Recap: Model-free RL

- **Goal:** solve an MDP = "choose a policy that maximizes cumulative (discounted) reward"

Typically represented as a tuple

$$\mathcal{M} = (\mathcal{X}, \mathcal{U}, T, R, \gamma)$$

$$\pi^* = \arg\max_{\pi} \mathbb{E}_p \left[\sum_{t \geq 0} \gamma^t R\left(x_t, \pi\left(x_t\right)\right) \right]$$

- Value functions can be decomposed into immediate reward plus discounted value of successor state

Bellman Expectation Equation

$$\begin{aligned} \mathrm{V}_\pi\left(x_t\right) &= \mathbb{E}_\pi\left[R\left(x_t, \pi\left(x_t\right)\right) + \gamma \mathrm{V}_\pi\left(x_{t+1}\right)\right] \\ &= R\left(x_t, \pi\left(x_t\right)\right) + \gamma \sum_{x_{t+1} \in X} T\left(x_{t+1} \mid x_t, \pi\left(x_t\right)\right) \mathrm{V}_\pi\left(x_{t+1}\right) \end{aligned}$$

Bellman Optimality Equation

$$V^*\left(x_t\right) = \max_u \left(R\left(x_t, u_t\right) + \gamma \sum_{x_{t+1} \in X} T\left(x_{t+1} \mid x_t, u_t\right) V^*\left(x_{t+1}\right) \right)$$

- Bellman equations can be used to solve known MDPs:

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

Recap: Model-free RL

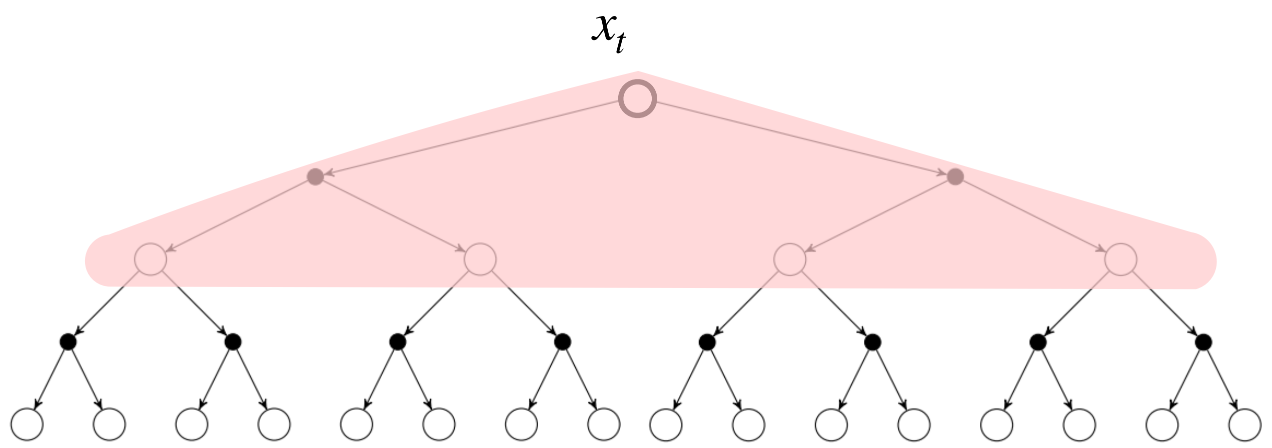
- We discussed different ways to estimate value functions

Dynamic Programming

Exact

Requires knowledge of MDP

$$\hat{V}(x_t) \leftarrow \mathbb{E}\left[R_t + \gamma \hat{V}(x_{t+1})\right]$$



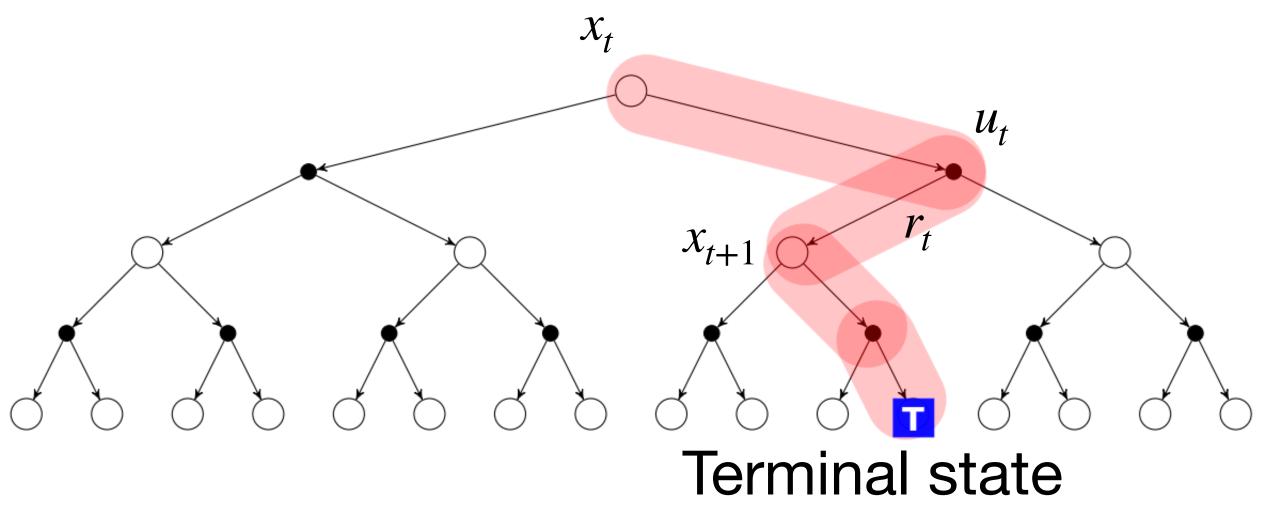
$$\hat{Q}(x_t, u_t) \leftarrow \mathbb{E}\left[R_t + \gamma \hat{Q}(x_{t+1}, u_{t+1})\right]$$

