Problem Setting

batch set of data D, consisting on N trajectories of length H

$s_{i1}, a_{i1}, r_{i1}, s_{i2}, a_{i2}, r_{i2} \ldots$

↑ time ↑ index into traj

gen by behavior π_b (could be nonstationary, unknown, multiple)

S is very large or continuous, may be represented as a vector of feature values

goal: estimate optimal state action values* relates to 1st main challenge

Why is this an interesting, important setting?

Why batch
- in many real world applications (customer marketing, electronic health records, intelligent tutoring systems) have access to prior data about seq of decisions and their outcomes
- but may be hard/costly to gather more data
- would like to use existing data to inform future decision making
- Why large/factored describe S? realistic
- Why estimate Q?

sufficient for making good decisions

more precisely, if compute a close estimate to V^*/Q^* then greedy policy wrt that estimate will have close to V^* performance in real world (at least in discrete S, A)

thm: (Satinder & Yee 1994). Let V^* be the optimal value func for a discrete-time MDP having finite state and action sets and an ∞ horizon w/geometric discounting $\gamma \in [0,1)$. If $\check{V}$ is a function s.t. $\forall s \in S$

$$|V^*(s) - \check{V}(s)| \leq \epsilon$$

and the greedy policy $\pi_{\check{V}}$ is defined as

relate to Q

$$\pi_{\check{V}}(s) = \arg\max_{a \in A(s)} r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)\check{V}(s')$$

then for $\forall s \in S$

define what $V^{\pi_{\check{V}}}$ means

$$V^*(s) - V^{\pi_{\check{V}}}(s) \leq 2\gamma\epsilon/(1-\gamma)$$

proof:

for shorthand define loss $L_{\check{V}}(s) = V^*(s) - V^{\pi_{\check{V}}}(s)$

$\exists$ a state z that achieves the max loss. Call this state z

$\forall s \in S$, $L_{\check{V}}(z) \geq L_{\check{V}}(s)$

for state z consider an optimal action $a = \pi^*(z)$ and the action specified by $\pi_{\check{V}}$, $b = \pi_{\check{V}}(z)$.

because $\pi_{\check{V}}$ is the greedy π for $\check{V}$, b must look at least as good as a

$$r(z,a) + \gamma \sum_{s' \in S} p(s'|z,a)\check{V}(s') \leq r(z,b) + \gamma \sum_{s' \in S} p(s'|z,a)\check{V}(s'$$

proof cont

because $\forall s' \in S \quad V^*(s') - \epsilon \le \breve{V}(s') \le V^*(s') + \epsilon$ (by assump)

$$r(z,a) + \gamma \sum_{s' \in S} p(s'|z,a)(V^*(s') - \epsilon)) \le r(z,b) + \gamma \sum_{s'} p(s'|z,b)(V^*(s') + \epsilon)$$

$$(*) \quad r(z,a) - r(z,b) \le 2\gamma\epsilon + \gamma \sum_{s'} p(s'|z,b)V^*(s') - p(s'|z,a)V^*(s')$$

the loss at z is

$$L_{\breve{V}}(z) = V^*(z) - V^{\pi_{\breve{V}}}(z)$$

$$(**) \quad = r(z,a) - r(z,b) + \gamma \sum_{s'} p(s'|z,a)V^*(s') - p(s'|z,b)V^{\pi_{\breve{V}}}(s')$$

substitute (*) into (**)

$$L_{\breve{V}}(z) \le 2\gamma\epsilon + \gamma \sum_{s'} p(s'|z,b)V^*(s') - p(s'|z,b)V^{\pi_{\breve{V}}}(s')$$

