University of California, Irvine

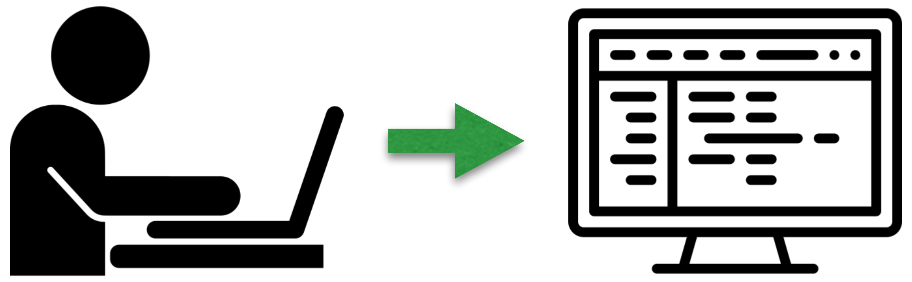
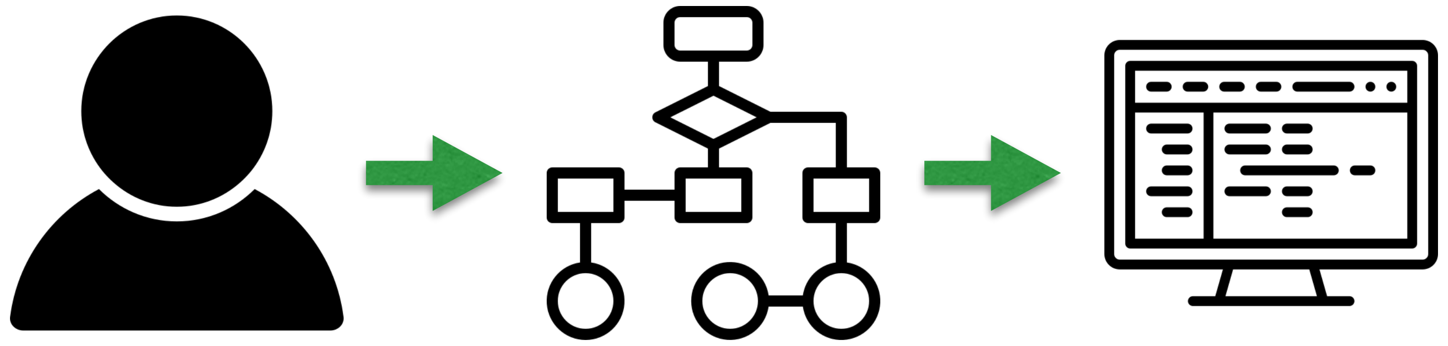
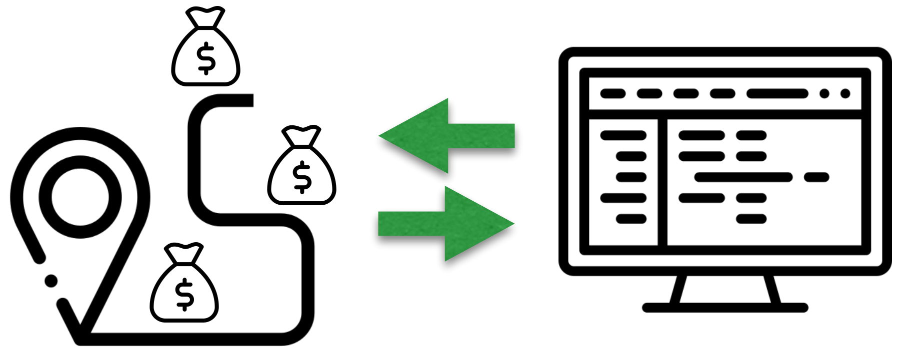
CS 277: Control and Reinforcement Learning

