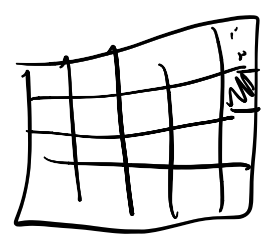
Lecture 24: RL

- HW #5 released
  - Due 4/21
- Last time
  - Dynamic programming
  - Markov decision processes
  - Value iteration
- Today

core
  - Value iteration, $\overbrace{\text{Q-iteration}}^{\text{Jacob's made-up term}}$
  - Learn from Data
  - Function approximation

advanced
  - TD learning
  - Policy gradient



Value iteration

MDP w/ transition probability $P(s' \mid a, s)$ "$T(s', a, s)$"

reward $R(s', a, s)$

discount γ

"$V^*(s)$"

Value function $V(s)$ for optimal policy

$$\underbrace{V(s) = \max_a \sum_{s'} P(s'|a,s)\left(R'(s,a,s) + \gamma V(s')\right)}_{\text{problem: V defined in terms of itself}}$$

<u>Solution:</u> Add time component $V(s,t)$

$V_t(s)$

go up instead of going down

$$\begin{cases} V_0(s) = 0 \\ V_{t+1}(s) = \max_a \sum_{s'} P(s'|a,s)\left(R(s',a,s) + \gamma V_t(s')\right) \end{cases}$$

<u>Variant:</u> Q-iteration

$V(s)$: optimal exp. reward from s

$Q(s,a)$: optimal exp. reward from s, a

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

$$Q(s,a) = \sum_{s'} P(s'|s,a)\left(R(s',a,s) + \gamma V(s')\right)$$

$$= \sum_{s'} P(s'|s,a)\left(R(s',a,s) + \gamma \underline{\max_{a'}} Q(s',a')\right)$$

Same algo as before ↓

- $Q_0(s,a) = 0$
- $Q_{t+1}(s,a) = \sum_{s'} \cdots \left(\cdots \max_{a'} Q_t(s',a') \right)$

What if $\underbrace{P(s'|a,s)}_{\text{come from world}}$ unknown?

→ can still estimate $Q(s,a)$ from data

→ called Q-learning

$\underbrace{s_0, a_0, s_1, a_1, s_2, \ldots}_{\text{trajectories}}$

(come from policy π)

$\pi : S \to A$

Trajectory: $s_0, a_0, s_1, a_1, s_2, a_2, \ldots$

After each state-action (s_t, a_t), update:

$$\underbrace{Q(s_t, a_t)}_{\text{new value}} = (1-\alpha) \underbrace{Q_{old}(s_t, a_t)}_{\text{old value}} + \alpha \cdot \left(\underbrace{r_t}_{R(s_{t+1}, a_t, s_t)} + \gamma \cdot \max_{a'} Q_{old}(s_{t+1}, a') \right)$$

expected value of Right-hand term:

$$\mathbb{E}_{s_{t+1}}\left[R(s_{t+1}, a_t, s_t) + \gamma \max_{a'} Q_{old}(s_{t+1}, a')\right]$$

$$= \sum_{s'} P(s' \mid a_t, s_t)\left(R(s', a_t, s_t) + \gamma \max_{a'} Q_{old}(s', a')\right)$$

exactly what we had for Q-iteration

Convergence theorems

- require $\alpha \to 0$ "step size" | in practice, fix some small step size
- other caveat: need to explore
    - analogy to multi-armed bandits: need to visit all states sufficiently often

- exploration policy
- induced policy from $Q(s,a)$

$\hookrightarrow$ $Q_0(s,a) = 0$ rewards: all > 0

each update: $Q(s_t, a_t) > 0$

Exploration

→ some fraction of time, take random action

→ initialize $Q_0(s,a) =$ some large value
(optimistic value)

↳ similar idea to UCB

Large state spaces.

- Grid world: ~20 states
- Starcraft, DotA
  ↳ 200 units $\approx 10^5$ positions
  $$(10^5)^{200} = 10^{1000}$$
- Each Q-learning update:
  - Only updates $Q(s,a)$ for single (s,a)
- Idea: Function approximation

$$Q_\theta(s,a) \leftarrow \text{parameterization of possible } Q \text{ functions}$$

Old update:

$$Q(s_t,a_t) = (1-\alpha)\, Q_{old}(s_t,a_t) + \alpha\left(r_t + \max_{a'} Q_{old}(s_{t+1},a')\right)$$
$\uparrow$

$$| = Q_{old}(s_t,a_t) + \alpha\Big(\underbrace{r_t + \max_{a'} Q_{old}(s_{t+1},a')}_{\text{target new value}} - \underbrace{Q_{old}(s_t,a_t)}_{\text{old value}}\Big)$$

New update:

$$\theta = \theta_{old} + \alpha\Big(\underline{r_t + \max_{a'} Q_{\theta_{old}}(s_{t+1},a')} - \underline{Q_{\theta_{old}}(s_t,a_t)}\Big) \cdot \underbrace{\nabla_\theta Q_{\theta_{old}}(s_t,a_t)}_{}$$

