Meta Algorithm (Approx. Dynamic Programing)

initialize π^0

for $t=0,1,\ldots$

1) $\hat{Q}^t = \text{SAMPLEANDEVAL}(\pi^t)$
2) $\pi^{t+1} = \text{IMPROVEMENT}(\hat{Q}^t)$

1) Supervision via Rollouts

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^\infty \gamma^t r(s_t,a_t) \,\middle|\, \begin{matrix} s_0, a_0 = s, a \\ P \quad \pi \end{matrix}\right]$$

Alg: InfiniteRollout:

$s_0 = s$, $a_0 = a$

for $t=0,1,\ldots$

take action a_t, observe r_t & $s_t \sim P(s_t, a_t)$

update $a_{t+1} = \pi(s_{t+1})$

return $y = \sum_{t=0}^\infty \gamma^t r_t$

Then $\mathbb{E}(y) = Q^\pi(s,a)$

Alg: Rollout:

$s_0 = s$, $a_0 = a$

for $t=0,1,\ldots$

take action a_t, observe r_t & $s_{t+1} \sim P(s_t, a_t)$

with probability $1-\gamma$:

Break and return $y = \sum_{k=0}^t r_k$

update $a_{t+1} = \pi(s_{t+1})$

$\mathbb{E}_{ALG}[y]$?

$$y = \begin{cases} r_0 & \text{w.p. } 1-\gamma \\ r_0 + r_1 & \text{w.p. } \gamma(1-\gamma) \\ r_0+r_1+r_2 & \text{w.p. } \gamma^2(1-\gamma) \\ \vdots \quad \sum_0^t r_k & \text{w.p. } \gamma^t(1-\gamma) \end{cases}$$

$\frac{1}{1-\gamma}$ $\frac{1}{N}$

$$\mathbb{E}(y) = r_0 \cancel{(1-\gamma)} \sum_0^\infty \gamma^k + r_1 \cancel{(1-\gamma)} \gamma \sum_0^\infty \gamma^k + \ldots = \sum_{k=0}^\infty \gamma^k r_k$$

y is an unbiased estimate of $Q^\pi(s,a)$

$\{(\underline{s_i}, \underline{a_i}, y_i)\}_{i=1}^{N}$

<u>Recall:</u> prediction error guarantee in supervised ML

$x \sim D_x \quad y = f_*(x) + \underbrace{w}_{0 \text{ mean}} \quad (\mathbb{E}(y) = f_\gamma(x))$

$$\mathbb{E}_{x \sim D_x}\left[(\underline{f_\gamma}(x) - \underline{\hat{f}}(x))^2\right] \le \varepsilon \quad \left(\text{usually } \varepsilon \approx O\left(\tfrac{1}{\sqrt{N}}\right)\right)$$

Connection (s,a) = features x $\quad y$ label $(Q^\pi(s,a) = f_*)$

Use discount state-action $d^\pi_{\mu_0}$!

