Off-Policy Methods with Approximation

- much harder than in tabular case

+ do not converge robustly like on-policy cases

× explore convergence problems

× explore learnability

× build new algos

* never as strong theoretically nor empirically as on-policy

* approximation methods are powerful but require care

- basic issue: target π & behavior b

+ prediction, $\hat{q} \approx q_*$, $\hat{v} \approx v_*$

control learn best a, changing π, maybe b

- 2 part problem, divvy by solution

+ tabular: target of update

+ func approx: distribution of update

× target update extension is easy

× update distribution is hard

Semi-gradient Methods

- how to change update to target policy in func approx
+ may diverge, but are often still successfully used
x tabular is the special case of approx & guarantees stability
x may still be able to use good feature selection to produce stable results
- tabular sampling ratio

$$\rho_t \doteq \rho_{t:t} = \frac{\pi(A_t | S_t)}{b(A_t | S_t)}$$

- off-policy semigradient TD(0) is just

$$w_{t+1} \doteq w_t + \alpha \rho_t \delta_t \nabla \hat{v}(s, w_t)$$

- again, δ_t is defined differently
+ for episodic:

$$\delta_t \doteq R_{t+1} + \gamma \hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t)$$

+ for continuing

$$\delta_t \doteq R_{t+1} - \bar{R}_t + \hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t)$$

- for Expected Sarsa, no sampling ratio because it is "on-policy" in tabular
+ in approx, multiple pairs can relate to the same weight, confusing how the ratio would be applicable
- n-step version of semi-gradient Sarsa

$$w_{t+n} \doteq w_{t+n-1} + \alpha \rho_{t+1} \cdots \rho_{t+n} [G_{t:t+n} - \hat{q}(S_t, A_t, w_{t+n-1})$$

$$] \cdot \nabla \hat{q}(S_t, A_t, w_{t+n-1})$$

EXERCISE 11.1

$$w_{t+n} = w_t + \alpha \rho_{t:t+n-1} [G_{t:t+n} - \hat{v}(S_t, w_t$$

$$] \nabla \hat{v}(S_t, w_{t+n-1})$$

episodic

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \ldots \gamma^{n-1} R_{t+n-1} + \gamma^n$$

$$\hat{v}(S_{t+n}, w_{t+n-1})$$

$$= \sum_{k=0}^{n-1} \gamma^k R_{k+t+n} + \gamma \hat{v}(S_{t+n}, w_{t+n-1})$$

Continuous

$$G_{t:t+n} = R_{t+1} - \bar{R}_t + R_{t+2} - \bar{R}_{t+1} + \dots +$$

EXERCISE 11.2 n-step Q(σ)

$$w_{t+n} = w_{t+n-1} + \alpha \left[G_{t:t+n} - \hat{q}(S_t, A_t, w_{t+n-1}) \right] \nabla \hat{q}(S_t, A_t, w_{t+n-1})$$

episodic

$$G_{t:h} \doteq R_{t+1} + \gamma(\sigma_{t+1}\rho_{t+1} + (1-\sigma_{t+1})\pi(A_{t+1} \mid S_{t+1}))(G_{t+1:h} - \hat{q}(S_{t+1}, A_{t+1}, w_{h-1})$$

$h = t+n$

ρ_{t+1}

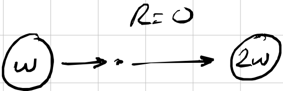
Continuous

$$G_{t:h} = R_{t+1} - (\overset{\curvearrowleft}{\sigma_{t+1}} \bar{R}_t + (1-\sigma_{t+1})\pi(A_{t+1} \mid$$

$S_{t+1})) \left(G_{t+1:n} - \hat{q}(S_{t+1}, A_{t+1}, w_{n-1}) \right)$

Examples of Off-Policy Divergence

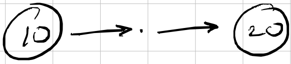
- start with examples of how semigradient & other simple off-policy algos diverge
- consider



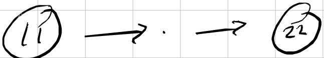
$\mathbf{w} = [w]$

weight vector of one weight

+ if $w = 10$



+ if $\gamma \approx 1$, $\delta \approx 10$
+ if $\alpha = 0.1$ $w + 1$, $w = 11$
+ now