Monte Carlo

Unbiased

High variance; must reach terminal state

$$\hat{V}(x_t) \leftarrow \hat{V}(x_t) + \alpha\left(G_t - \hat{V}(x_t)\right)$$



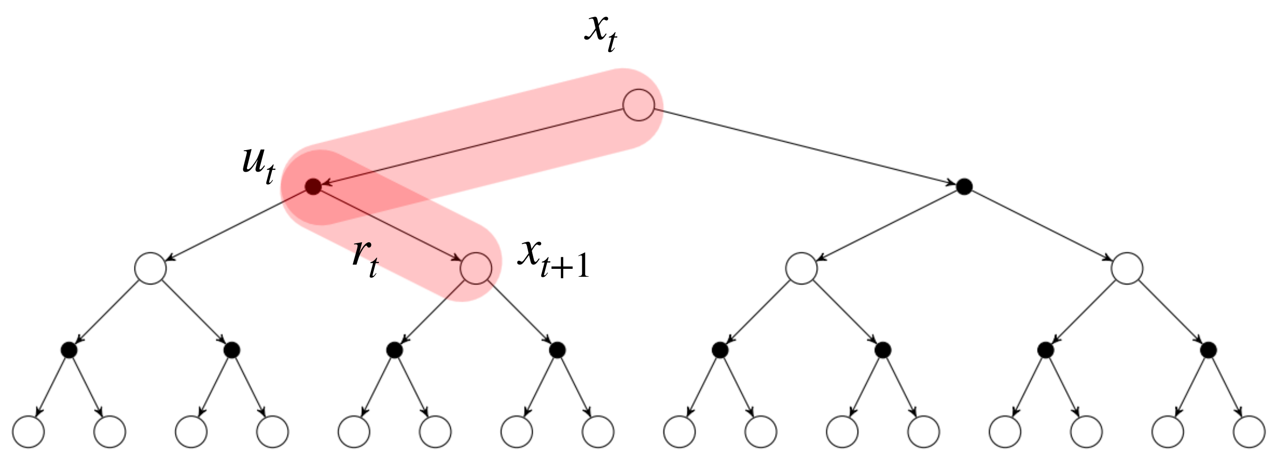
$$\hat{Q}(x_t, u_t) \leftarrow \hat{Q}(x_t, u_t) + \alpha\left(G_t - \hat{Q}(x_t, u_t)\right)$$

Temporal-Difference

Low variance; can learn online

Biased

$$\hat{V}(x_t) \leftarrow \hat{V}(x_t) + \alpha\left(R_t + \gamma \hat{V}(x_{t+1}) - \hat{V}(x_t)\right)$$



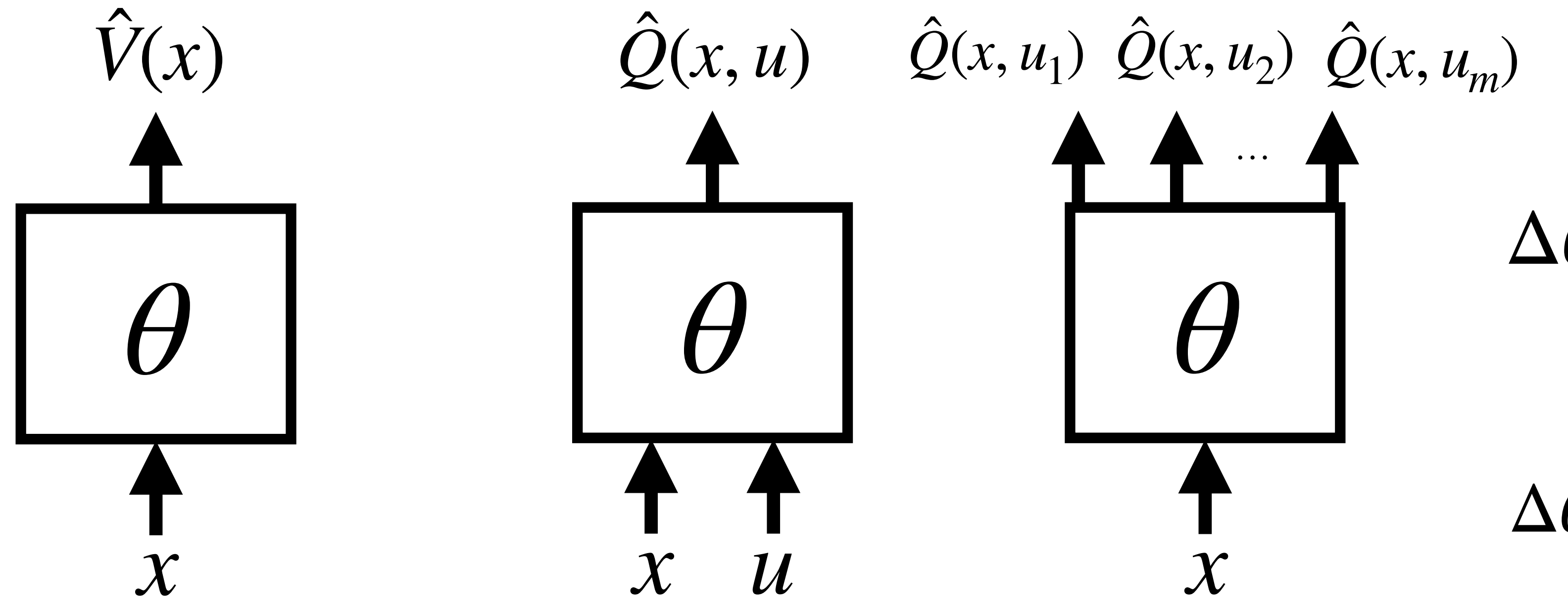
$$\hat{Q}(x_t, u_t) \leftarrow \hat{Q}(x_t, u_t) + \alpha\left(R_t + \gamma \hat{Q}(x_{t+1}, u_{t+1}) - \hat{Q}(x_t, u_t)\right)$$