<u>Alg: sample</u>

initialize $s_0 \sim \mu_0$ $a_0 = \pi(s_0)$

for $t = 0, 1, \ldots$

Take action a_t, observe $s_{t+1} \sim P(s_t, a_t)$

w.p. $1-\gamma$

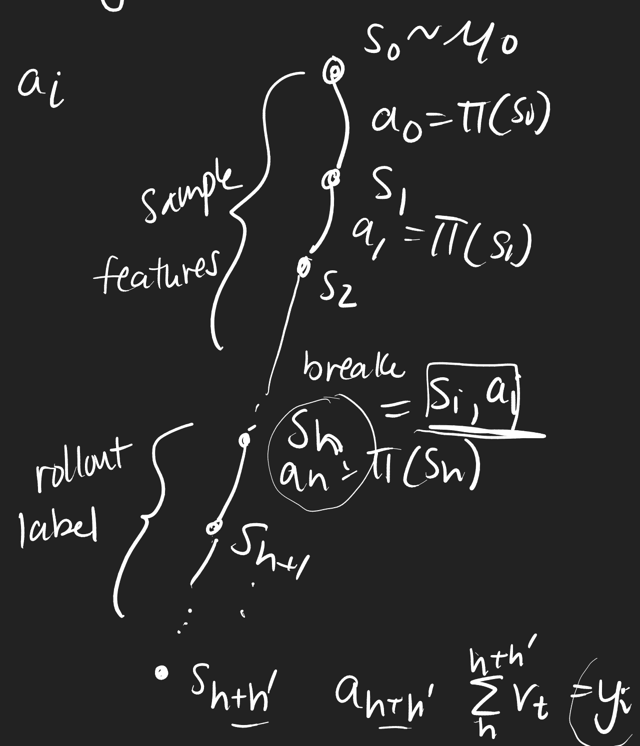
break & return a_t, s_t

update $a_{t+1} = \pi(s_{t+1})$

This algorithm is equivalent to sampling from $d^\pi_{\mu_0}$

To collect dataset, 1) sample s_i, a_i

2) rollout y_i



Terminology: MCMC

Markov Chain Monte Carlo

$\{(s_i, a_i, y_i)\}_{i=1}^{N}$

2) Approximate Policy Iteration

Putting together sampling method & estimation:

Alg: ROLLOUTEVAL(π)

for $i = 1, \dots, N$

$s_i, a_i = \text{SAMPLE}(\pi)$

$y_i = \text{ROLLOUT}(s_i, a_i, \pi)$

$$\hat{Q}^{\pi} = \underset{Q \in \mathcal{Q}}{\arg\min} \sum_{i=1}^{N} \left(Q(s_i, a_i) - y_i\right)^2$$

↑ function e.g. linear $Q(s,a) = \theta^T \phi(s,a)$

neural network

Alg: APPROX POLICY ITERATION

initialize π^0

for $t = 0, 1, \dots$

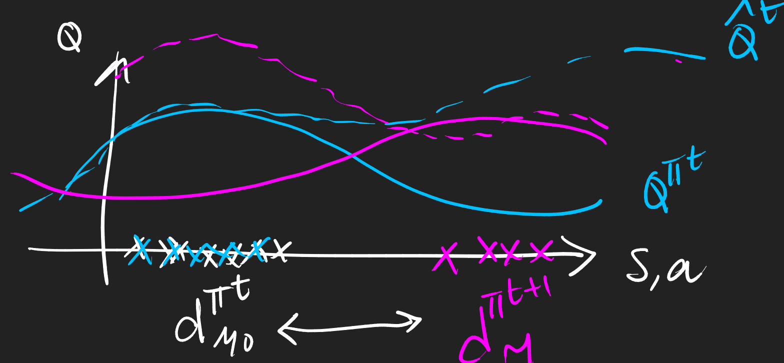
$\hat{Q}^t = \text{ROLLOUTEVAL}(\pi^t)$ ← sample & regression

$\pi_{t+1}(s) = \underset{a}{\arg\max}\ \hat{Q}^t(s,a)$ ← policy improvement

For Policy Iteration $V^{\pi^{t+1}}(s) \geq V^{\pi^t}(s)$ monotic improvement

$Q^{\pi^{t+1}}(s,a) \geq Q^{\pi^t}(s,a)$

$$\mathbb{E}_{s,a \sim d^{\pi}_{\mu_0}} \left(\hat{Q}^t(s,a) - Q^{\pi^t}(s,a)\right)^2 < \varepsilon$$ is our regression guarantee



oscillation!

4) Conservative Policy Iteration

Alg: conservative PI

initialize π^0

for $t = 0, 1, \dots$

$$\hat{Q}^t = \text{ROLLOUTEVAL}(\pi^t)$$

$$\pi'(s) = \arg\max_a \hat{Q}^t(s,a)$$

$$\pi^{t+1}(a|s) = (1-\alpha)\pi^t(a|s) + \alpha\, \pi'(a|s) \quad \forall a, s$$

$\boxed{\pi(a|s)}$ ← stochastic policy

$\pi'(a|s) = \begin{cases} 1 & \text{if } a = \arg\max \hat{Q}^t \\ 0 & \text{o.w.} \end{cases}$

randomize interpolation:
follow π^t w.p. $1-\alpha$
and π' w.p. α

α is a step size

$$\pi^{t+1}(a|s) = \pi^t(a|s) + \alpha\left(\pi'(a|s) - \pi^t(a|s)\right)$$

$$d^{\pi}_{\mu_0}(s,a) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \underbrace{\mathbb{P}^{\pi}_t(s, a; \mu_0)}_{\text{Prob. of } s_t = s,\ a_t = a \text{ under } \pi,\ s_0 \sim \mu_0}$$

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T}$$