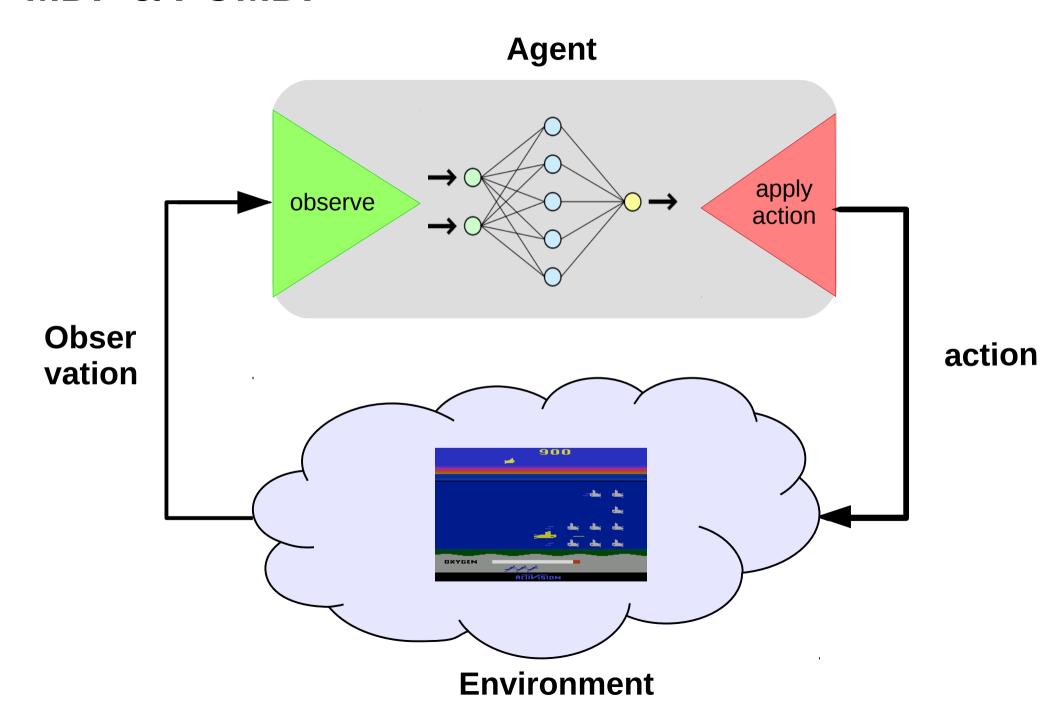
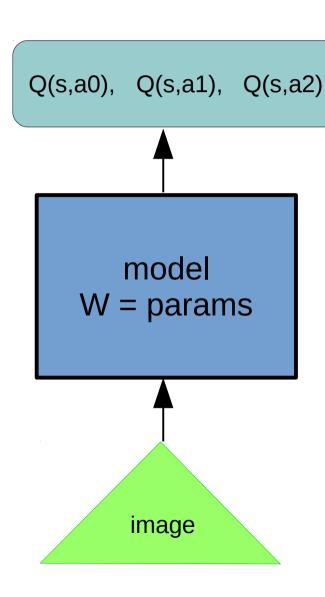
MDP & POMDP



Approximate Q-learning nice and simple



Q-values:

$$\hat{Q}(s_t, a_t) = r + \gamma \cdot argmax_a' \hat{Q}(s_{t+1}, a')$$

Objective:

$$L = (Q(s_t, a_t) - r + \gamma \cdot argmax_a' Q(s_{t+1}, a'))^2$$

Gradient step:

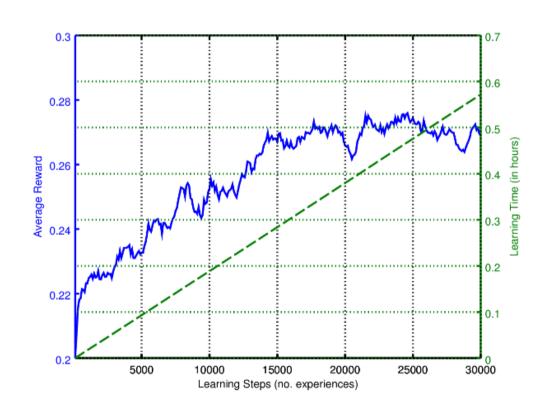
$$w_{t+1} = w_t - \alpha \cdot \frac{\delta L}{\delta w}$$

DQN ∈-greedy rule nutrition facts (tune ϵ or use apply action probabilistic rule) action Qvalues Qvalues is a dense layer with **no** nonlinearity Dense Change axis Any neural order to fit in network you with lasagne can think of. convolutions conv Conv1 pool dense dropout batchnorm image Dimshuffle Conv0 (i, w, h, 3)(i,3,w,h)



Approximate Q-learning problems

- Training samples are not "i.i.d",
- Model forgets parts of environment it haven't visited for some time,
- Fallbacks on the learning curve
- Any ideas?



Deep Q-learning Multiple agent trick

Idea: Throw in several agents with shared **W**.

- Chances are, they will be exploring different parts of the environment,
- More stable training,
- Requires a lot of interaction,
- Alternative to experience replay.