- And how to scale these ideas through function approximation

Tabular representation:

$$\hat{V}(x) = \begin{bmatrix} \hat{V}(x_1) \\ \hat{V}(x_2) \\ \vdots \\ \hat{V}(x_n) \end{bmatrix} \quad \hat{Q}(x, u) = \begin{bmatrix} \hat{Q}(x_1, u_1) & \hat{Q}(x_1, u_2) & \dots & \hat{Q}(x_1, u_m) \\ \hat{Q}(x_2, u_1) & \hat{Q}(x_2, u_2) & \dots & \hat{Q}(x_2, u_m) \\ & \vdots & & \\ \hat{Q}(x_n, u_1) & \hat{Q}(x_n, u_2) & \dots & \hat{Q}(x_n, u_m) \end{bmatrix}$$

Function approximation:



MC update

$$\Delta\theta = \alpha\left(G_t - \hat{V}_\theta(x_t)\right)\nabla_\theta \hat{V}_\theta(x_t)$$

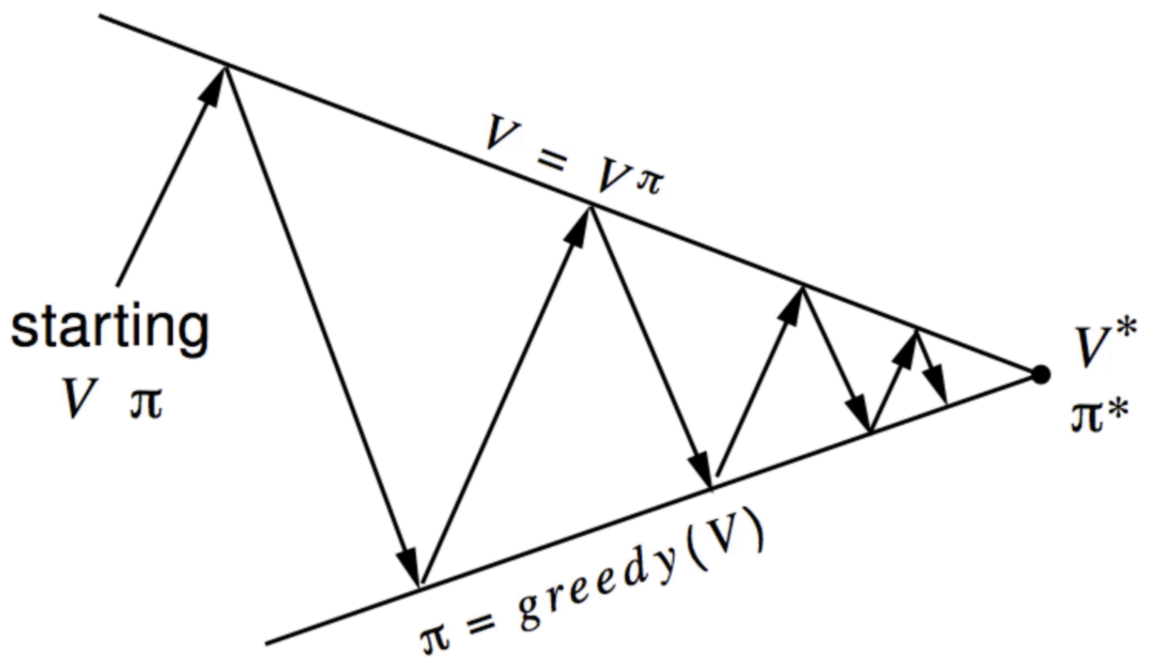
TD update

$$\Delta\theta = \alpha\left(r_t + \gamma \hat{V}_\theta(x_{t+1}) - \hat{V}_\theta(x_t)\right)\nabla_\theta \hat{V}_\theta(x_t)$$