(the a terms cancel)

$$= 2\gamma\epsilon + \gamma \sum_{s'} p(s'|z,b)\left[V^*(s') - V^{\pi_{\breve{V}}}(s')\right]$$

$$= 2\gamma\epsilon + \gamma \sum_{s'} p(s'|z,b) L_{\breve{V}}(s') \quad \text{defn loss}$$

$$\le 2\gamma\epsilon + \gamma \sum_{s'} p(s'|z,b) L_{\breve{V}}(z) \quad \text{max loss} \ge \text{other loss}$$

$$= 2\gamma\epsilon + \gamma L_{\breve{V}}(z) \quad \text{pull } L_{\breve{V}}(z) \text{ out, indep of } s'$$

$$L_{\breve{V}}(z) \le 2\gamma\epsilon/(1-\gamma) \quad \text{re-arrange} \qquad \square$$

- if we only have data, how get $\breve{V}$? No model parameters to do planning /dynamic programming ...
- how could we estimate V of behavior policy(ies) used to generate D?

Monte carlo estimation, model free, uses complete episodes, e.g. just look at actual (discounted) sum of returns from trajectories

sample to approx expectation

$$s_{i1}, a_{i1}, r_{i1}, s_{i2}, \ldots r_{iT} \longrightarrow \sum_{k=1}^{T} \gamma^{k-1} r_{ik}$$

$$s_{i+1,1}, a_{i+1,1}, r_{i+1,1}, s_{i+1,2} \ldots r_{i+1,T} \longrightarrow \sum_{k=1}^{T} \gamma^{k-1} r_{i+1,k}$$

average over return from each traj is an estimation of $V^{\pi_b}(s_{i1})$ (assuming all traj start from same starting s)

(empirical mean return instead of expected return)

one justification: average converges to expectation

- Chernoff-Hoeffding inequality

  let $X_1 \ldots X_n$ be indep random var

  assume X_i a.s. bounded, $P(X_i \in [a_i, b_i]) = 1 \quad 1 \le i \le n$

  then $P\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - E[X]\right| \ge \epsilon\right) \le 2\exp\left(\frac{-2n^2\epsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right)$

  so can have finite sample guarantees on estimate quality

- every visit MC evaluation: every time s visited in an episode, incremental
- but using MC estimation above only yields estimate of V^{π_b}

  what if we want to know $Q^*/V^*/\pi^*$?

need off policy learning

learn about one policy π by potentially executing another

(also relevant to value function)

to do this, 1st briefly review temporal difference (TD) learning

TD methods

model free

learn from incomplete trajectories by <u>bootstrapping</u> ← updating V involves an estimate

MC vs TD

incremental every visit MC. G_t = actual episode return starting at s_t

$$V(s_t) = V(s_t) + \alpha(G_t - V(s_t))$$

TD(0): update $V(s_t)$ towards estimated return $r_{t+1} + \gamma V(s_{t+1})$

$$V(s_t) = V(s_t) + \alpha\underbrace{(\underbrace{r_{s_t} + \gamma V(s_{t+1})}_{\text{TD target}} - V(s_t))}_{\text{TD error}}$$

both TD & MC approximate expectation of future rewards by averaging over samples

in online learning TD has some benefit over MC because don't have to wait until the end of the episode to learn

bias/variance

$G_t = r_{t+1} + \gamma r_{t+2} + \dots$ = unbiased estimate of $V^\pi(s_t)$

true TD target $r_{t+1} + \gamma V^\pi(s_{t+1})$ is an unbiased estimate of $V^\pi(s_t)$

TD target $r_{t+1} + \gamma V(s_{t+1})$ is a biased estimate because $V(s_{t+1}) \simeq V^\pi(s_t)$ (and may have large error w/little data or early in computation)

but G_t /return is much higher variance than TD target

MC high var, 0 bias

TD(0) low var, some bias

in batch data setting for discrete state & actions

MC converges to soln w/min MSE (best fit to observed returns)

$$\sum_{d=1}^{D} \sum_{t=1}^{H_d} (G_t^d - V(s_t^d))^2$$

TD(0) converges to max likelihood Markov model

[*] for discrete s-a

TD exploits Markov property
MC does not assume nor exploit Markov property $\Big\}$ → $\boxed{\text{ask about}}$

○ Q-learning

like TD(0) but estimates state-action value instead of state value (useful for extracting a policy)

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(\underbrace{r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')}_{} - Q(s_t, a_t))$$

converges to optimal Q^* [*]

- approximate dynamic programming

  2 challenges / sources of error

  1) in our assumed setting, state space S is large or ∞ intractable or impossible in general to represent V exactly
  $\Rightarrow$ instead use function (parametric or nonparametric) ← give ex of each
  to approximate the Q and/or V function
  yields approximation error - why?

