Q Learning

while not done

  $s \leftarrow s_0$ initial state

  while s not terminal (episode)

    choose action a — ε-greedy

    observe transition (s, a, r, s') — r: immediate reward $(R(s,s',a))$

    update $Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$

α: learning rate (can ↓ as episodes ↑)

$\max_{a'} Q(s',a') - Q(s,a)$: surprise (error)

$\max_{a'} Q(s',a')$: $V(s')$

Q-learning converges if $\sum_{t=1}^{\infty} \alpha_t(s,a) = \infty$ — so requires that each (s,a) is explored infinitely often

$\sum_{t=1}^{\infty} \alpha_t^2(s,a) < \infty$ — so decrease α as time goes on, but not so fast to ruin 1st constraint

Linear Approximator

Define features of states and possibly actions

Ms. Pac-Man

$f_1(s,a) = \begin{cases} 1 & \text{if move is towards ghost} \\ 0 & \text{otherwise} \end{cases}$

$f_2(s,a) = \begin{cases} 1 & \text{if move is towards power pill} \\ 0 & \text{otherwise} \end{cases}$

$\vdots$

QFL: yards-to-score / time left

80 yards to score, 80 seconds left = 1

40 yards to score, 40 seconds left = 1

yards-to-first/downs left

if feature f is a fxn of state only (doesn't depend on action) then make a copy for each action a

$f_a(s,a') = \begin{cases} f(s) & \text{if } a=a' \\ 0 & \text{otherwise} \end{cases}$

normalize features to be in same range

can quantize features — turn continuous features into discrete

low ≤ 2.5 $\Rightarrow$ 0

mid $\Rightarrow$ 0.5

high ≥ 8 $\Rightarrow$ 1.0

can separate quantized features

$f_{\text{short-yardage}} = \begin{cases} 1 & \text{if yards-to-first/downs left} \leq 2.5 \\ 0 & \text{otherwise} \end{cases}$

$f_{\text{long-yardage}} = \begin{cases} 1 & \text{if yards-to-first/downs left} \geq 8 \\ 0 & \text{otherwise} \end{cases}$

$f_{\text{medium-yardage}} = \{ \ldots$

$$Q(s,a) = w_1 \cdot f_1(s,a) + \cdots + w_n f_n(s,a)$$

In state s

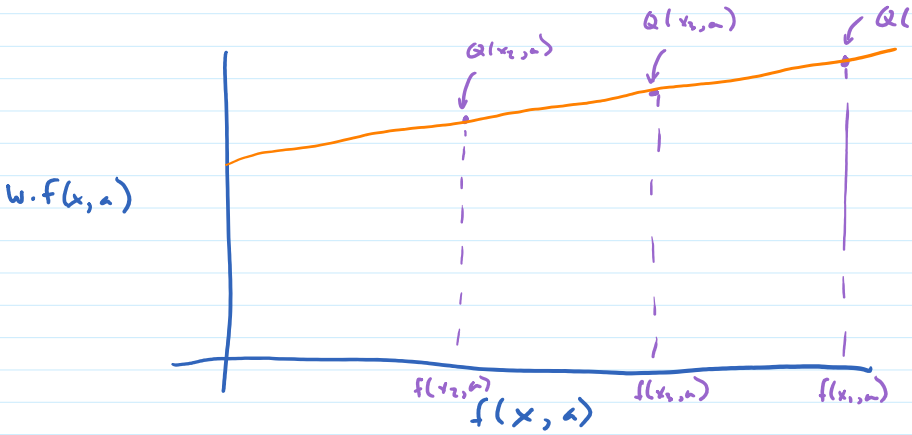
Choose action a using exploit/explore policy (for example ε-greedy)

Observe transition (s, a, r, s')

Update

~~$Q(s,a) \leftarrow Q(s,a) + \alpha(\max_{a'} Q(s',a') - Q(s,a))$~~

$$w_i \leftarrow w_i + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right) \cdot f_i(s,a)$$



$y = mx + b$

make sure features have a bias term so linear combination can approx line w/ non-zero intercept

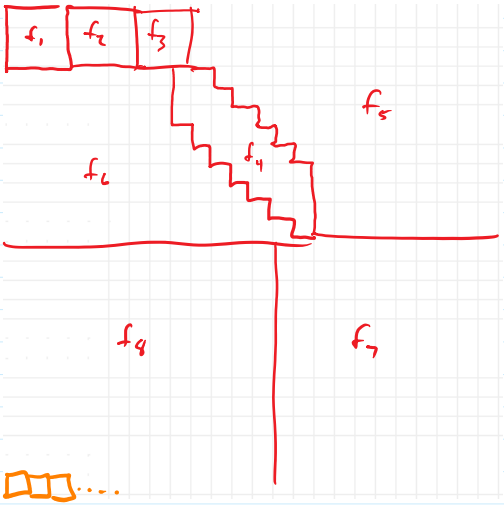
can introduce non-linearity in features

If features are functions of state only, are binary (0 or 1), and $\sum_i f_i(s) = 1$ (so mutually exclusive with 1 set to 1)

$$w_i \leftarrow w_i + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right) \cdot f_i(s,a)$$