× $\delta \approx 11$, has increased now, increasing $w \approx 12.1$

* obvious divergence to ∞

- in math:

$$\delta_t = R_{t+1} + \gamma \hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t)$$

$$= 0 + \gamma 2 w_t - w_t = (2\gamma - 1) w_t$$

+ off-policy:

$$w_{t+1} = w_t + \alpha \rho_t \delta_t \nabla \hat{v}(S_t, w_t)$$

$$= w_t + \alpha \cdot 1 \cdot (2\gamma - 1) w_t \cdot 1$$

$$= (1 + \alpha(2\gamma - 1)) w_t$$

- update is now old value times scalar

+ > 1, diverge

× here, happens because $\rho = 1$ since no other transitions possible

* thus, only possible to train this in an offpolicy way where $\rho = 0$

- full Example: Baird's counterexample

+ looks well suited for func approx. & convergence.

× actually diverges quickly simply in off-policy behavior

- Q-learning still diverges too, even though it has the strongest guarantees for convergence

+ practically, keeping $b \approx \pi$ (ε-greedy)

The Deadly Triad

- divergence / instability come from

1) Function Approximation = generalize state space for computation / memory
2) Bootstrapping = updates using existing estimates rather than full experience and returns for speeding up learning
3) Off policy training = using a distribution of transitions diff from target policy

- not just about control; prediction gets wonky
- not about env; perfect DP goes awry
- if < 3, all good
- What can we give up?

+ not func approx
+ not "bootstrapping"

× Speed, cost saving, actual learning too good

- off policy? only essential in some cases

+ psychology of learning from sensory events, not just rewards
+ an event may be neutral in reward but unexpected in planning, offering new intel for how to behave

× off-policy allows for parallel processing and Interpretation even with one experience Stream

Linear Value Function Geometry

- to understand challenge better, abstract value function thinking

+ for all value functions $v: S \rightarrow \mathbb{R}$ few represent any policy

x most cannot be represented by a function approximation, which has fewer params than states

- suppose $S = \{ s_1, s_2, \dots, s_{|S|} \}$
  & feature vector $[v(s_1), v(s_2), \dots, v(s_{|S|})]^T$

+ almost impossible, but conceptually good

x case $S = \{ s_1, s_2, s_3 \}$ $w = (w_1, w_2)^T$

* value function is in 3D space, but parameters are 2D, map to 3D

* linear approximation creates a simple planar subspace

- take fixed π, whose true value func cannot be represented

- goal: figure out what value function is closest to π's

+ first, distance measure

x $v_1 - v_2 = v$ some difference vector

x size

$$\|v\|_\mu^2 \doteq \sum_{s \in S} \mu(s) \, v(s)^2$$

* Euclidean dist doesn't work because we need distribution of states to matter
- finding closest value func approx to value func is approximation problem

+

$$\Pi v \doteq v_w \quad w = \underset{w \in \mathbb{R}^d}{\arg\min} \| v - v_w \|^2_\mu$$

here Π is projection operator

+ hence Πv_π is the projection of the true function v_π to closest representable function

x MC methods approach this approx really slowly

- TD methods find diff. solution based on Bellman equation

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

- v_π solves this exactly.

+ any other v_w gives an error called Bellman error

$$\bar{\delta}_w(s) \doteq \left(\sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v_w(s')] \right)$$

$- v_w(s)$

$= \mathbb{E}_\pi \left[R_{t+1} + \gamma v_w(S_{t+1}) - v_w(S_t) \mid S_t = s, A_t \sim \pi \right]$

- $\bar{\delta}_w \in \mathbb{R}^{|S|}$ is the Bellman error vector

+

$\overline{BE}(w) = \| \delta_w \|^2$ mean BE

- There exists a unique w for which $\overline{BE}$ is minimized
+ will be diff from $\overline{VE}$
- One gets BE by applying the Bellman operator $B_\pi : \mathbb{R}^{|S|} \rightarrow \mathbb{R}^{|S|}$ to approx value func

$$(B_\pi v)(s) \doteq \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \cdot \left[r + \gamma v(s') \right]$$

+ for v_w, $\bar{\delta}_w = B_\pi v_w - v_w$
+ if B_π is applied to points outside the representable space, eventually reach

$v_\pi = B_\pi v_\pi$, the fixed point

x requires DP (no func approximation)

- $\Pi \bar{\delta}_w$ is BE projected back in sample space

+

$$\overline{PBE}(w) = \| \Pi \bar{\delta}_w \|^2_\mu$$

size of error vec in represented space

x Zero error → w_{TD} fixed point

* not always stable in off-policy

* diff from min for $\overline{VE}$ / $\overline{BE}$

Gradient Descent in the Bellman Error

- off policy stability using SGD, one more look

- SGD stable when updates are made to

downward in expectation w.r.t. objective

+ only MC methods are true SGD so far

× converge for off-policy, on-policy, non linear func. approx.