Winter 2021

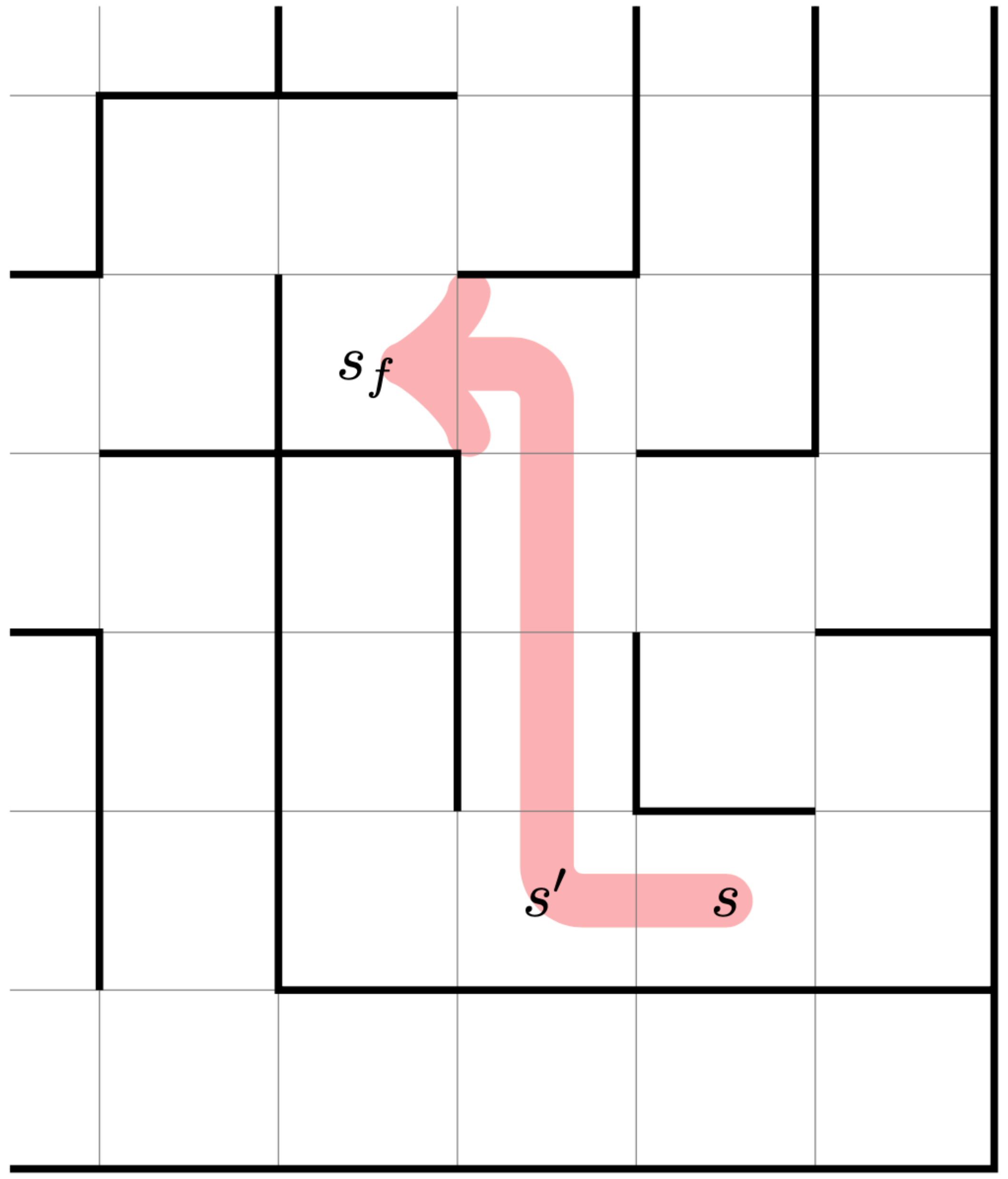
Review

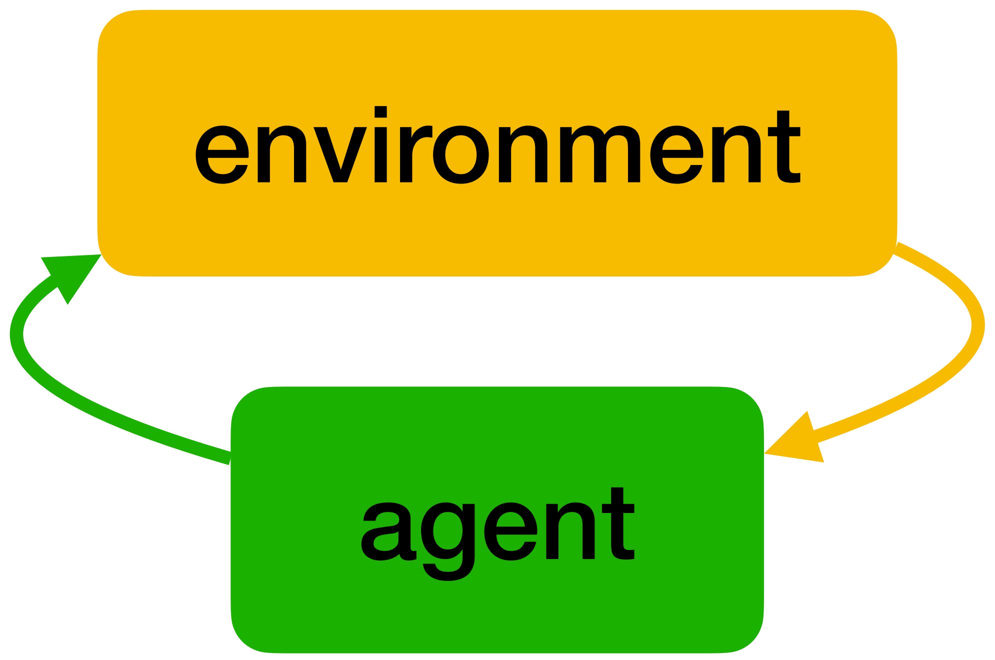
Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

