
- Also know as A2C, or A3C (when combined with asynchronous parameter updates)

MacBook Pro

On-policy estimate:

$$G_t^{(n)} = \frac{\pi_\theta(A_t \mid S_t)}{b(A_t \mid S_t)} \left(R_{t+1} + \gamma G_{t+1}^{(n-1),\rho} \right)$$

$$G_t^{(0),\rho} = V_w(S_t) \approx V_\pi(S_t)$$

λ - returns

Multi-step return: (Mixture of n-step returns)

$$G_t^{(n)} = R_{t+1} + \gamma G_{t+1}^{(n-1)}$$

$$G_t^{(0)} = V_w(S_t) \approx V_\pi(S_t)$$

$\Longleftrightarrow$

$$G_t^{\lambda} = R_{t+1} + \gamma(1-\lambda_{t+1}) V_w(S_{t+1}) + \gamma \lambda_{t+1} G_{t+1}^{\lambda}$$

$\lambda_k = 1$ for $k \in \{t+1, \cdots, t+n-1\}$

$\lambda_k = 0$ for $k = t+n$

$\lambda_t \in [0, 1]$ ("λ-return")

Trust region policy optimization

(trust regions)

$\boxed{\text{regularize the policy}}$

prevent instability

Method: limit the difference between subsequent policies

Kullbeck - Leibler Divergence:

$$KL(\pi_{old} \| \pi_\theta)$$

$$= \mathbb{E}\left[\int \pi_{old}(a|s) \log \frac{\pi_\theta(a|s)}{\pi_{old}(a|s)} da \right]$$

$$\underbrace{J(\theta) - \eta \, KL(\pi_{old} \| \pi_\theta)}_{\text{Maximize}}$$

large batches $\begin{cases} \text{TRPO} & 2015 \\ \text{PPO} & 2016 \end{cases}$

Continuous action spaces

$\downarrow$

Gaussian Policy:

$$a \sim \mathcal{N}(\mu(s), \sigma^2)$$

gradient:

$$\nabla_\theta \log \pi_\theta(s, a) = \frac{a - \mu(s)}{\sigma^2} \nabla \mu(s)$$

Continuous actor-critic learning automaton (Cacla)

- $a_t = \text{Actor}_\theta(S_t)$ (get current (continuous) action proposal)
- $A_t \sim \pi(\cdot|S_t, a_t)$ (e.g., $A_t \sim \mathcal{N}(a_t, \Sigma)$) (explore)
- $\delta_t = R_{t+1} + \gamma v_{\boldsymbol{w}}(S_{t+1}) - v_{\boldsymbol{w}}(S_t)$ (compute TD error)
- Update $v_{\boldsymbol{w}}(S_t)$ (e.g., using TD) (policy evaluation)
- If $\delta_t > 0$, update $\text{Actor}_\theta(S_t)$ towards A_t (policy improvement)
- If $\delta_t \leq 0$, do not update Actor_θ

difference difference error is positive
we're happily surprised