Deep Q learning

Review

Model-Based

- ML MB RL
- Dyna
- Prioritized Sweeping
- Bayesian RL

Model-Free

- Sarsa — On-Policy
- Q-learning — Off-Policy
- Eligibility Traces
- Double Q-Learning

} Classical / Traditional RL

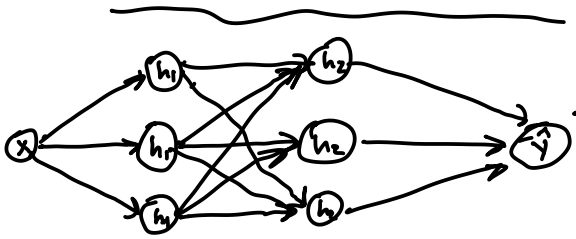
---

1. Ex/Ex
2. Credit Assignment
3. **Generalization** ←

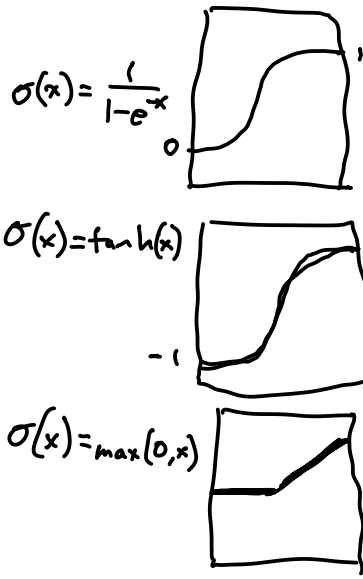
Function Approx

$Q_\theta(s,a)$

Neural Networks



$\mathcal{L}(\hat{y})$ $(\hat{y}-y)^2$ y

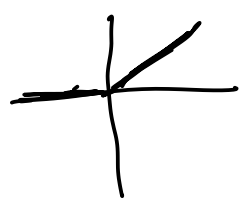


$$i_1 = w_1^T x + b_1$$

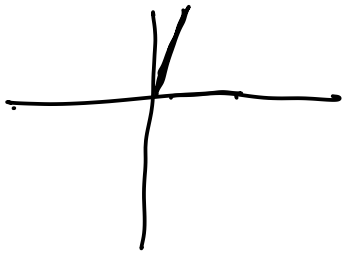
$$h_1 = \sigma(w_1^T x + b_1)$$

$$\hat{y} = w_3^T \sigma_2\left(w_2^T \sigma_1(w_1^T x + b_1) + b_2\right) + b_3\Big)$$

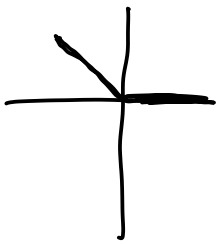
$$\hat{y} = W_3\,\sigma_2\left(W_2\,\sigma_1(W_1 x + b_1) + b_2\right) + b_3\Big)$$



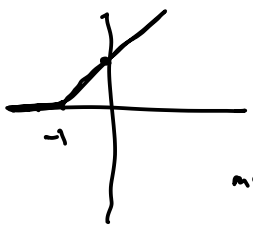
relu(x)



relu(2x)

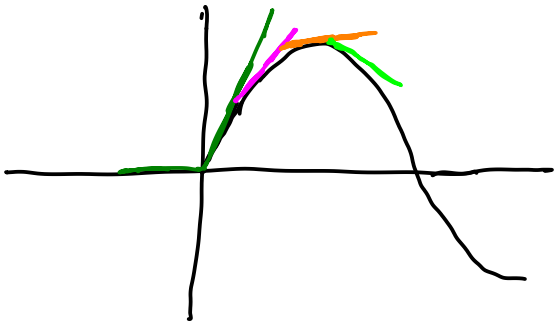


relu(-x)



max(0, x+1)

relu(x+1)