Recap: Model-free RL

Value-based methods

- Generalized Policy Iteration



- Sarsa & Q-learning

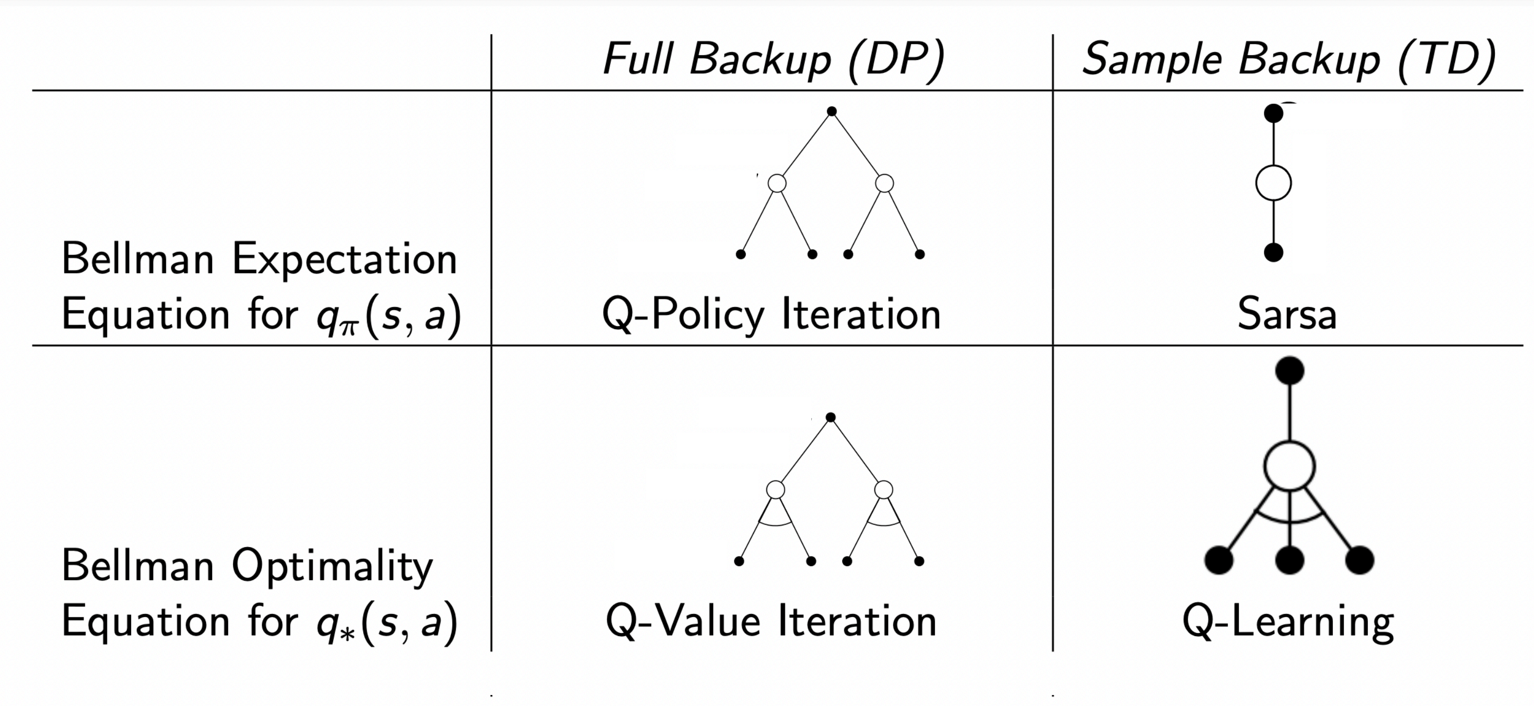
SARSA: on-policy

$$Q(x_t, u_t) \leftarrow Q(x_t, u_t) + \alpha \left(r_t + \gamma Q\left(x_{t+1}, u_{t+1}\right) - Q(x_t, u_t) \right)$$

Q-learning: off-policy

$$Q(x_t, u_t) \leftarrow Q(x_t, u_t) + \alpha \left(r_t + \gamma \max_{u'_{t+1}} Q\left(x_{t+1}, u'_{t+1}\right) - Q(x_t, u_t) \right)$$

On-policy: evaluate or improve the policy that is used to make decisions
Off-policy: evaluate or improve a policy different from that used to generate the data

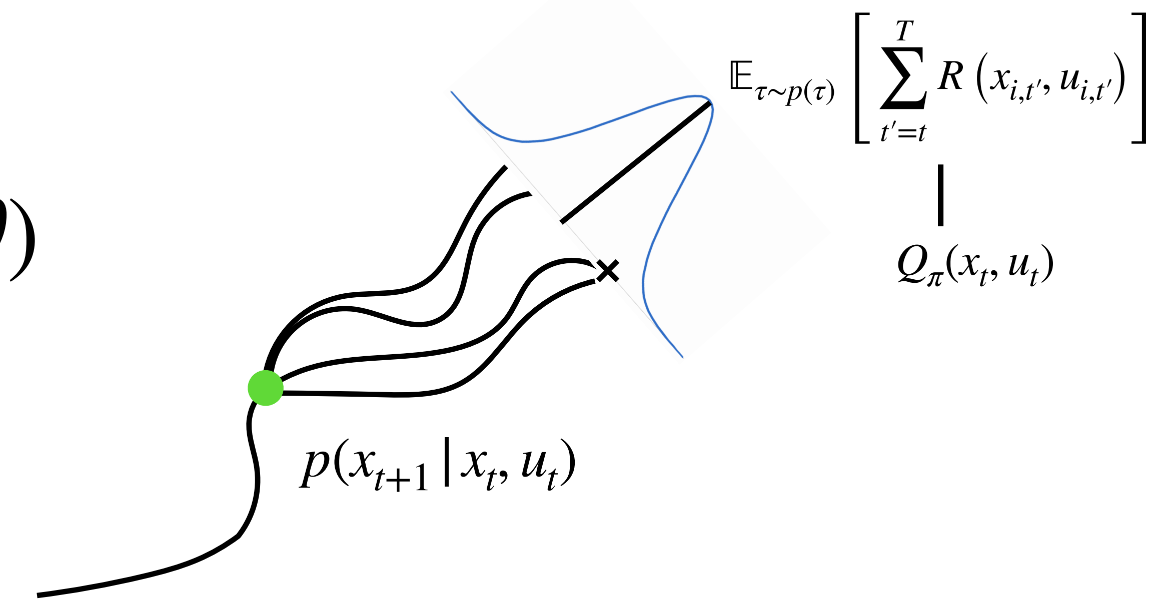


- Deep RL:
  (1) Use **deep neural nets** to represent Q_θ
  (2) Uses **experience replay** and **fixed Q-targets**

