Lecture 13

state-action-reward-state-action (SARSA)

Q-learning:

Go through a number of episodes

start an episode with an S

Go through the episode step by step

Sarsa

$Q = Q\text{-new}$ — give you policy

based on Q, choose an action a. — ϵ-greedy

base on

$V = \max_a (Q(s,a))$

or

random

Take action a, get reward, get to a new state s'

$Q_{-curr} = r + \lambda V(s')$

— Update

$Q_{-new}(s,a) = Q(s,a) + \alpha(Q_{-curr}(s,a) - Q(s,a))$

if $\alpha = 1$

$s \leftarrow s'$

continue until to the end of the episode.

Sarsa:

Go through a number of episode

start with an S

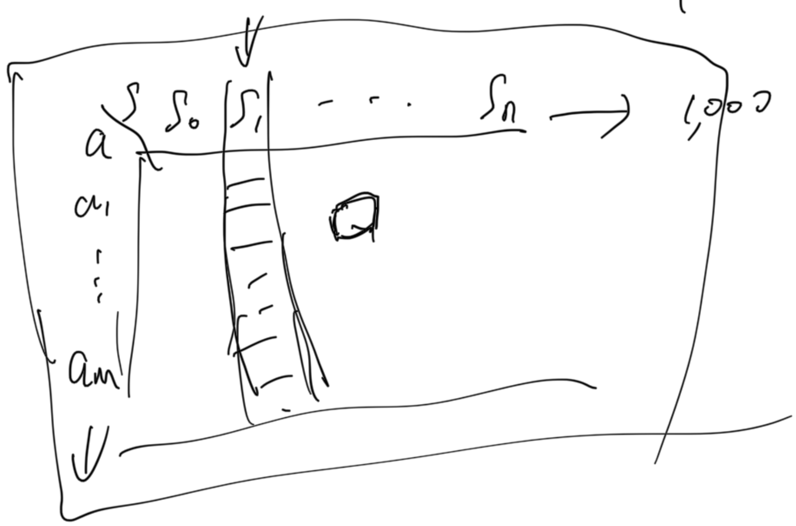
→ Choose actions use the current Q (ε-greedy)

Then carry out those chosen actions.

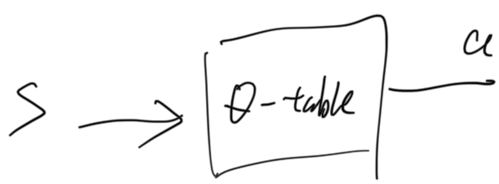
take one action at a time, get reward,

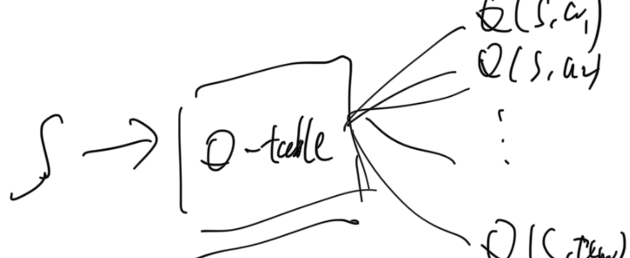
reach to a new state s'

$$Q\_new(s,a) \Leftarrow Q(s,a) + \alpha(Q\_curren(s,a)' - Q(s,a)$$



Q-learn



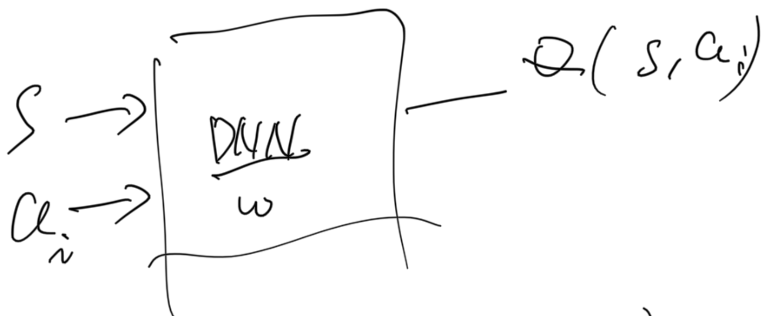


X	Y
x_1	y_1
x_2	y_2
⋮	⋮
x_n	y_n

$\Rightarrow y = f(x)$

look up table

regression



$Q(s,a) \rightarrow Q(s, a, w)$

w — neural network

if you know the ground truth $\hat{Q}(s,a)$

input : $(s, a)_k$, out is one-D $\hat{Q}(s,a)$

s X_1	a X_2	$Q(s,a)$ Y

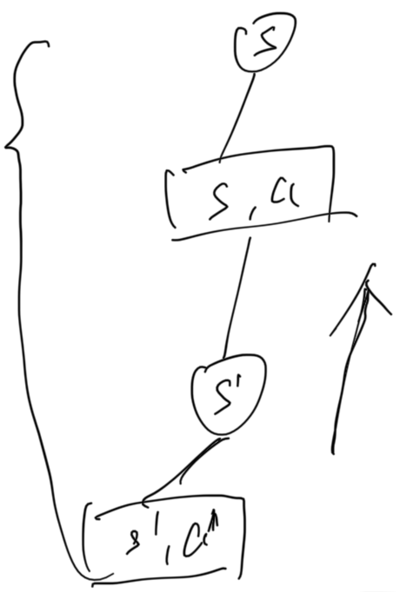
$$L = \left(Q(s,a,w) - \hat{Q}(s,a)\right)^2$$

$\frac{\partial L}{\partial w}$

Q_0 $\underline{Q\text{-truth}}$

$$\begin{cases} Q(s,a,w) & \text{— current QDNN} \\ \underline{Q\_curr(s,a)} = r + \gamma \underline{\max_{a'} Q(s',a',w)} & \end{cases}$$

s' is next state after take action a.



QDNN gives $\underline{Q_1(s,a)}$

calculate with $\underline{Q_2(s',a')}$

$\underline{Q(s,a)}$

↑

$V(s')$

↑

$\underline{Q(s',a')}$ — one step into future step

↑ Q-table, $Q(s',a',\underline{w})$

$s, a \to \boxed{w} \to Q(s,a)$ $\qquad$ $s', a' \to \boxed{w} \to Q(s',a')$

↑ $\qquad$ ↑

$\underline{(s,a,r,s')}$

$$\underline{L(w)} = \left(\underline{Q\_curr(s,a,w)} - \underline{Q(s,a,w)}\right)^2$$

↑ contain r $\qquad$ ↑ purely from NN

$\uparrow$

$\hat{Q}_1$ $\qquad\qquad$ Q_0

$$\frac{\partial L(w)}{\partial w}$$

$$L = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$, N samples. for regular DNN

for one $\underline{(s_i, a_i, r_i, s_{i+1})}$, we have on L_i

$$L = \sum_{i=1}^{N} L_i$$, N could be very large.

Get all (s_i, a_i, r_i, s_{i+1}),
sample, put into mini-batch,
calculate loss use mini-batch

Double Q-learning:
Train two Q's. : Q_1 and Q_2
Do Q-learning on both

$$\underline{Q_{1\_new}}(s, a) = r + \lambda \max(\underline{Q_2}(s', a'))$$

alternate

$$Q_{2,curr}(s,a) = r + \lambda \max\left(Q_1(s',a')\right)$$