  2) estimation error. If reward model and dynamics / transition model are unknown, cannot exactly compute a Bellman backup (an equivalent way to say this is that cannot exactly compute the Bellman operators $\mathcal{T}$ and $\mathcal{T}^{\pi}$) ← see earlier def
  $\Rightarrow$ instead estimate the Bellman operator from samples

- Popular approach: Fitted Q Iteration (FQI) ← why useful to compute Q instead of V?

  Gordon 1999: fitted value iteration
  Ormoneit & Sen 2002
  Ernst et al. 2005 FQI

key idea: pose Q-function determination problem as a sequence of regression problems

recall value iteration in tabular domains

$$Q_{k+1}(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} Q_k(s',a')$$

$$Q_1(s,a) = r(s,a)$$

also can express backup as $\mathcal{T} Q_k$ where $\mathcal{T}$ is Bellman operator

two ideas. Assume start w/some rep of Q_k

1) Approximate backup / expectation using samples & prior estimate of Q_k
2) fit function approximator (representing Q_{k+1}) to those samples

FQI alg

input: set of N tuples (s, a, r, s') $\mathcal{F}$ and a regression algorithm ← note: we can get these from traj
where $(s_i, a_i) \sim \rho$ and $s' \sim p(\cdot|s_i, a_i)$ where ρ sampling distrib over $S \times A$

$k=0$, define $\hat{Q}_k$ to be a func equal to 0 everywhere on $S \times A$

while stopping condition not reached

- $k = k+1$
- construct new dataset $D'_k = \{(s_i, a_i), y_i\}_{i=1}^{N}$ where

$$\left.\begin{aligned} y_i &= r_i + \gamma \max_{a'} \hat{Q}_{k-1}(s_i', a') \\ &\equiv \hat{\mathcal{T}} Q_k(s_i, a_i) \end{aligned}\right\} \text{can view as constructing bootstrapped samples of } Q$$

- Use regression algorithm to use D'_k to fit $\hat{Q}_k$

$\hat{Q}_k := \text{fit}(f, F, D'_k)$ (find best regression func $f \in F$ that fits data) essentially have a

target function $= \mathcal{T} Q_K$

noisy observ $= \hat{\mathcal{T}} Q_K$ at a finite set of points

minimize empirical error in the fitting process

$\hat{Q}_K = \underbrace{\min_{f \in \mathcal{F}}}_{\text{regressor class}} \| f - \hat{\mathcal{T}} \hat{Q}_{K-1} \|_{\hat{\rho}}$, $\hat{\rho}$ = empirical measure of ρ given N samples

with the goal of minimizing the true error

$Q_K = \min_{f \in F} \| f - \mathcal{T} \hat{Q}_{K-1} \|_\rho$

regression problem of $\hat{Q}_K \approx \mathcal{T} \hat{Q}_{K-1}$

objective is $\| \hat{Q}_K - \mathcal{T} \hat{Q}_{K-1} \| = \underbrace{\| \hat{Q}_K - Q_K \|_\rho}_{\text{estim error}} + \underbrace{\| Q_K - \mathcal{T} \hat{Q}_{K-1} \|_\rho}_{\text{approx error}}$] ★ important idea

doesn't address distance to true Q w/no func approx

- what should the error function be?
- what should the regression function be?
- does it matter
    - for desirable properties
    - for performance
- what if ρ (sampling distrib) is biased? how could this affect things?