- In policy optimization, we care about learning an (explicit) parametric policy π_θ, with parameters θ to directly maximize:

$$\theta^* = \arg\max_\pi \underbrace{\mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{t \geq 0} \gamma^t R\left(x_t, u_t\right) \right]}_{J(\theta)}$$

(1) estimate its gradient $\nabla_\theta J(\theta)$
(2) do approximate gradient ascent on $J(\theta)$: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



Problem: high variance of PG

Solution: baselines, "critics"

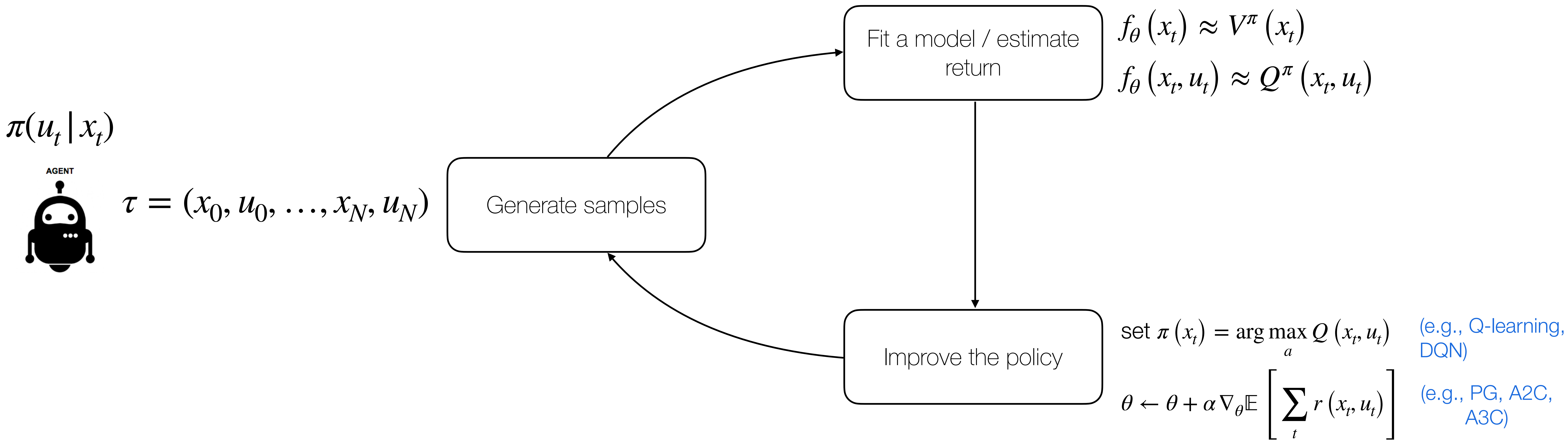
Policy gradient: $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(u_{i,t} \mid x_{i,t}\right) \right) \left(\sum_{t=1}^{T} R\left(x_{i,t}, u_{i,t}\right) \right) \right]$

Maximum Likelihood: $\nabla_\theta J_{MLE}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(u_{i,t} \mid x_{i,t}\right) \right) \right]$

"Change parameters θ s.t. trajectories with higher reward have higher probability"

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(u_{i,t} \mid x_{i,t}\right) Q_\phi(x_t, u_t)$$

Policy Optimization



$\pi(u_t | x_t)$
AGENT
$\tau = (x_0, u_0, \ldots, x_N, u_N)$
Generate samples
Fit a model / estimate return
$f_\theta\left(x_t\right) \approx V^\pi\left(x_t\right)$
$f_\theta\left(x_t, u_t\right) \approx Q^\pi\left(x_t, u_t\right)$
Improve the policy
set $\pi\left(x_t\right) = \arg\max_a Q\left(x_t, u_t\right)$
(e.g., Q-learning, DQN)
$\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}\left[\sum_t r\left(x_t, u_t\right)\right]$
(e.g., PG, A2C, A3C)

Recap: Model-based RL

- In model-based RL, we aim to (1) **estimate an approximate model** of the dynamics, and (2) **use it for control**

Approach 1: "learn a model $p(x_{t+1} | x_t, u_t)$ from experience and **use it to plan**"

1. Run base policy $\pi_0(u_t | x_t)$ in the environment (e.g., random policy, exploration policy) and collect dataset of transitions $\mathcal{D} = \{(x_t, u_t, x_{t+1})_i\}$
2. Fit dynamics model to data to minimize error (or equivalently, maximize (log) likelihood)

$$\theta^* = \arg\min_{\theta} \sum_i \left\| f_\theta\left(x_t, u_t\right) - x_{t+1} \right\|^2$$

