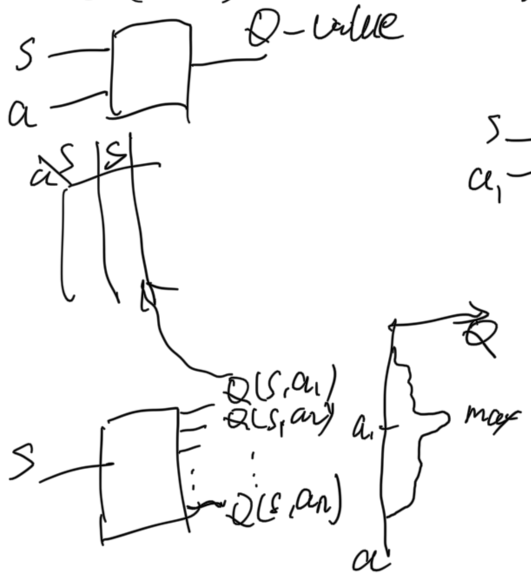
Lecture 14

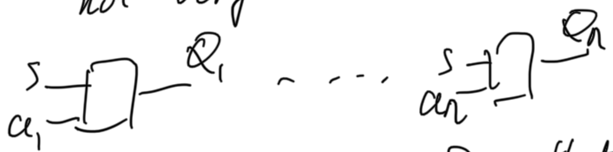
Deep Q-learning

Look at inputs, outputs, Loss, dataset, structure

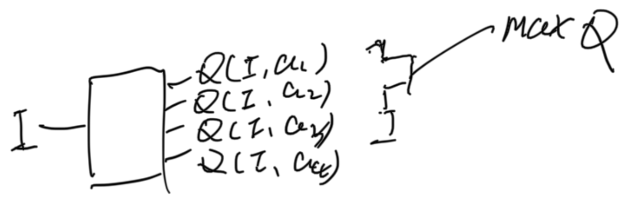
$Q(s, a)$



when we use it, it is not very convenient



then get the max Q_i, that point to the action to take



√ random

① (s, a, r, s')

Episode, run from one random state, for several steps

record $\{$ (s_1, a_1, r_1, s_2) step_index

(s_2, a_2, r_2, s_3)

$\vdots$

$(s_{19}, a_{19}, r_{19}, s_{20})$

run several episodes.

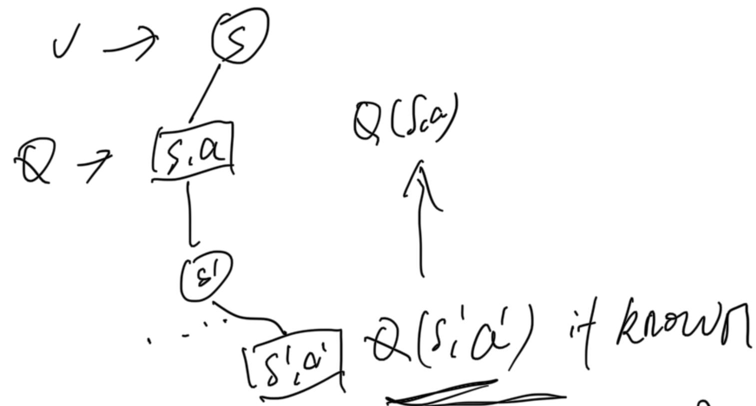
put the record of these episodes into a memory — experience relay memory

remove old episodes, add new episodes.

Memory ⇒ training dataset.

Batch for training, randomly select steps in in memory. calculate loss

(2) $L = \| \text{target} - \text{current\_Network\_output} \|^2$



$$\text{target } Q(s,a) = r + \lambda \max_{a'} Q(s', a')$$

current DQN

ground truth

instead of $Q - Q^*$

we do $Q - Q\_\text{target}$

$$L = \| \underbrace{Q(s,a,w)}_{\text{DQN}} - [r + \lambda \max_{a'} \underbrace{Q(s', a', w)}_{\text{DQN}}] \|^2$$

for one state, one action, ⇒ one step in the batch

we one L, for n steps in the batch.

$$L = \sum L_i$$

$L = \ldots$

③ $w = w - \gamma \frac{\partial L}{\partial w}$, opitimization

↳ give a new $Q(s, a)$

---

revisite memory:

At beginning, DQN is useless, because output is random., random episodes are ok.

when you get better DQN, you want to use it to select action.