example: least squares value iteration (from Tsitsiklis & Van Roy 1996)

choose sum of squared errors as error function

compute parameters to find least squares fit

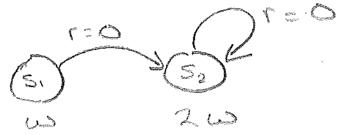
let ω be the parameters that define a function f

then $\hat{Q}_K$ = fit (f, F, D'_K) becomes

$\omega_K = \operatorname{argmin}_\omega \sum_{i=1} (\hat{Q}_{K,\omega_K}(s_i, a_i) - \hat{\mathcal{T}} \hat{Q}_{K-1,\omega_{K-1}}(s_i, a_i))^2$

how good is least-squares value/q iteration?

ex. (from Tsitsiklis & Van Roy)



2 states

1 action

$p(s_2 | s_1, a_1) = 1$

all rewards 0

$p(s_2 | s_2, a_1) = 1$

$Q^*(s_1, a_1) = V^*(s_1) = V^*(s_2) = Q^*(s_2, a_1) = 0$

define a feature ϕ over state space s.t. $\phi(s_1) = 1$ $\phi(s_2) = 2$

use a linear function approximator of the (state/action) value

$\hat{Q}_{K,\omega_K}(s) = \omega_K \phi(s)$ where ω_K is a scalar

1) can we represent optimal value func w/this approx?

2) assume a discount factor γ

imagine 2 data tuples: s_1 a_1 0 s_2

s_2 a_1 0 s_2

if sample [illegible]

$\omega_{K+1} = \operatorname{argmin}_\omega \sum_{i=1}^{2} (\omega \phi(s_i) - [\mathcal{T} \omega \phi](s_i))^2$

define $\mathcal{T} \omega \phi$

solve for ω

3) how does ω change as k gets larger?

$$\omega_{k+1} = \text{argmin}_{\omega} \sum_{i=1}^{2} (\omega\phi(s_i) - [T\omega\phi](s_i))^2$$

$$= (\omega - (0 + \gamma \cdot 2\omega_k))^2 + (2\omega - (0 + \gamma 2\omega_k))^2$$

$$= \omega^2 - 4\omega_k \gamma \omega + \gamma^2 4\omega_k^2 + 4\omega^2 - 8\omega\omega_k\gamma + 4\gamma^2\omega_k^2$$

$$d/d\omega \rightarrow 10\omega - 4\omega_k\gamma - 8\omega_k\gamma = 0$$

$$10\omega = 12\omega_k\gamma$$

$$\omega = 6/5\,\omega_k\gamma$$

$$w_{k+1} = \arg\min_w \left[\underbrace{(w - \gamma 2 w_k)^2}_{\text{from } s_1} + \underbrace{(2w - \gamma 2 w_k)^2}_{\text{from } s_2} \right]$$

d/dw & set to 0

$w_{k+1} = 6/5 \gamma w_k$

if $\gamma > 5/6$ then $w_{k+1} \to \pm\infty$ if $w \neq 0$

diverges!

Why does this occur?

Bellman backup is a contraction

but projecting back to func approximator class may not be

see Geoff Gordon

- what function approximators do guarentee convergence of FQI?
  averagers / Kernels / nearest neighbors

- even if not guaranteed to converge, can do well

- guarentees if can bound error per iteration of FVI (similar to FQI) Mohammad

assume we can approximate Bellman optimality operator up to ϵ at iteration k

$\hat{V}_{k+1} = \hat{T} V_k$ s.t. $\hat{V}_{k+1} = T V_k + \epsilon_k$ or $\| \hat{V}_{k+1} - T V_k \|_\infty = \epsilon_k$

--> Mohammad

also exist work on finite sample bounds

- summary
  - FVI/FQI popular and powerful algorithm
  - performing/solving supervised learning problem at each iteration
  - due to iterations, error can compound (worse than linear) across iterations
  - input data sampling distrib ρ has a crucial influence on algorithm performance. (Equivalent to role of exploration)
  - poor performance (divergence) is possible: choice of error function & regressor important
    - issue also of capacity of regressor vs true function vs available data - to be discussed more!
  - falls into class of model free value function based techniques