YES

Sys. ID

NO

Distribution mismatch

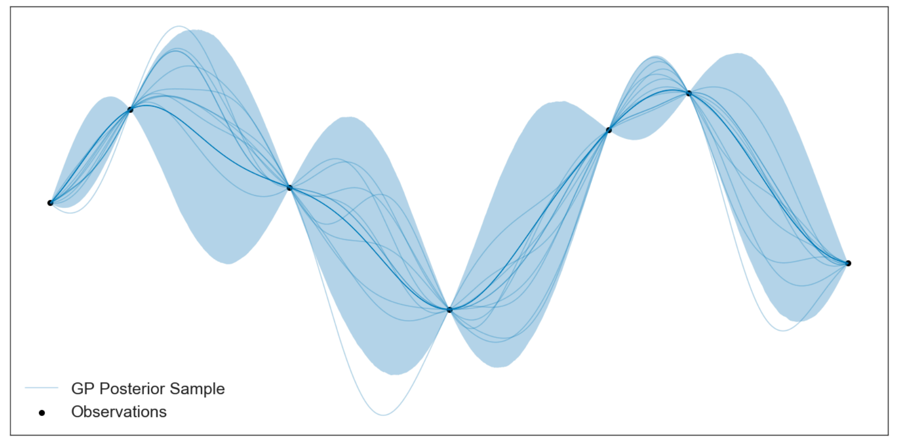
Exploitation of errors

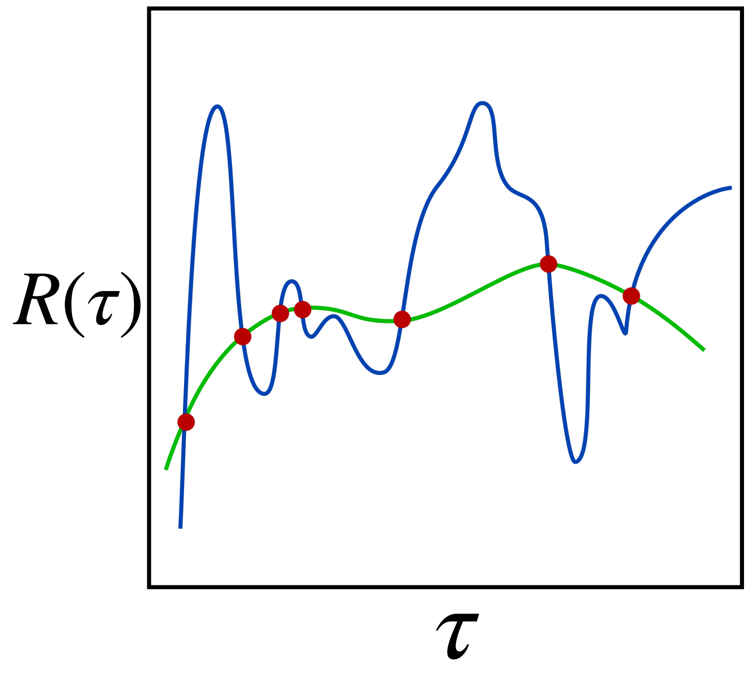
3. Use the learned model to plan a sequence of actions

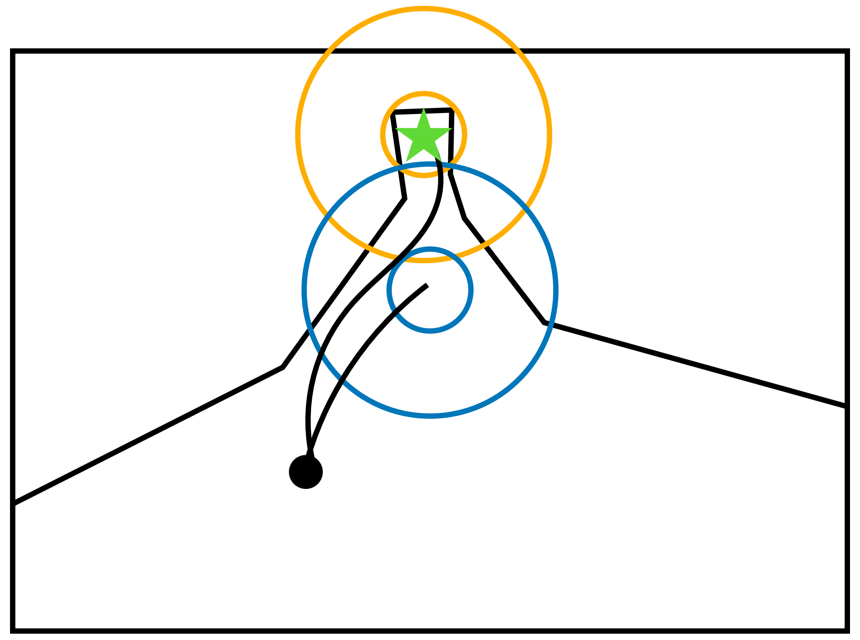
Problem: we'll likely *erroneously* exploit our model where it is less knowledgeable

(Possible) Solution: consider how "certain" we are our about the prediction

- A structured way to represent uncertainty over a parametric model is through a **posterior distribution** over the parameters $p(\theta | \mathcal{D})$







Recap: Model-based RL

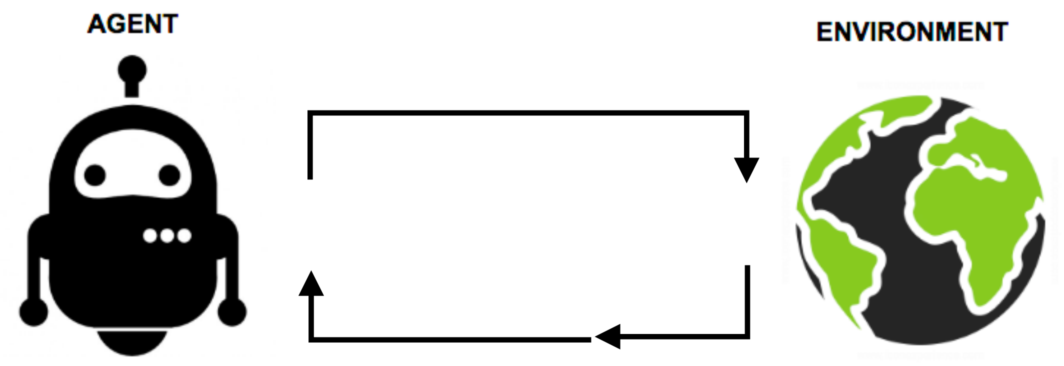
- In model-based RL, we aim to (1) **estimate an approximate model** of the dynamics, and (2) **use it for control**