1 in $s_t a_t$
0 everywhere else

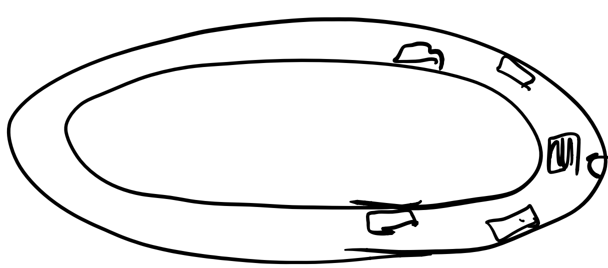
Special case: one-hot encoding

$\theta_{s,a}$ for each state-action

$Q(s,a) = \theta_{s,a}$

Environment

↳ racecar



- our car → x_0
- other cars ← $x_1, \ldots, x_K$
- gas remaining ← g
- time left ← t

$$s = (x_0, x_1, \ldots, x_K, g, t)$$

features we should care about:

- velocity, heading
- distance to closest car
- angle closest car
- gas
- time
- ⋮

features

$$Q_\theta(s,a) = \theta_1 (\text{velocity}) + \theta_2 \cdot (\text{distance}) + \theta_3 \cdot (\text{angle}) + \cdots$$

old

New update:

$$\theta = \theta_{old} + \alpha\left(r_t + \max_{a'} Q_{\theta_{old}}(s_{t+1}, a') - Q_{\theta_{old}}(s_t, a_t)\right) \cdot \nabla_\theta Q_{\theta_{old}}(s_t, a_t)$$

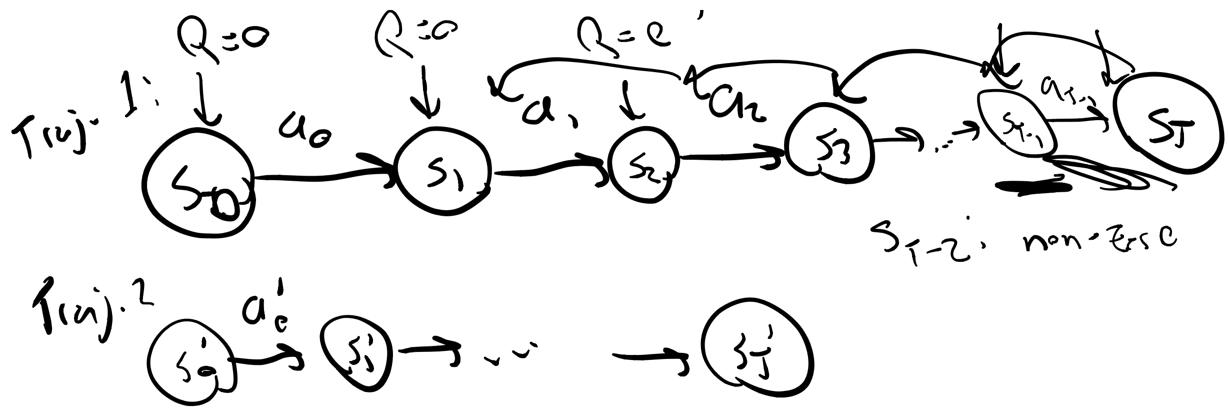
"Intuitive derivation"

Prediction task: ~~[illegible]~~ less $(\text{target} - \text{value})^2$

$$+2 \cdot \nabla_\theta\left(\left(r_t + \max_{a'} \underbrace{Q_{\theta_{old}}}_{\text{fixed}}(s_{t+1}, a') - \underbrace{Q_{\theta_{old}}}_{\text{change-able}}(s_t, a_t)\right)^2\right)$$

Temporal difference learning

$TD(\lambda)$ $Q(s,a) = \theta$ Q non-zeese



Often only see reward at end
"credit assignment" problem

regular Q-learning:
need T updates for trajectory of length T

TD(λ): propagate rewards backward
↳ TD-Gammon

Policy gradient

- Q-learning $Q_\theta(s,a)$
- directly learn policy $\pi_\theta(a|s)$

probability dist. of action | state

$$\max_\theta \mathbb{E}_{\pi_\theta}\left[R(s_0,a_0,s_1)+\gamma R(s_1,a_1,s_2)+\dots\right]$$

wrt gradient

log-derivative trick

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[R]$$

$$= \mathbb{E}_{\pi_\theta}\left[R \cdot \underbrace{\nabla_\theta \log \pi_\theta}_{\frac{1}{\pi_\theta} \nabla \pi_\theta}\right]$$

cancel

Logistic regression

$(x_1, y_1), \ldots, (x_n, y_n)$

Stochastic GD:

(x_i, y_i)

labels   features

$$\theta = \theta_{old} + \alpha \cdot \nabla_\theta \underbrace{\log(1 + \exp(-y_i \theta^T \varphi(x_i)))}_{\text{logistic loss}}$$