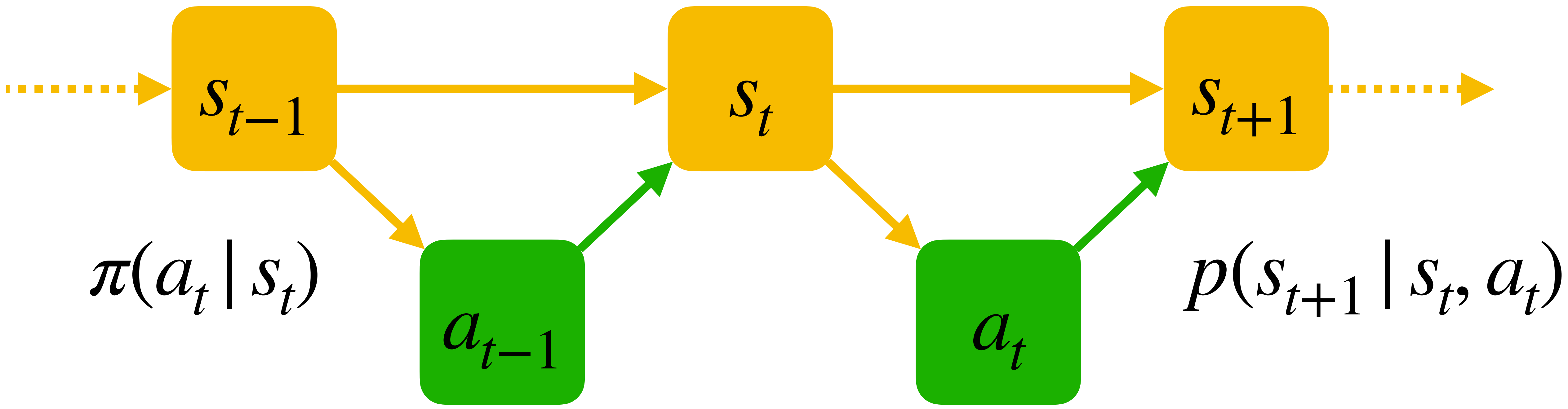
Control preference elicitation

	Explicit	Implicit
"how"	Programming 	Imitation Learning 
"what"	Instruction following 	Reinforcement Learning 