Approach 2: "learn a model $p(x_{t+1} | x_t, u_t)$ from experience and **improve model-free learning**"

- Having a model enables us to consider two sources of experience

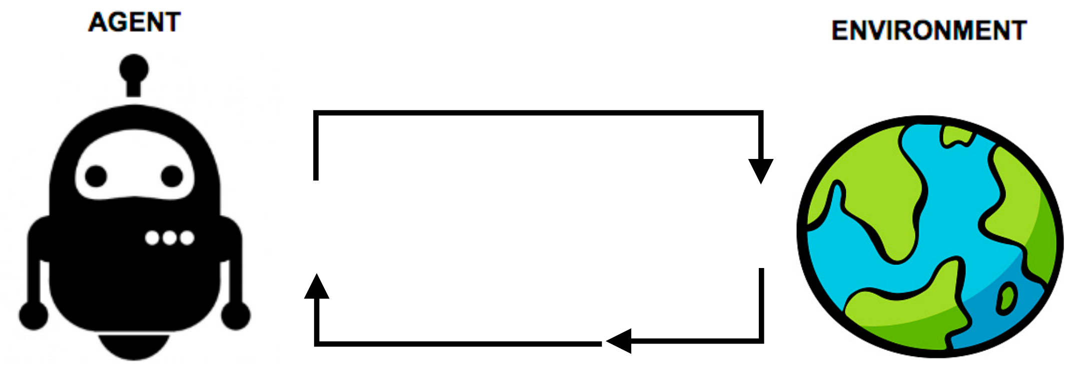
Real experience: sampled from the environment (true MDP)

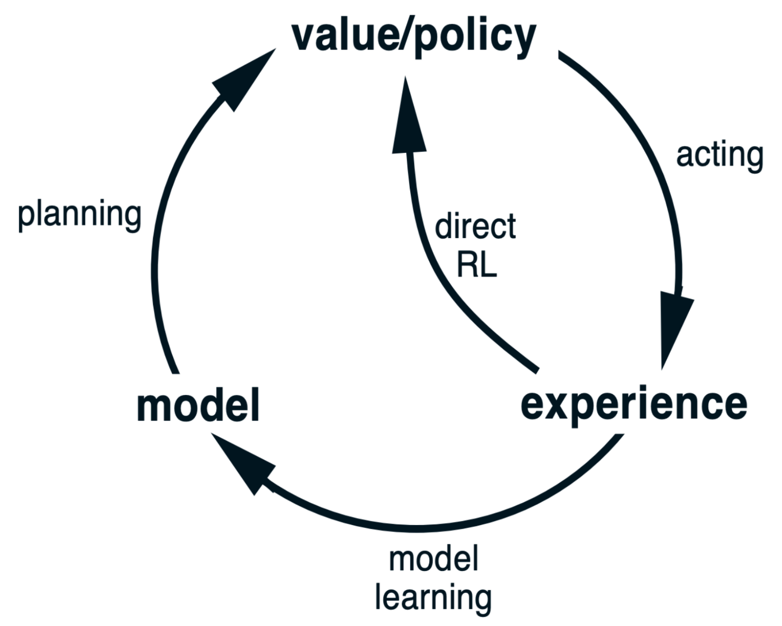
$$x_{t+1} \sim P(x_{t+1} | x_t, u_t)$$
$$R_t = R(x_t, u_t)$$



Simulated experience: sampled from the model (approximate MDP)

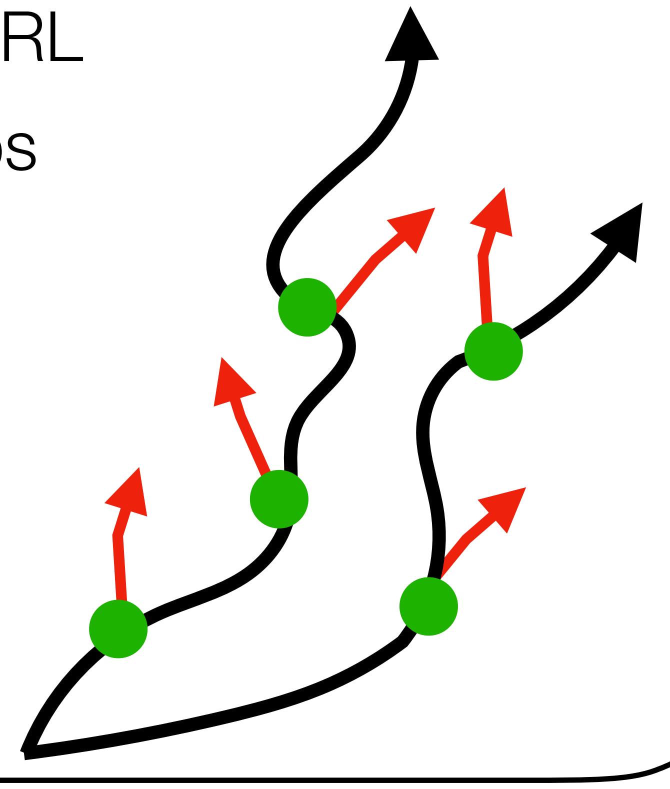
$$x_{t+1} \sim p_\theta(x_{t+1} | x_t, u_t)$$
$$R_t = r_\theta(x_t, u_t)$$

