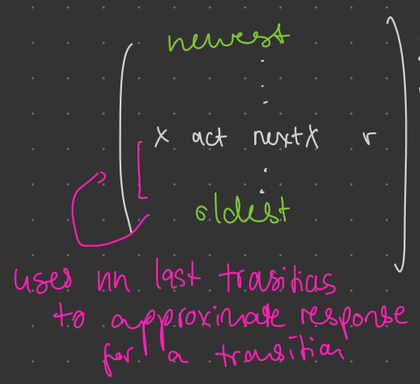
nn-step Q-learning

→ every n steps nn-step Q1 function is called through game_events_occured

→ n new transitions have been added to the buffer



uses nn last transitions to approximate response for a transition

n new transitions → compute $\sum_{t'=t+1}^{t+nn} \gamma^{t'-(t+1)} R_{\tau,t} + \gamma^{nn} \max Q(S_{\tau,t+n}, a)$

buffersize - n old transitions → compute $R_{\tau,t} + \gamma \max Q(S_{\tau,t+1}, a)$

$=: bs$

self.gamma_matrix

$*$

actual rewards

temp

creation:

for i in range (buffersize - nn)

$\overrightarrow{gamma} = np.array([1, \gamma, \dots, \gamma^{nn-1}])$ add to lines

(length nn)

append lists

$[0] + \overrightarrow{gamma} + [0 \text{ for i in range}(buffersize - n + 1)] + \overrightarrow{gamma}$

$+ [0 \text{ for i in range}(buffersize - n + 1)]$

$\longrightarrow$ gammat $\overset{!}{=}$ $\underbrace{[0]}_{\text{list}} + \underbrace{\overrightarrow{gamma}^{-1}}_{\text{list}} + \underbrace{\oplus}_{buffersize-(n+1)}$ $\left[\underbrace{[0]_{bs-n+1} \oplus \overrightarrow{gamma}^{-1}}_{\text{list}}\right]$

append

gammat as list of floats $\xrightarrow{np.Array}$ gammat $\xrightarrow{np.reshape}$ gammat

$(bs * (bs-n))$ $(bs-n, bs)$