θ_{old}

model parameters

$Q(s,a)$: expected reward for taking action a in state s, then following optimal policy

approximate this by current guess based on $\max_{a'} Q(s',a')$ in new state

---

Small State space case

EV over s'

$$Q(s,a) \approx \sum_{s'} P(s'|s,a)\left(\overbrace{R(s',a,s)}^{\text{immediate reward}} + \overbrace{\gamma \max_{a'} Q(s',a')}^{\text{looking ahead}}\right)$$

samples

100 times where $s_t = s$, $a_t = a$

$s_{t+1}^{(1)}, \dots, s_{t+1}^{(100)}$

$$\frac{1}{100}\sum_{i=1}^{100}\left(R(s_{t+1}^{(i)}, a_t, s_t) + \gamma \dots\right)$$

hoping that $Q(s',a') \approx$

$$\sum_{s'} P(s'|s,a)\left(R(s',a,s) + \gamma \max_{a'} \sum_{s''} P(s''|s',a') \cdot \left(R(s'',a',s') + \dots\right.\right.$$

$\alpha = 0.01$

100 samples

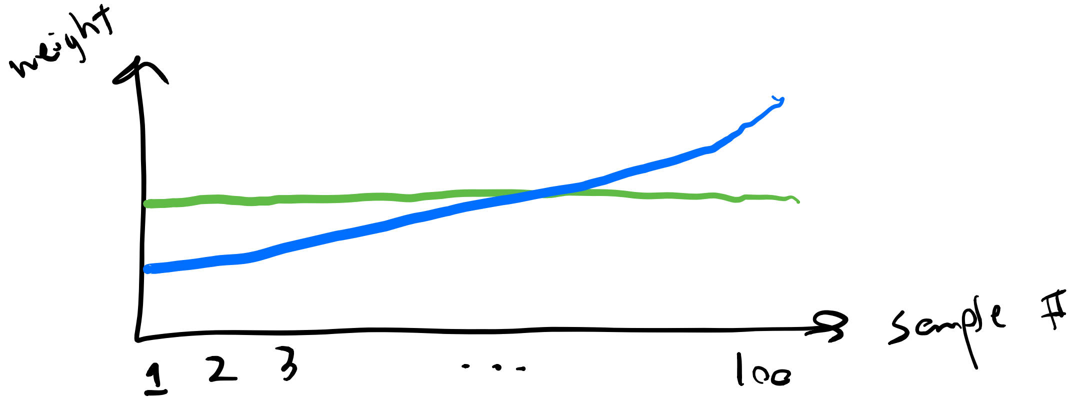
sample: weight $\alpha = 0.01$

Most recent sample: ...

2^{nd} most recent sample: weight $\alpha \cdot (1-\alpha) = 0.01 \cdot 0.99$

3^{rd} most recent: $\alpha \cdot (1-\alpha)^2$

1^{st} sample: $\alpha \cdot (1-\alpha)^{99}$



$$Q(s,a) \approx \sum_{s'} P(s'|a,s) \cdot \left(R(s',a,s) + \gamma \cdot \max_{a'} Q(s',a')\right)$$

$$Q_\theta(s,a) \approx \sum_{s'} P(s'|a,s) \cdot \left(R(s',a,s) + \gamma \max_{a'} Q_\theta(s',a')\right)$$

Find θ such that $\approx$ true

$$\underbrace{\left[\sum_{s'} P(s'|a,s)\left(R(s',a,s) + \gamma \max_{a'} Q_{\theta_{old}}(s',a')\right)\right]}_{\text{target}} - \underbrace{Q_{\theta_{old}}(s,a)}_{\text{prediction}}$$

next time step t

current time $t-1$

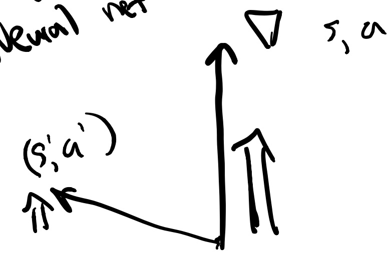
direction:

$(\text{target} - \text{prediction}) \cdot \nabla_\theta Q_{\theta_{old}}(s,a)$

$\theta_{old}(s,a)$

Neural net

$-\gamma \nabla_\theta Q_{\theta_{old}}(s',a')$



$$\theta = \theta_{old} + \alpha \cdot (\text{target} - \underline{\text{prediction}}) \cdot \nabla_\theta Q_{\theta_{old}}(s,a)$$

$Q_{\theta_{old}}(s,a)$

$\sum_{s'} P(s'|a,s) \cdot (\ldots)$

replace w/ sample

$r_t + \max_{a'} Q_{\theta_{old}}(s_{t+1}, a')$

$R(s_{t+1}, a_t, s_t)$

// s_{t+1} is a sample from $P(s'|a_t, s_t)$