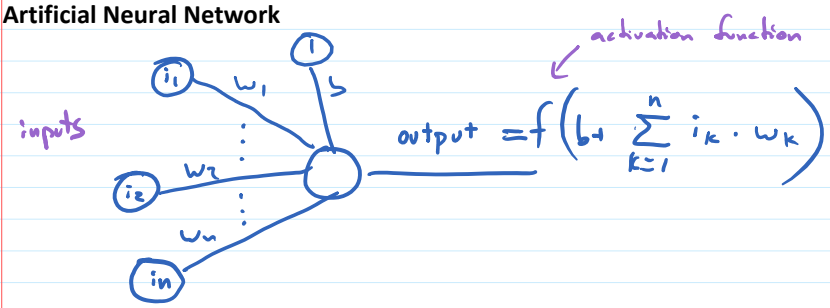
now 0 for all but 1 i

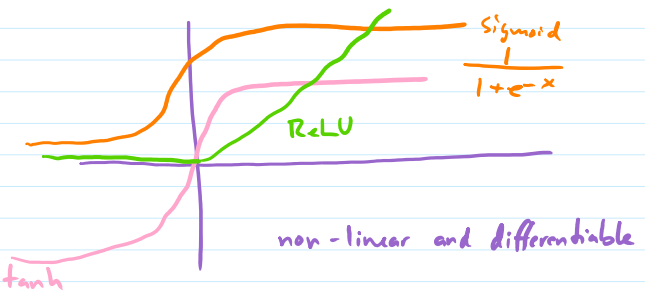
let i be the f_i s.t. $f_i(s) = 1$ (so for $i \neq j$ $f_j(s) = 0$)
i' be the $f_{i'}$ s.t. $f_{i'}(s') = 1$
the one update is $w_i \leftarrow w_i + \alpha\left(r + \gamma \max_{a'} w_{i'} - w_i\right)$
same as original Q-learning!
we've just renamed $Q(s,a)$ to w_i

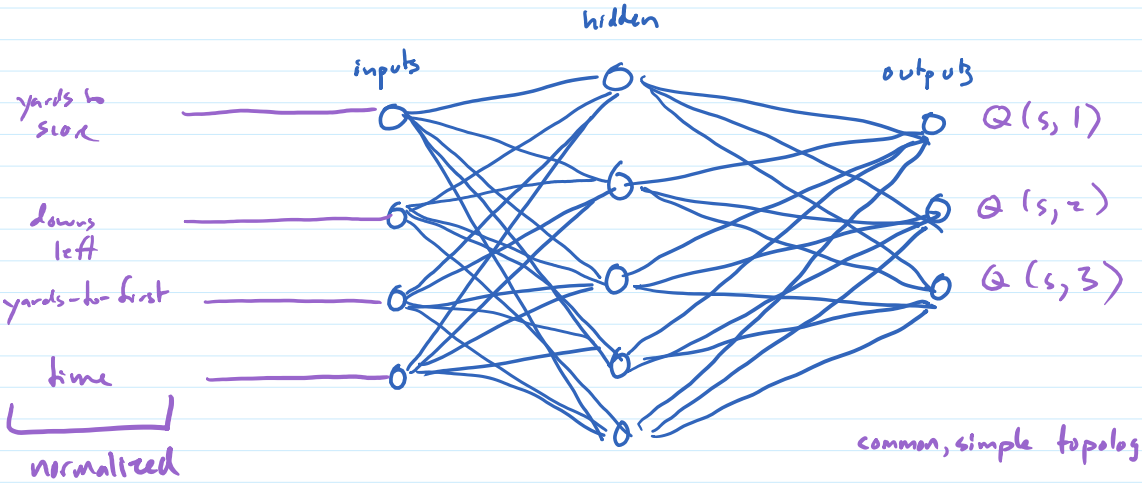


if features partition state space into individual states
|||
normal Q-learning

Artificial Neural Network







common, simple topology: fully-connected feed-forward

supervised learning: have some known input/output pairs, want to generalize to unknown inputs (or know all and want more compact representation)

initialize weights randomly

until sufficiently trained
- compute output for training examples
- compute distance between network's output and correct output
- adjust weights to decrease that distance

for reinforcement learning (no known examples), make up "correct" output by simulation

observed $Q(s,a) = r + \gamma \cdot \max_{a'} Q(s',a')$

observed reward; output of ANN w/ current weights

deal with sparse rewards (QFL: only reward is +1 win at end of game)
by reward shaping: introduce artificial rewards for results of actions that are good (for some definition of "good")