System = agent + environment





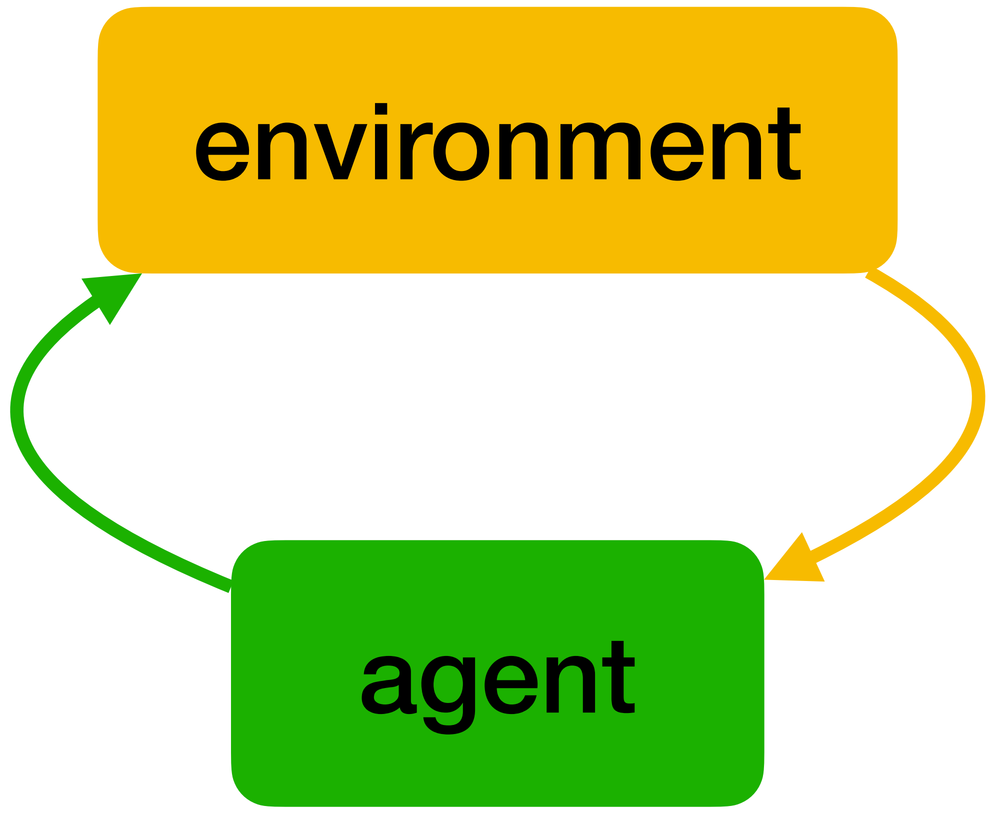


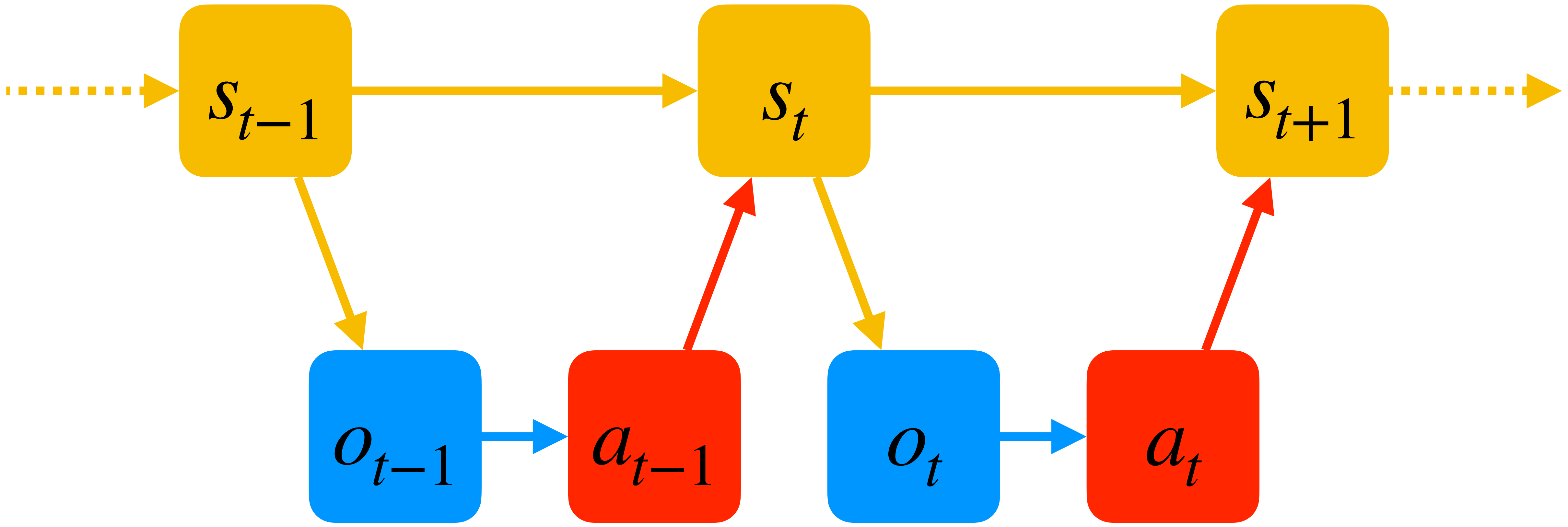
Basic RL concepts

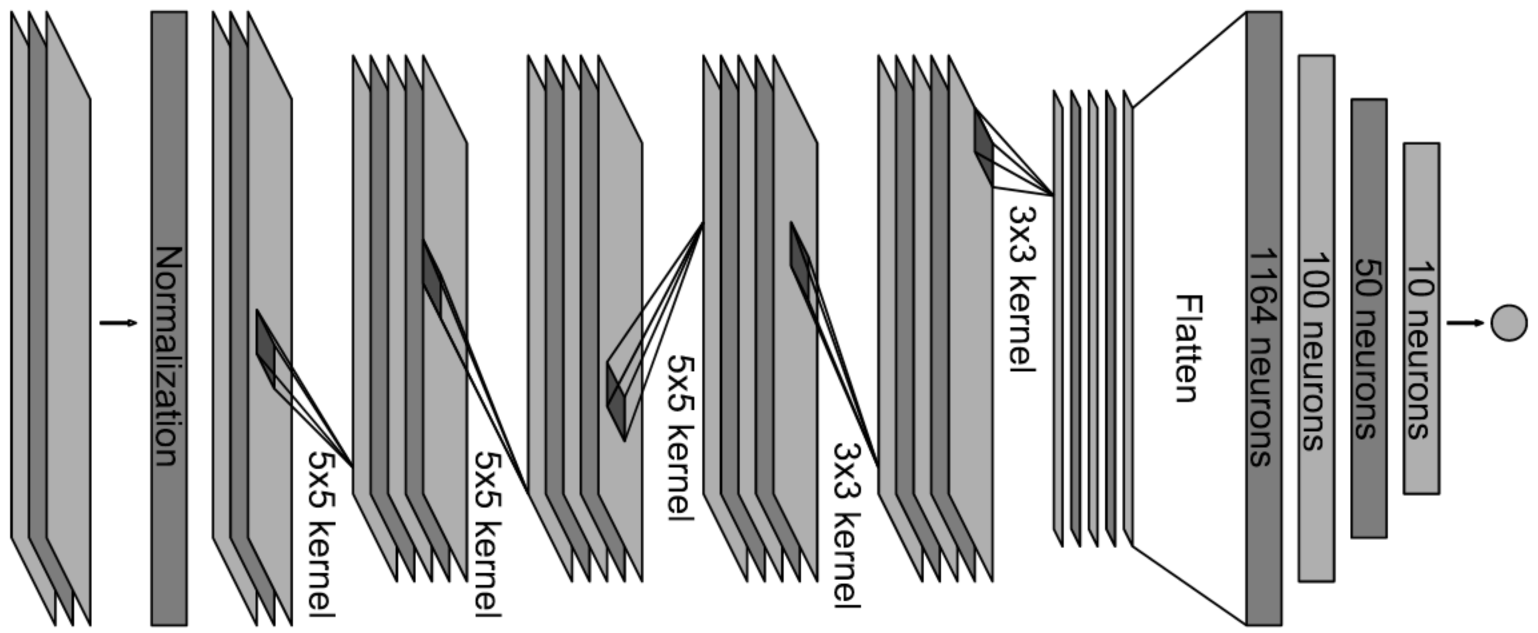
- State: $s \in S$; action: $a \in A$; reward: $r(s, a) \in \mathbb{R}$
- Dynamics: $p(s_{t+1} \mid s_t, a_t)$ for stochastic; $s_{t+1} = f(s_t, a_t)$ for deterministic
- Policy: $\pi(a_t \mid s_t)$ for stochastic; $a_t = \pi(s_t)$ for deterministic
- Trajectory: $p_\pi(\xi = s_0, a_0, s_1, a_1, \ldots) = p(s_0) \prod_t \pi(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)$
- Return: $R(\xi) = \sum_t \gamma^t r(s_t, a_t) \qquad 0 \leq \gamma < 1$
- Value: $V(s) = \mathbb{E}_{\xi \sim p_\pi}[R \mid s_0 = s]$