ε-greedy, with DQN,

$P(i) = \frac{L_i^\alpha}{\sum_k L_k^\alpha}$ if $\alpha = 0$, uniform

stochastic Prioritization

Double Q-learning:

Train two DQNs: Q_1, Q_2

when decide which action to take in an episode,

use ε-greedy with $Q_1 + Q_2$

Calculating Loss.

$$L = (\underline{\text{target}} - \underline{\text{current DQN output}})^2$$

$$\text{target} = r + \lambda \max_{a'} (Q(s', a'))$$

with 50/50 chance:

$$\rightarrow L = r + \lambda \max_{a'} Q_1(s', a') - Q_2(s, a)$$

select a', use $Q_2(s', a')$ to select a'

→ update Q_2

another 50/50 chance:

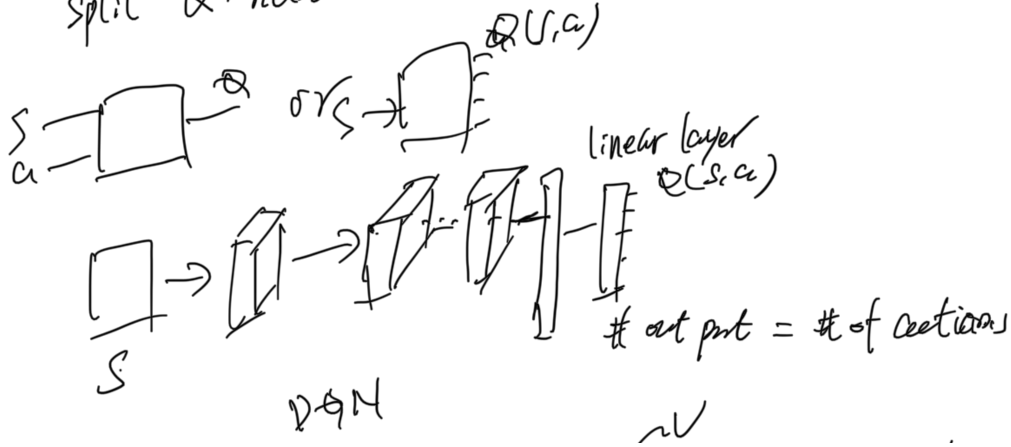
$$\rightarrow L = r + \lambda \max_{a'} Q_2(s', a') - Q_1(s, a)$$

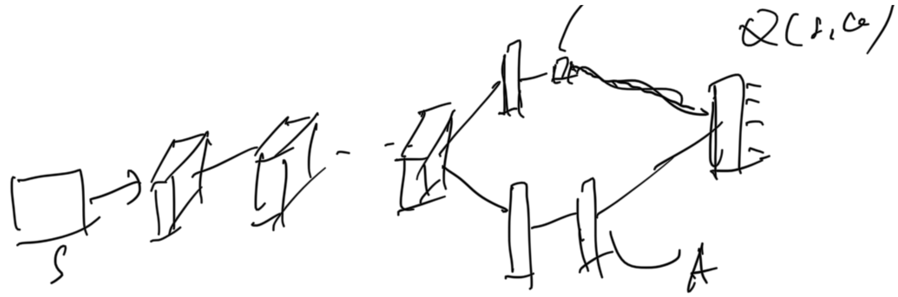
select a', use $Q_1(s', a')$

→ update Q_1

Dueling network.

split Q-network into two





V-function, Q-function

$$Q(s,a) \approx V(s) + A(s,a)$$

$$A(s,a) = Q(s,a) - V(s)$$

$\uparrow$ Advantage function.