- Semi-SGD diverges in off-policy cases & in simple nonlinear func. approximators.
- FOR REFERENCE, explore Bellman error as new obj function, and why it fails
- first consider naive expected TDE

$$\overline{TDE}(w) = \mathbb{E}_b\left[\rho_t \delta_t^2\right]$$

+ Sample update from TDE with gradient

$$w_{t+1} = w_t - \frac{1}{2}\alpha\nabla(\rho_t\delta_t^2)$$

$$= w_t - \alpha\frac{1}{2}\rho_t\delta_t\nabla\delta_t$$

$$= w_t + \alpha\rho_t\delta_t\left(\nabla\hat{v}(S_t, w_t) - \gamma\nabla\hat{v}(S_{t+1}, w_t)\right)$$

- now full gradient with

+ naive-residual gradient

× naive because on trivial A-split example, simply does not work.

- so now use full expected error, not sample

$$w_{t+1} = w_t - \frac{1}{2} \alpha \nabla \left(\mathbb{E}_\pi[\delta_t] \right)^2$$

. . .

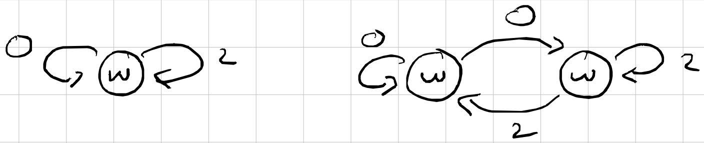
$$w_{t+1} = w_t + \alpha \left[\mathbb{E}_b \left[\rho_t \left(R_{t+1} + \gamma \hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t) \right] \right] \left[\nabla \hat{v}(S_t, w) - \gamma \mathbb{E}_b \left[\rho_t \nabla \hat{v}(S_{t+1}, w) \right) \right] \right]$$

- residual-gradient algorithm
+ requires two independent samples to not introduce bias
- robust, on all types, in convergence
+ however, it is slow, still converges to wrong values, and is not learnable

The Bellman Error is Not

Learnable

- not just "learnable in polynomial time" but "learnable" at all!
- quantities that just cannot be computed given only knowledge of observable events



- indistinguishable from just rewards (0, 0, 2, 0, 2, 2, 2, 0 ... random looking seq)

+ cannot tell anything about it.

- not just that, but $\overline{VE}$ isn't unique from reward dist

⊥ if it can't be learned, how is $\overline{VE}$ useful???

x a way out: if the same parameter vector optimizes all indistinguishable $\overline{VE}$, then knowing $\overline{VE}$ is unnecessary

* just focus on optimizing the black box

- Mean square return error $\overline{RE}$, onpolicy:

$$\overline{RE}(w) = \mathbb{E}\left[(G_t - \hat{v}(S_t, w))^2\right]$$

$$= \overline{VE}(w) + \mathbb{E}\left[(G_t - v_\pi(S_t))^2\right]$$

+ $\overline{RE}$ is same as $\overline{VE}$ with some indipendent variance term
- $\overline{BE}$ simply does not follow the same rules: it's min cannot be computed
+ without model knowledge

Gradient TD Methods

- $\Pi \doteq X(X^T D X)^{-1} X^T D$

+ D is an $|S| \times |S|$ diag matrix with $\mu(s)$ on diag
+ X is $|S| \times d$ matrix rows are feature vecs $x(s)^T$, one for each state

- $\|v\|_\mu^2 = v^T D v$
- $v_w = Xw$

- focus on SGD at cost of computational complexity for convergence

$$\overline{PBE}(w) = \|\Pi \bar{\delta}_w\|^2_\mu$$

$$= \|\Pi \bar{\delta}_w\|^T D \Pi \bar{\delta}_w$$

$$= \bar{\delta}_w^T \Pi^T D \Pi \bar{\delta}_w$$

$$= \bar{\delta}_w^T D X (X^T D X)^{-1} X^T D \bar{\delta}_w$$

$$= (X^T D \delta_w)^T (X^T D X)^{-1} (X^T D \bar{\delta}_w)$$

- w.r.t. w, ∇ becomes

$$\nabla \overline{PBE}_w = 2 \nabla [X^T D \bar{\delta}_w]^T (X^T D X)^{-1} (X^T D \bar{\delta}_w)$$