  $Q(s, a) = \mathbb{E}_{\xi \sim p_\pi}[R \mid s_0 = s, a_0 = a]$

A policy is a (stochastic) function







$\pi(a_t | o_t)$

observation

action

Horizon classes

- Finite: $R(\xi) = \sum_{t=0}^{T-1} r(s_t, a_t)$
- Infinite: $R(\xi) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} r(s_t, a_t)$
- Discounted: $R(\xi) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \qquad 0 \le \gamma < 1$
- Episodic: $R(\xi) = \sum_{t=0}^{T-1} r(s_t, a_t)$ s.t. $s_T = s_f$

Recap

- Marginal state distributions can be computed recursively forward

$$p_\pi(s') = \mathbb{E}_{s \sim p_\pi}[\mathbb{E}_{a|s \sim \pi}[p(s' \mid s, a)]]$$

- Value functions can be computed recursively backward

$$V_\pi(s) = \mathbb{E}_{a|s \sim \pi}[r(s, a) + \gamma \mathbb{E}_{s'|s,a \sim p}[V_\pi(s')]]$$

- Forward and backward recursions are a recurring theme...

Recap

- Trajectory: $\xi = s_0, a_0, s_1, a_1, s_2, \ldots$
- Probability of a trajectory: $p_\pi(\xi) = p(s_0) \prod_t \pi(a_t \,|\, s_t) p(s_{t+1} \,|\, s_t, a_t)$
- Marginal expectation:

$$\mathbb{E}_{\xi \sim p_\pi}[f(s_t)] = \sum_\xi p_\pi(\xi) f(s_t)$$

Recap

- Trajectory: $\xi = s_0, a_0, s_1, a_1, s_2, \dots$
- Probability of a trajectory: $p_\pi(\xi) = p(s_0) \prod_t \pi(a_t | s_t) p(s_{t+1} | s_t, a_t)$
- Marginal expectation:

$$\mathbb{E}_{\xi \sim p_\pi}[f(s_t)] = \sum_\xi p_\pi(\xi) f(s_t)$$

$$= \mathbb{E}_{s_0 \sim p}[\mathbb{E}_{a_0 | s_0 \sim \pi}[\cdots \mathbb{E}_{s_t | s_{t-1}, a_{t-1} \sim p}[\cdots \mathbb{E}_{s_T | s_{T-1}, a_{T-1} \sim p}[f(s_t)] \cdots] \cdots]]$$

$$= \mathbb{E}_{s_0 \sim p}[\mathbb{E}_{a_0 | s_0 \sim \pi}[\cdots \mathbb{E}_{s_t | s_{t-1}, a_{t-1} \sim p}[f(s_t)] \cdots]]$$

$$= \mathbb{E}_{s_t \sim p_\pi}[\mathbb{E}_{\xi | s_t \sim p_\pi}[f(s_t)]] = \mathbb{E}_{s_t \sim p_\pi}[f(s_t)]$$

Recap

- We often take expectations over functions $f(\xi)$ that decompose into a sum
- Finite horizon: $\mathbb{E}_{\xi\sim p_\pi}[f(\xi)] = \mathbb{E}_{\xi\sim p_\pi}\left[\sum_{t=0}^{T-1} f(s_t)\right]$
  - Where $p_\pi(s) = \frac{1}{T}\sum_{t=0}^{T-1} p_\pi(s_t)$, i.e. $p_\pi(s) = \sum_{t=0}^{T-1} p_\pi(t, s_t)$ with $t \sim \mathrm{U}(\{0,\dots,T-1\})$
- Discounted horizon: $\mathbb{E}_{\xi\sim p_\pi}[f(\xi)] = \mathbb{E}_{\xi\sim p_\pi}\left[\sum_{t=0}^{\infty} \gamma^t f(s_t)\right]$
  - Where $p_\pi(s) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t p_\pi(s_t)$, i.e. $p_\pi(s) = \sum_{t=0}^{\infty} p_\pi(t, s_t)$ with $t \sim \mathrm{Geom}(1-\gamma)$

Recap

- We often take expectations over functions $f(\xi)$ that decompose into a sum
- Finite horizon: $\mathbb{E}_{\xi\sim p_\pi}[f(\xi)] = \mathbb{E}_{\xi\sim p_\pi}\left[\sum_{t=0}^{T-1} f(s_t)\right] = T\mathbb{E}_{s\sim p_\pi}[f(s)]$
  - Where $p_\pi(s) = \frac{1}{T}\sum_{t=0}^{T-1} p_\pi(s_t)$, i.e. $p_\pi(s) = \sum_{t=0}^{T-1} p_\pi(t, s_t)$ with $t \sim \mathrm{U}(\{0,\dots,T-1\})$
- Discounted horizon: $\mathbb{E}_{\xi\sim p_\pi}[f(\xi)] = \mathbb{E}_{\xi\sim p_\pi}\left[\sum_{t=0}^{\infty} \gamma^t f(s_t)\right] = \frac{1}{1-\gamma}\mathbb{E}_{s\sim p_\pi}[f(s)]$
  - Where $p_\pi(s) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t p_\pi(s_t)$, i.e. $p_\pi(s) = \sum_{t=0}^{\infty} p_\pi(t, s_t)$ with $t \sim \mathrm{Geom}(1-\gamma)$

Taxonomy

- **Imitation Learning**
  - Off-policy
    - BC
    - DART
    - GAIL
  - On-policy
    - DAgger
- **Reinforcement Learning**
  - Temporal Difference
    - DQN
  - Policy Gradient
    - PG
  - DDPG
  - A2C
  - TRPO
- **Model-Based Learning**
  - Planning
    - iLQR
    - MPC
  - MFRL w/ model
    - Dyna

Imitation Learning

- Pros:
  - Teacher action is a globally optimal learning signal
  - Experience distribution is already focused on good behavior
  - Safe
- Cons:
  - Teacher and learner action spaces may mismatch
  - Covariate shift: training distribution $\neq$ test distribution
  - Expensive
  - Teacher may be fallible, inconsistent, etc.

