## Evaluating a Policy

**Policy:** function $\pi$: observed info → action
position

$v(\pi)$ = expected reward following policy $\pi$ — prob of win for QFL

sample: noisy but quick

exact: possibly slow

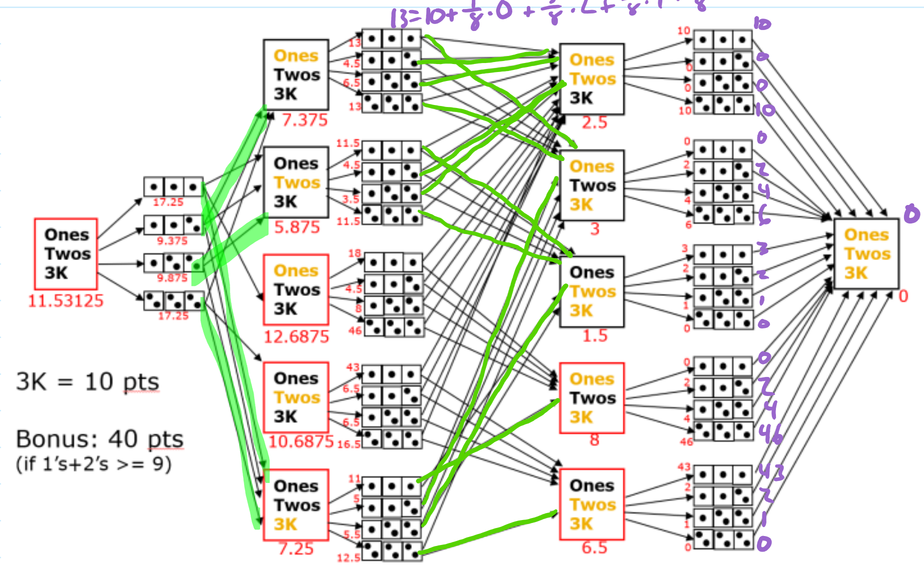
↓

$$V_\pi(s) = \begin{cases} \text{value determined by game} & \text{if } s \text{ is terminal} \\ \sum_{s'} \underbrace{P(s \xrightarrow{\pi(s)} s')}_{\text{prob selected action results in state } s'} \cdot \Big( \underbrace{R(s \xrightarrow{\pi(s)} s')}_{\text{reward earned by selected action}} + \overbrace{V_\pi(s')}^{\text{future reward earned from new state}} \Big) & \end{cases}$$

for a finite game, can compute exactly!

red = values under optimal policy
purple = values under policy defined by choices highlighted in green

$13 = 10 + \frac{1}{8}\cdot 0 + \frac{3}{8}\cdot 2 + \frac{3}{8}\cdot 4 + \frac{1}{8}\cdot 6$




value of taking action $a$ in state $s$ $= Q(s,a)$

$$V_a(s) = \sum_{P_a(s \to s') \neq 0} P_a(s \to s')\big(R_a(s \to s') + V(s')\big)$$

special case of MDP ($\gamma = 1$, finite)

$\pi_{OPT}(s) = \underset{a}{\arg\max}\ V_a(s)$ optimal action is one that maximizes value

$V_{OPT}(s) = V_{\pi_{OPT}(s)}(s)$ value of state = value of optimal action from that state

# Temporal Difference (TD) Learning

Learning Values

need discount to avoid getting trapped in cycles

reward 1

observe $s \xrightarrow{a = \pi(s)} s'$ giving immediate reward $R(s, s', a)$

get discounted future reward $\underbrace{\gamma}_{\text{discount}} \cdot \overbrace{V^{\pi}(s')}^{\text{estimate of future reward from } s' \text{ following policy } \pi}$

discount (maybe 1 for short finite game)

sample of $V^{\pi}(s) = R(s, s', a) + \gamma V^{\pi}(s')$

update estimate $V^{\pi}(s) \leftarrow V^{\pi}(s) + \underbrace{\alpha}_{\text{learning rate}} \big( \underbrace{R(s,s',a) + \gamma V^{\pi}(s') - V^{\pi}(s)}_{\text{error}} \big)$

$$= (1 - \alpha) V^{\pi}(s) + \alpha \big( R(s,s',a) + \gamma V^{\pi}(s') \big)$$

$$\max_a \sum_{s \to s'} P(s \xrightarrow{a} s') \cdot \big( R(s \xrightarrow{a} s') + V(s') \big)$$

$\varepsilon$-greedy: w/ prob $\varepsilon$ choose random
$1 - \varepsilon$ choose action $a$

but requires knowledge of model:
$P(s \xrightarrow{a} s')$ and $R(s \xrightarrow{a} s')$

# Q Learning

$Q(s,a)$ = estimate of expected discounted future reward when choosing action a in state s

$V(s) \approx \max_a Q(s,a)$

if you know this

initialize $Q(s,a) = \begin{cases} R(s) & \text{for terminal } s \\ 0 & \text{otherwise} \end{cases}$

while not done

sample: $r + \gamma V(s')$ — $\max_a Q(s',a)$

episode

$s \leftarrow \underline{s_0}$ initial state

while s not terminal

choose action a

ε-greedy is good
prob ε choose a randomly uniformly
1-ε choose $\arg\max_a Q(s,a)$

observe transition $(s, a, \underline{r}, s')$ immediate reward $(R(s,s',a))$

update $Q(s,a) \leftarrow Q(s,a) + \underbrace{\alpha}_{\text{learning rate}} \underbrace{\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)}_{\text{surprise (error)}}$

learning rate
(can ↓ as episodes ↑)

surprise
(error)

## Function Approximators

– learn a fxn that approximates $Q(s,a)$

### Linear Approximator

Define features of states and possibly actions

↳ yards-to-1st-down / plays left

can ignore action for features of state only

$f_1(s,a)$ — on pace to earn upper bonus

$f_2(s,a)$ — is chance unused

$\vdots$ — both LS, SS unused

$\vdots$ — upper category $a$ is unused

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

learn the weights

In state $s$

Choose action $a$ using exploit/explore policy

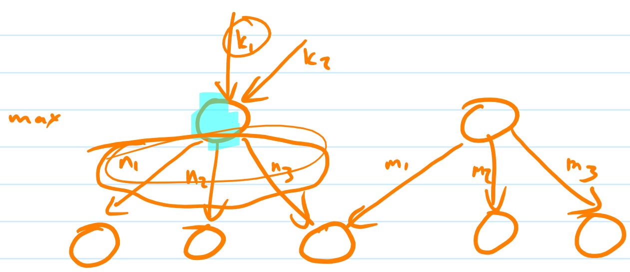
Observe transition $(s, a, r, s')$

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

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

key

$(p, c)$




$$r + \sqrt{\frac{2 \ln (n_1 + n_2 + n_3)}{n_i}}$$