Lecture 01

Bellman equation :
$$Q(\underset{\text{current state}}{s},\overset{\text{actions}}{a}) = \underset{\text{next step reward}}{r_t} + \overset{0\le\gamma\le1}{\gamma} \max_{a'} Q(s',a')$$

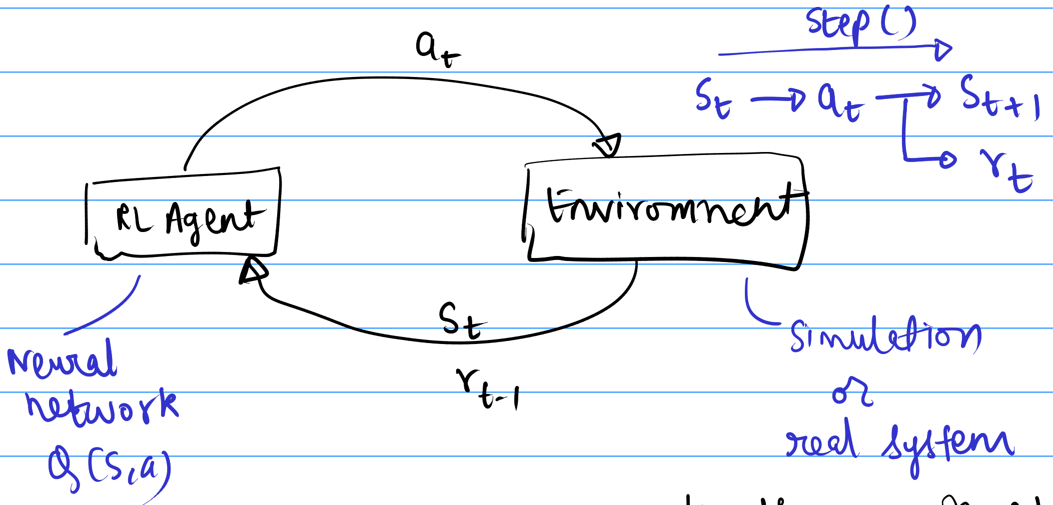
$$G_t = r_t + \gamma \underbrace{r_{t+1} + \gamma^2 r_{t+2} + \dots}_{G_{t+1}}$$

$$= r_t + \gamma G_{t+1}$$

$$\begin{array}{c|cccc} & a_0 & a_1 & \cdots & a_m \\ s_0 & Q(s_0,a_0) & & & \\ s_1 & & & & \\ s_2 & & & Q(s_k,a_l) & \\ \vdots & & & & \\ s_n & & & & \end{array}$$

① Scale is large s , a

② Convergence issues because of lack of time



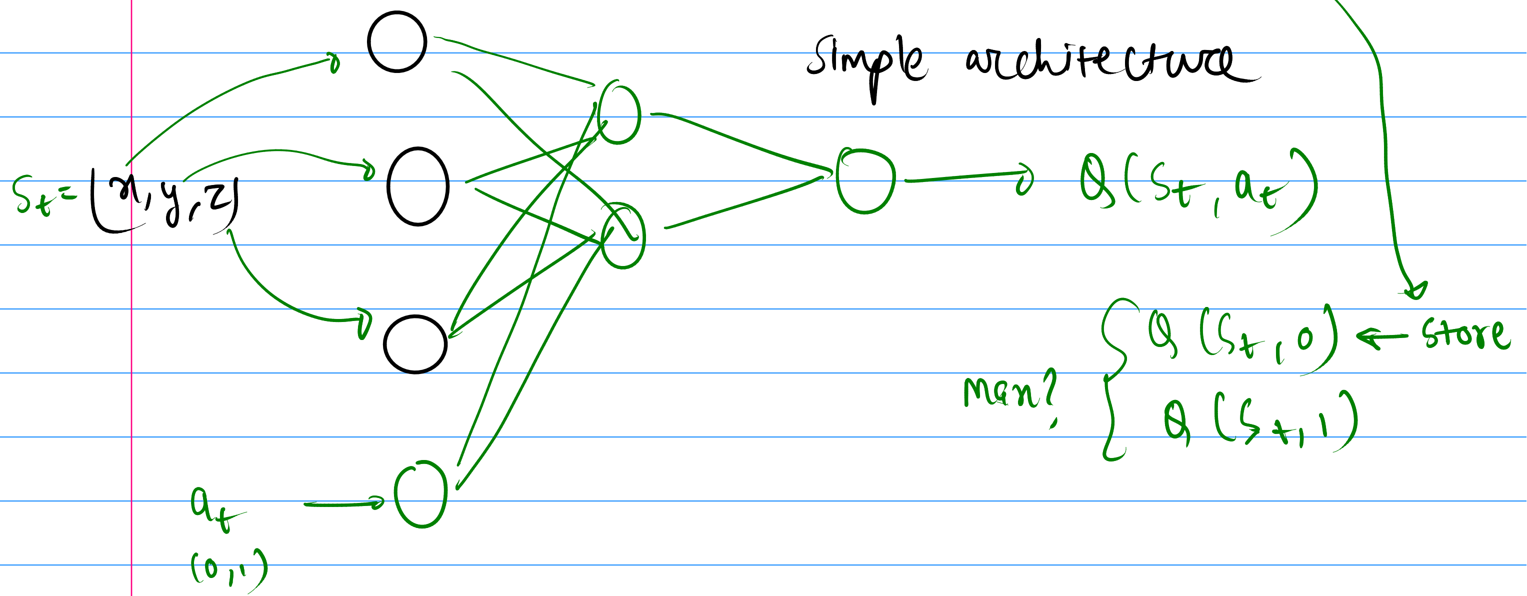
"sample" : $[s_t, a_t, r_t, s_{t+1}]$ $\forall t$ → keep them → off-policy
→ throw away → on-policy

$$\underbrace{Q(s_t,a_t)}_{\text{predictions or inference}} = \underbrace{r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})}_{\text{target}}$$