Universal Function Approximator

Regression: Choose W, b $\quad \underset{}{\operatorname{argmin}} \underline{(m(x)-y)^2}$

$$\hat{y} = W_2 \sigma(W_1 x + b_1) + b_2$$

$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \left(\frac{\partial \hat{y}}{\partial W_2} \right)$$

$$= \frac{\partial \mathcal{L}}{\partial \hat{y}} \sigma(W_1 x + b_1)$$

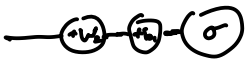
update W_2

$$W_2 \leftarrow W_2 - \alpha \frac{\partial \mathcal{L}}{\partial W_2}$$

Julia Flux

Python/C++ TensorFlow
└ Keras

Computational Graph



Deep Q Learning

Approximate $Q(s,a)$ with $Q_\theta(s,a)$

$$\mathcal{L}(\theta) = \left(\underline{R(s,a) + \gamma \max_{a'} Q_\theta(s',a')} - \underline{Q_\theta(s,a)} \right)^2$$

Naive Deep Q Learning

loop
- choose a
- observe s', r
- $y = r + \gamma \max_{a'} Q_\theta(s',a')$
- $\theta \leftarrow \theta - \alpha \frac{dQ}{d\theta}(s,a)(Q_\theta(s,a) - y)$ — take step along gradient of Q wrt. θ

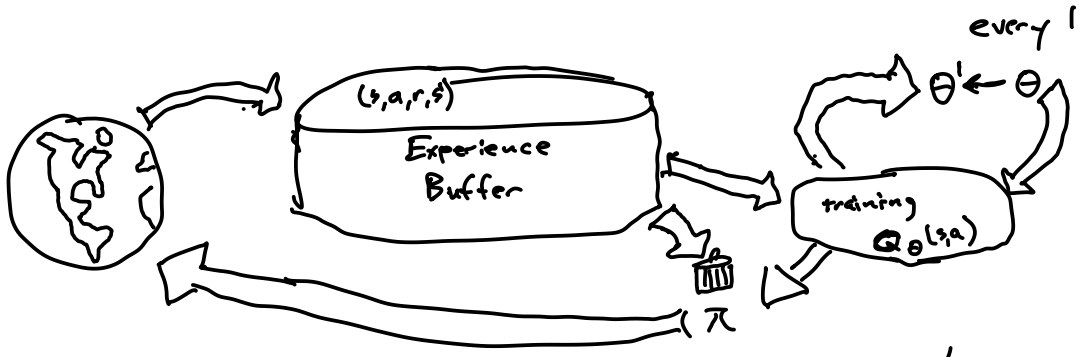
Problem
1. Fitting a moving target
2. batches of size 1
3. data is highly correlated

$(s,a,r,s') \quad (s,a,r,s')$

Soln. to 1.: Freeze target $\theta \to \theta'$

$$y = r + \gamma \max_{a'} Q_{\theta'}(s', a')$$

Soln. to 2+3: Use experience replay buffer



$$\left(r + \gamma \max_{a'} Q_{\theta'}(s,a) - Q_\theta(s,a)\right)^2$$

DQN

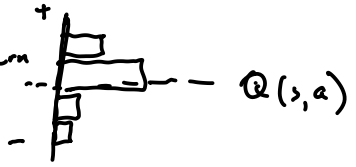
- Double DQN

$$\left(r + \gamma Q_{\theta'}\left(s', \operatorname{argmax}_{a'} Q_\theta(s',a')\right) - Q_\theta(s,a)\right)^2$$

- Prioritized Replay
  (s,a,r,s') chosen in proportion to size of last TD error
- Dueling Networks
  $Q(s,a) = A(s,a) + V(s)$ ← value, ← advantage
- n-step Multistep learning

$$\left(r_t + \gamma r_{t+1} + \dots \gamma^{n-1} r_{t+n-1} + \gamma^n \max_{a'} Q_{\theta'}(s_{t+n}, a') - Q_\theta(s_t, a_t)\right)^2$$

- Distributional RL
  instead of learning Q, learn distribution of returns



- Noisy Nets
  add noise in Qnetwork

Rainbow