- to become SGD must sample something on every timestep with this expectation

+ suppose μ is behavior policy distribution of S

× rewrite each factor in gradient as expectation in μ

$$X^T D \bar{\delta}_w = \sum_s \mu(s)\, x(s)\, \delta_w = \mathbb{E}[\rho_t \delta_t x_t]$$

- next, transpose

$$\nabla \mathbb{E}[\rho_t \delta_t x_t]^T = \mathbb{E}[\rho_t \nabla \delta_t^T x_t^T]$$

$$= \mathbb{E}[\rho_t \nabla (R_{t+1} - \gamma w^T x_{t+1} - w^T x_t)^T x_t^T]$$

$$= \mathbb{E}[\rho_t (\gamma x_{t+1} - x_t) x_t^T]$$

- middle factor

$$X^T D X = \sum_s \mu(s) x(s) x(s)^T = \mathbb{E}[x_t x_t^T]$$

- final form

$$\nabla \overline{PBE}(w) = 2\mathbb{E}[\rho_t (\gamma x_{t+1} - x_t) x_t^T] \, \mathbb{E}[x_t x_t^T]^{-1} \cdot$$

$$\mathbb{E}[\rho_t \delta_t x_t]$$

- if computed naively, bias remains in first & last terms
- could store estimates of last two, compute second, but still $\mathcal{O}(d^2)$

+ Gradient TD does that.

$$v \approx \mathbb{E}[x_t x_t^T]^{-1} \mathbb{E}[\rho_t \delta_t x_t]$$

× then the optimization step for least mean square:

$$v_{t+1} \doteq v_t + \beta \rho_t (\delta_t - v_t^T x_t) x_t$$

in $O(d)$ storage & per step computation

- now w_t update in SGD (simplest form)

$$w_{t+1} = w_t - \frac{1}{2} \alpha \nabla \overline{PBE}(w_t)$$

$$\approx w_t + \alpha \rho_t (x_t - \gamma x_{t+1}) x_t^T v_t \quad \boxed{\underline{GTD2}}$$

- OR, slightly better

$$\approx w_t + \alpha \rho_t (\delta_t x_t - \gamma x_{t+1} x_t^T v_t)$$

TD(0) with gradient correction (TDC) or GTD(0)

- GTD2 & TDC have 2 learning processes, one for w, one for v
+ w depends on v, but not v on w

× cascade

* must assume secondary reaches convergence first and is available for primary
* Extremely well understood and widely used stable, off-policy method

Emphatic TD Methods

- second method of efficient offpolicy learning with func approx.
- linear semi-gradient TD efficient with on-policy dist.
+ on-policy dist μ matters $p(S' | s, a)$
+ off-policy weights are ratio for target policy, but distribution is of behavior

× try reweighting states to reflect dist.

× $\boxed{\text{Emphatic TD}}$

- however one starts, as long as transitions are followed to termination, it is called on-policy
- consider discounting as "partial termination"
  + $\gamma = 0.9$, then $p(0.1)$ "termination" occurs, restarting in the "new" state.
- hence, emphatic TD is

$$\delta_t = R_{t+1} + \gamma \hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t)$$

$$w_{t+1} = w_t + \alpha M_t \rho_t \delta_t \nabla \hat{v}(S_t, w_t)$$

$$M_t = \gamma \rho_{t-1} M_{t-1} + I_t$$

- I_t is interest, M_t is emphasis
  + in theory, converges well
  * in practice, ridiculous amounts of variance do not.

Reducing Variance

- off-policy obv. higher variance than on-policy.

+ you can't learn to cook dinner by drawing

+ behavior policy must relate some way to target policy

- however, optimal policy as many neighborhood decent policies.

+ raison d'etre of off-policy is generalization

- off-policy = product of policy ratios

+ products make step-size large

x algos to try fixing step size

+ weighted importance sampling hard in func approx

+ Tree Backup ideas work well

+ or just make b not too off from π, or use some of $\underline{\pi}$, called recognizers

Summary

- it aint easy in func approx.
+ why do?: generalization, explore vs exploit.
x learn multiple things from a single stream of experience
- part one, correct targets at cost of high variance & slow learning
- part two, stability of bootstrapping
+ Deadly triad hard to avoid
x BE can't be learned
x SGD on PBE, Gradient TD, increased cost of complexity