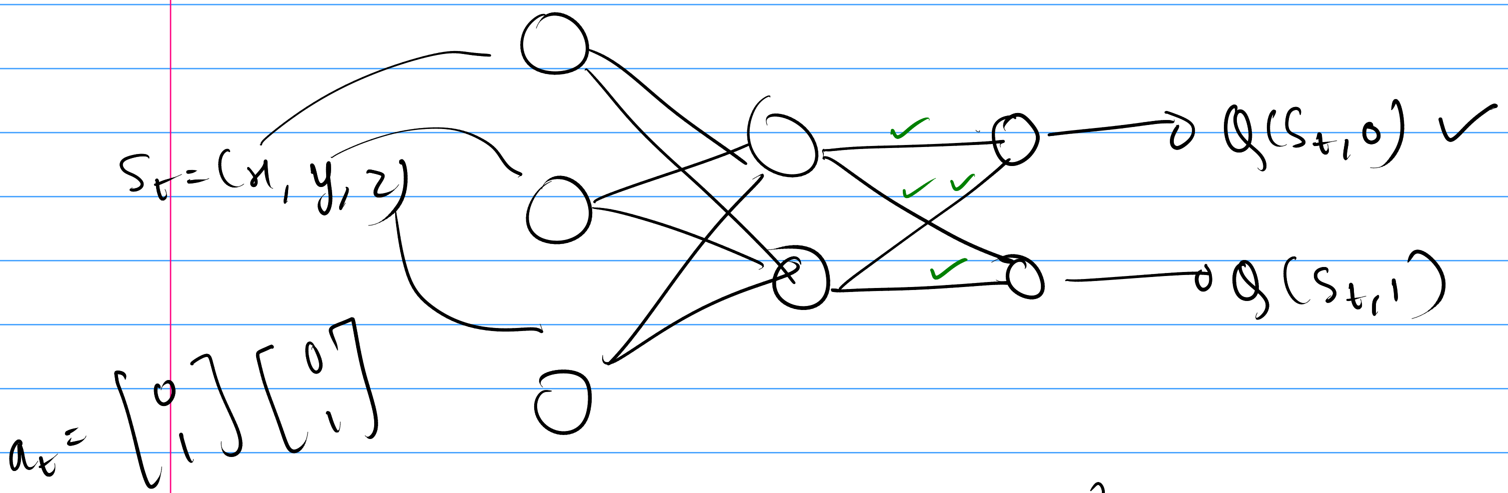
Error = MSE

$$\left[\underbrace{\overset{b}{Q}(s_t, a_t)} - r_t - \gamma \max Q(s_{t+1}, a_{t+1})\right]^2$$

Simple architecture



Practical architecture


$s_t = (x, y, z)$
$a_t = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
$Q(s_t, 0)$ ✓
$Q(s_t, 1)$
✓
✓ ✓
✓

$$\left[Q(s_t, 0) - r + \gamma Q(s_{t+} \ldots)\right]^2 + 0$$

"Vanilla DQN"

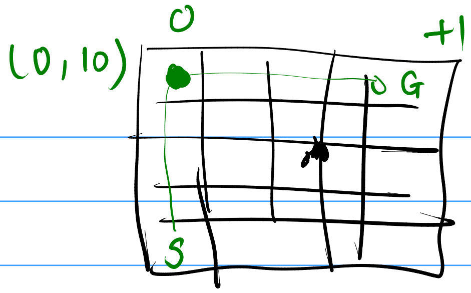
Issues

① Bootstrapping

② Stabilisation

③ $\Delta Q(s, a)$ implemented only by last layer

④ Off/ on policy decisions



$$\overset{(x,y)}{s_t},\ a_t,\ \overset{0}{r_t},\ \overset{Q(s_{t+1}, a_{t+1})}{s_{t+1}}$$

$$\left[\begin{array}{l} s_t, (10,10), a_t, r_t, s_{t+1} \\ \vdots \end{array}\right] 0$$

$$\left[\begin{array}{l} \boxed{s_t, (0,10),}\ a_t, r_t, s_{t+1} \\ \vdots \\ (0,10) \quad (0,10), a_t, 1 \quad , s_{t+1} \end{array}\right]$$

within agent

① choose_action → State as input
compute $Q(s,a)$ by inference on NN
ε-greedy action

② Save to memory → s_t, a_t, r_t, s_{t+1}

③ train → sample from memory
compute loss
update NN

$$q(s,a) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$$

$$= \underbrace{r_t + \gamma r_{t+1}} + \gamma^2 [\underbrace{r_{t+2} \rightarrow \dots}_{G_{t+2}}$$