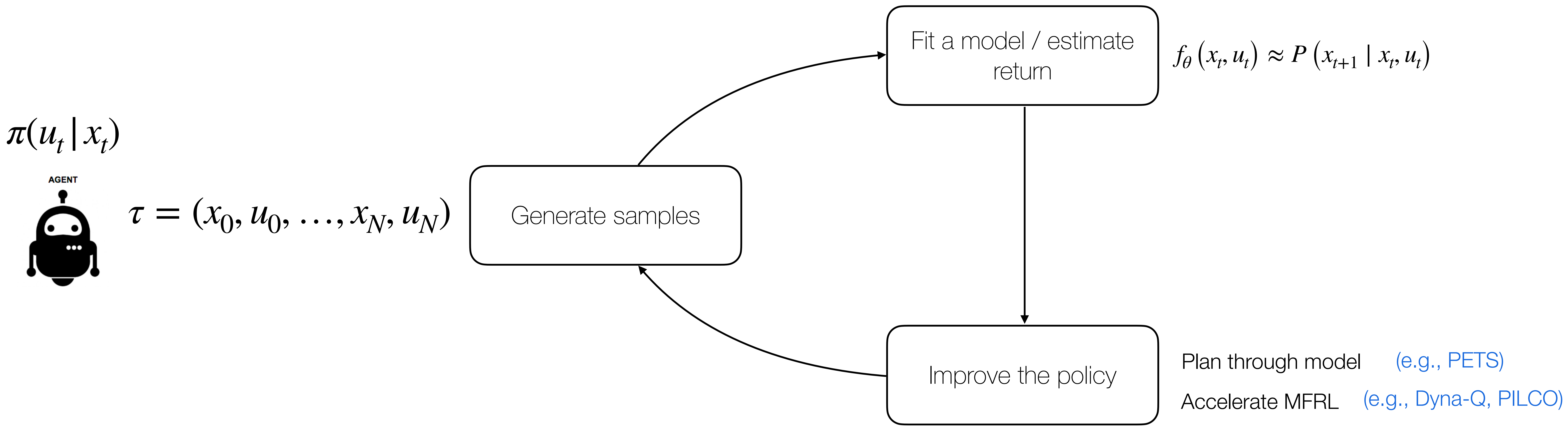




- Dyna-style algorithms:

1. Collect data $\{(x_t, u_t, r_t, x_{t+1})\}$
2. Learn dynamics / reward model, i.e., $p_\theta(x_{t+1} | x_t, u_t)$, $r_\theta(x_t, u_t)$
3. Repeat n times
    1. Sample x_t from buffer
    2. Choose action u_t (from dataset, π, random, exploration, etc.)
    3. Simulate dynamics / reward $\hat{x}_{t+1} \sim p_\theta(x_{t+1} | x_t, u_t)$, $\hat{r}_t = r_\theta(x_t, u_t)$
    4. Train on $\{(x_t, u_t, \hat{r}_t, \hat{x}_{t+1})\}$ via model-free RL
    5. Optionally, take k more model-based steps




$\pi(u_t \mid x_t)$
AGENT
$\tau = (x_0, u_0, \ldots, x_N, u_N)$
Generate samples
Fit a model / estimate return
$f_\theta(x_t, u_t) \approx P(x_{t+1} \mid x_t, u_t)$
Improve the policy
Plan through model (e.g., PETS)
Accelerate MFRL (e.g., Dyna-Q, PILCO)

Recap: Central idea of this class

Value functions

$$V(x), Q(x, u)$$

- For example, in LQR:

Given the problem definition:

$$J_0\left(\mathbf{x}_0\right) = \frac{1}{2}\mathbf{x}_N^T Q_N \mathbf{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}\left(\mathbf{x}_k^T Q_k \mathbf{x}_k + \mathbf{u}_k^T R_k \mathbf{u}_k + 2\mathbf{x}_k^T S_k \mathbf{u}_k\right)$$

We derived the Riccati recursion

$$\begin{aligned} J_{N-1}^*(\mathbf{x}_{N-1}) &= \min_{\mathbf{u}_{N-1}} \frac{1}{2}\left(\begin{bmatrix}\mathbf{x}_{N-1}\\ \mathbf{u}_{N-1}\end{bmatrix}^T \begin{bmatrix} Q_{N-1} & S_{N-1}\\ S_{N-1}^T & R_{N-1}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{N-1}\\ \mathbf{u}_{N-1}\end{bmatrix} + \mathbf{x}_N^T P_N \mathbf{x}_N\right) \\ &= \min_{\mathbf{u}_{N-1}} \frac{1}{2}\left(\begin{bmatrix}\mathbf{x}_{N-1}\\ \mathbf{u}_{N-1}\end{bmatrix}^T \begin{bmatrix} Q_{N-1} & S_{N-1}\\ S_{N-1}^T & R_{N-1}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{N-1}\\ \mathbf{u}_{N-1}\end{bmatrix} + \right. \\ &\qquad \left. (A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1})^T P_N (A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1})\right) \end{aligned}$$

- In MPC:

We discussed terminal cost and constraint set to ensure (1) Persistent feasibility

(2) Stability

$$\min_{u_{t|t},\ldots,u_{t+N-1|t}} l_T\left(x_{t+N|t}\right) + \sum_{k=0}^{N-1} l\left(x_{t+k|t}, u_{t+k|t}\right)$$

s.t $x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k|t}, k = 0,\ldots,N-1$

$x_{t+k|t} \in X, \quad k = 0,\ldots,N-1$

$u_{t+k|t} \in U, \quad k = 0,\ldots,N-1$

$x_{t+N|t} \in X_f$

$x_{t|t} = x(t)$