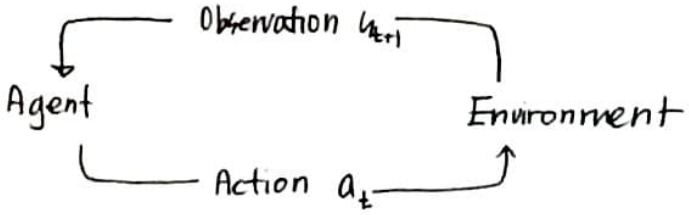
• Deep Reinforcement Learning

→ Basic Model



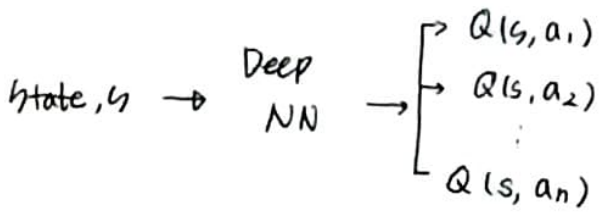
state s_t → action a_t → state s_{t+1}
↳ Reward r_t

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{i=t}^{\infty} \gamma^i r_i$$

$Q(s, a) = \mathbb{E}[R_t]$: expected total future reward in state s with certain a

$\pi^*(s) = \underset{a}{\text{argmax}}\, Q(s,a)$: policy that maximizes future reward

→ DQN

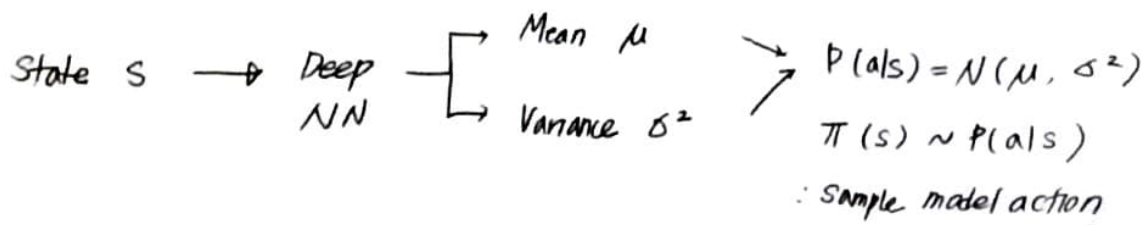


Maximize target return → train the agent

$$L = E\left[\| (r + \gamma \max_{a'} Q(s', a')) - Q(s, a) \|^2\right]$$

but, cannot handle continuous action spaces

→ Policy Gradient: directly optimize the policy Enabling modeling of continuous action space



$$L = -\underbrace{\log P(a_t | s_t)}_{\text{log likelihood}} \underbrace{R_t}_{\text{reward}}$$

+ Training Policy Gradients

1. Initialize the agent
2. Run a policy until termination
3. Record all states, actions, rewards
4. Decrease probability of actions that resulted in low reward
5. Increase probability of actions that resulted in high reward

$$W' = W - \nabla loss$$

$$W' = W + \nabla \log P(a_t | S_t) R_t$$