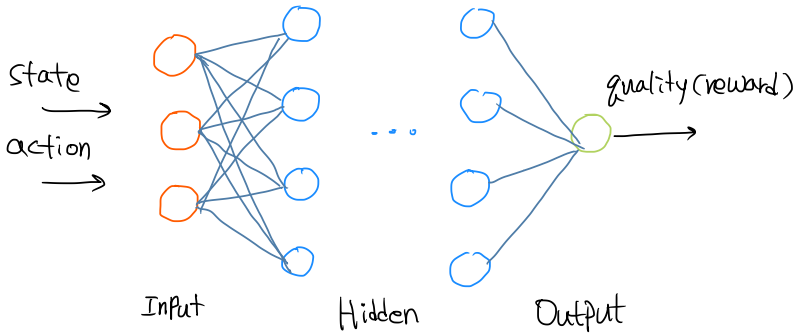
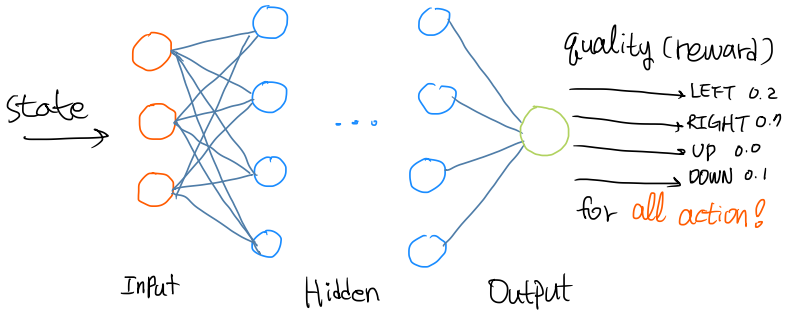
Q-Network

- Q-Table 이 굉장히 쉽고 간결하지만 실전에 적용하려 하면 크기가 너무 커져서 실전문제에서 사용하기에 적합하지 않다.

- 그래서 나온 것이 Q-Network!



이 형태 모델은 잘 사용 X



이 형태의 모델을 사용하게 된다!

Deep Q- learning Algorithm

Algorithm 1 Deep Q-learning ~~with Experience Replay~~

~~Initialize replay memory $\mathcal{D}$ to capacity N~~
Initialize action-value function Q with random weights 랜덤 weight로 Q 초기화
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$ 전처리 $\phi = S$ 라 생각하면됨
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t ← ϵ-greedy
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$ ϵ-greedy 나 가장 좋은 action 값을 선택
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    ~~Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$~~
    ~~Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$~~
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$ 다음상태에서 받을 수 있는 보상 / 마지막상태를 의미 (Goal)
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

학습하는 부분

실생활도 non-deterministic world 이므로 된다!

Target / Label