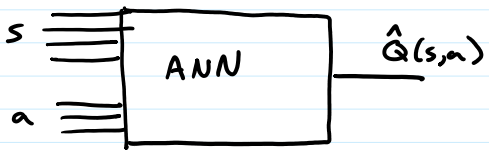
Deep Q Learning



two networks: learning target

play using learning observe (s,a,s',r) → add to replay database

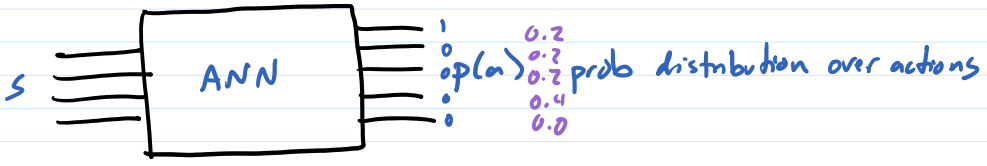
compute error $\hat{Q}_t(s,a) \sim (r + \gamma \max_{a'} \hat{Q}_t(s',a'))$

↳ (s,a,s',r) sampled from replay database

adjust learning accordingly

periodically copy learning to target

REINFORCE: learn policy directly



objective: maximize $E\left[\sum_{t=1}^{T} r_t\right]$

(s_1,a_1) (s_2,a_2) (s_3,a_3) ....

↑ over all possible trajectories τ where probability of τ is given by ANN and its weights θ

$$J(\theta) = \int \pi_\theta(\tau) \cdot r(\tau) \, d\tau$$

$r(\tau)$ → sum of reward at each step

$\pi_\theta(\tau)$ → prob ANN w/ weights θ gives trajectory τ

objective as fxn of weights on neural network

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) \cdot r(\tau) \, d\tau$$

gradient w.r.t. weights θ

how to adjust weights θ to increase $J(\theta)$ most? so in dir of steepest ascent

$$= \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) \, r(\tau) \, d\tau$$

$$= E\left[\nabla_\theta \log \pi_\theta(\tau) \cdot r(\tau)\right]$$

r

$\pi_\theta(\tau) \cdot \nabla_\theta \log \pi_\theta(\tau)$

$= \pi_\theta(\tau) \cdot \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)}$

$= \pi_\theta(\tau) \cdot \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)}$

$= E\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right) \cdot \left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]$

↑ sum over all trajectories τ with prob $\pi_\theta(\tau)$

$\pi_\theta(\tau)$ = prob of trajectory τ

$= \pi_\theta((s_1,a_1)(s_2,a_2)\dots)$

$= p(s_1) \cdot \pi_\theta(a_1|s_1) \cdot p(s_2|s_1,a_1) \cdot \pi_\theta(a_2|s_2) \cdots$

prob ANN w/ weights θ chooses a_1 in s_1 ; prob of transition in MDP

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

estimate expected value by running policy N times

$\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right) \cdot \left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t})\right)$

π_θ: output of ANN

$\log \pi_\theta(\tau) = \log$ ↑

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

$\nabla_\theta \log \pi_\theta(\tau) = \nabla_\theta \log p(s_1) + \sum_{t=1}^{T} (\log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t))$

$\nabla_\theta \log p(s_1) = 0$; $\log p(s_{t+1}|s_t, a_t) = 0$; $\log \pi_\theta(a_t|s_t)$: not functions of θ

REINFORCE

$\tau_i = (s_{i,1}, a_{i,1})(s_{i,2}, a_{i,2}) \dots$

get sample trajectory by running policy

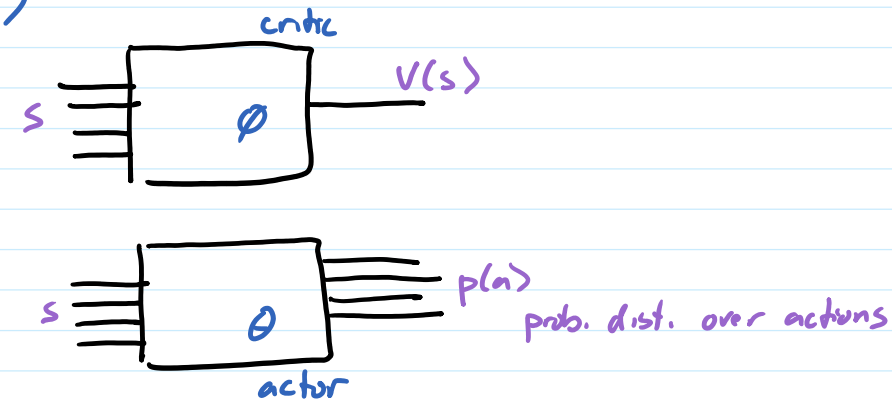
compute estimate of $\nabla_\theta J(\theta)$

update θ : $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

α: learning rate

Advantage Actor-Critic (A2C)

Two networks



get sample trajectory using actor

fit critic to observed rewards — critic predicts $\hat{V}_\phi(s_t)$; observed $r_t + \hat{V}_\phi(s_{t+1})$

evaluate advantage

↳ difference between critic's expectation and actual result

compute gradient $\nabla_\theta J(\theta) = \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \cdot \hat{A}(s_i, a_i)$

do gradient ascent on actor $\theta \leftarrow \theta + \alpha \cdot \nabla_\theta J(\theta)$