NOTATION

1

- k : index for behavior policy
- e : index for evaluation policy (fixed)
- N : Total number of policies
- K : Number of behavior policies
- H_K : Number of trajectories, τ, in policy π_K : Horizon
- τ_t : (s_t, a_t, r_t, s_{t+1}), s = state, a = action, r = reward ; Trajectory
- π_B : [π_i for i in range([0 --- b, b+1 --- N])]
- π_E : dict {π_i : $\pi_{b \neq i}$, for i in range(N)]
- $\rho_k = \prod_{t=1}^{H_K} \frac{\pi_e(a_t^K \mid s_t^K)}{\pi_K(a_t^K \mid s_t^K)}$, Importance Sampling ratio
- ξ_k : $[{}^{*}\rho_k]$; i.e ρ_K unwrapped where size $\xi_K = H_K$ and $\xi_K^i = \frac{\pi_e(a_i^K \mid s_i^K)}{\pi_K(a_i^K \mid s_i^K)}$
- R_k^i : Return (total reward) of τ_i in policy k.
- $\sigma(\pi_e, \pi_k) = \sum_{i=1}^{H_k} R_k^i \times \rho_k^i$
- $X_1 = \frac{1}{K \cdot N} \sum_{k=1}^{K} \sigma(\pi_e, \pi_k)$

- $X_2 = \frac{1}{K \cdot N} \sum_{k=1}^{K} \sum_{i=1}^{N} R_k^i$

- $V(\pi_e) = \frac{1}{N} \sum_{i=1}^{N} R_e^i \in R^N$ (vector of values of evaluation policies for N policies)

- R_e^i : Total reward of the trajectory i of evaluation policy e.

- $$\text{Error} = \Big(\underbrace{[X_1 \;\; X_2 \;\; 1] \begin{bmatrix} c \\ d \\ e \end{bmatrix}}_{\text{estimated value function}} - \underbrace{V(\pi_e)}_{\text{true Value function}} \Big)^2$$

Sampling Trajectories 2

Define: $k, e, \text{model}_\pi, H$ // model_π ⇒ TD / Monte Carlo / PPO / DQN / etc ---

```
def getAction(state, Q): // get discrete action & associated probability (softmax)
    return argmax(Q[state]), max{ exp(Q[state]) / Σ exp(Q[state]) }
R=0 // reward total
for i in range(H):
    a, prob_k = getAction(s_t, model_πk);
    _, prob_e = getAction(s_t, model_πe);
    // note that behavior policy samples trajectories, τ

    s_t+1, r = π_k(a_t|s_t) // π // generates next state & reward from current state, s_t
    R = R+r
    --- // code
```

Importance Sampling Optimization

- Assumes IS weights are already calculated (see (1)). This section uses 1-step gradient update using MAML algorithm. Note: this is a single-task problem

- <u>Goal</u>: find set of importance sampling weights, w, such that $\rho \to 1$.

  This is also equivalent to finding parameters w_K from π_K such that

$$\sum_{i=1}^{H_K} \frac{\pi_e(a_i^K \mid s_i^K)}{\pi_K(a_i^K \mid s_i^K)} \to H_K$$

let weights, w, $= \left[\frac{1}{\pi_K(a_1^K \mid s_1^K)}, \frac{1}{\pi_K(a_2^K \mid s_2^K)}, \ldots, \frac{1}{\pi_K(a_{H_K}^K \mid s_{H_K}^K)}\right] \in \mathbb{R}^{H_K}$

let feature matrix, $X = \left[\pi_e(a_1^K \mid s_1^K), \pi_e(a_2^K \mid s_1^K), \ldots, \pi_e(a_{H_K}^K \mid s_{H_K}^K)\right] \in \mathbb{R}^{H_K}$

$$\rightarrow w \cdot X = H_K = [w_1, \ldots, w_{H_K}] \begin{bmatrix} X_1 \\ \vdots \\ X_{H_K} \end{bmatrix} = H_K \rightarrow (1 \times H_K)(H_K \times 1) = \underset{\in \mathbb{R}^1}{\underline{1 \times 1}}$$

- $X \to$ fixed, $w \to$ parameters to optimize.

Set MAML framework such that $\underline{\underline{\text{MAML}(w) \cdot X - H_K < w \cdot X - H_K}}$

MAML
- while not done do:
  - Evaluate $\nabla_w \mathcal{L}_X(f(w))$
  - Compute adapted parameters w/ SGD: $w_i' = w - \alpha \nabla_w \mathcal{L}_X(f(w))$
- $w \leftarrow w - \beta \nabla_w \sum \mathcal{L}_X(f w_i')$

- Let $N=3$

→ data point 1:

$$\pi_k = \{\pi_2, \pi_3\}, \quad \pi_e = \pi_1$$

→ data point 2:

$$\pi_k = \{\pi_1, \pi_3\}, \quad \pi_e = \pi_2$$

→ data point 3:

$$\pi_k = \{\pi_1, \pi_2\}, \quad \pi_e = \pi_3$$

- Sample trajectories based on π_K & π_e (see (2))
- MAML (π_K), see (3)
- Compute ρ_K
- Compute $\sigma(\pi_e, \pi_k)$
- Compute $V(\pi_e)$
- Compute X_1 X_2
- Find MSE.; $\left([X_1\ X_2\ 1] \begin{bmatrix} c \\ d \\ e \end{bmatrix} - V(\pi_e) \right)^2$ — ☆
- Optimize parameters c, d, and e such that $(☆) = \min$