Behavior Cloning (BC)

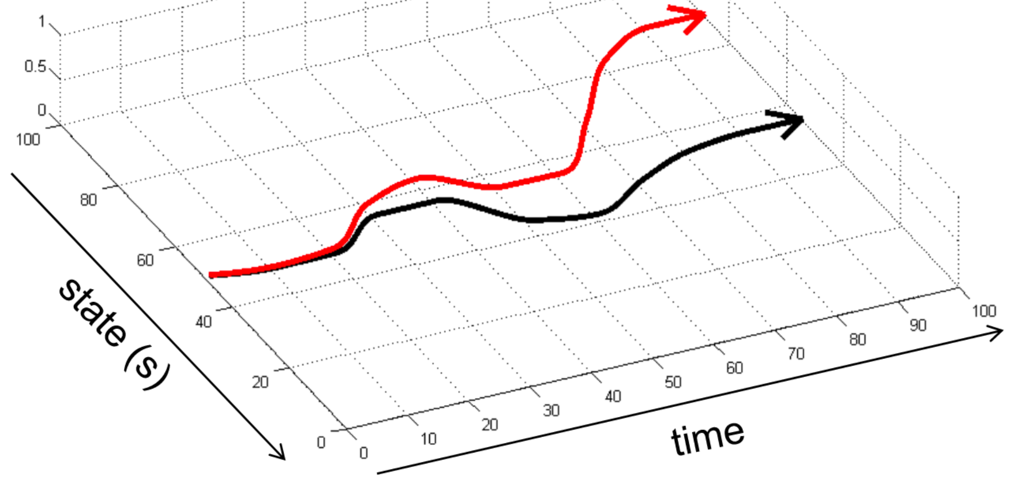
- The simplest IL is just supervised learning:
  - Break trajectories into examples (s_t, a_t)
  - Learn a function $\pi : s \mapsto a$, or a distribution $\pi(a \mid s)$
- One possible loss: negative log-likelihood $\mathscr{L} = -\sum_{(s,a)\in\mathscr{D}} \log \pi(a \mid s)$

Inaccuracy in BC

observations + actions → **training data** → **supervised learning** → $\pi_\theta(a_t | o_t)$

- The state transition distribution is linear in the policy
$$p_\pi(s_{t+1} | s_t) = \sum_{o_t, a_t} p(o_t | s_t)\pi(a_t | o_t)p(s_{t+1} | s_t, a_t)$$

no data here!



- If the policy approximates the teacher $\pi_\theta(a_t | o_t) \approx \pi^*(a_t | o_t)$
  - The dynamics approximates the teacher behavior $p_{\pi_\theta}(s_{t+1} | s_t) \approx p_{\pi^*}(s_{t+1} | s_t)$
- But errors accumulate over time
  - May reach states not seen in the training dataset

Behavior Cloning

- Pros:
  - Simple
  - Data-efficient
- Cons:
  - Very susceptible to train–test mismatch

DAgger: Dataset Aggregation

- Can we collect demonstration data for $p_{\pi_\theta}(o_t)$?

Algorithm 1 DAgger

Collect dataset $\mathcal{D}$ of teacher demonstrations
$(o_0, a_0^*, o_1, a_1^*, \ldots) \sim p_{\pi^*}$
Train π_θ on $\mathcal{D}$
Execute π_θ to get $(o_0, a_0, \ldots) \sim p_{\pi_\theta}$
Ask teacher to label $a_t^* | o_t \sim \pi^*$
Aggregate $(o_0, a_0^*, o_1, a_1^*, \ldots)$ into $\mathcal{D}$
Repeat!

but how? challenging...

- DAgger can reduce the imitation loss from $O(\epsilon T^2)$ to $O(\epsilon T)$

DAgger

- Pros:
  - Addresses covariate shift
  - Good theoretical guarantees
- Cons:
  - Technically challenging: UI for human feedback
  - Burdens human teacher: how to provide good action in weird states, no context

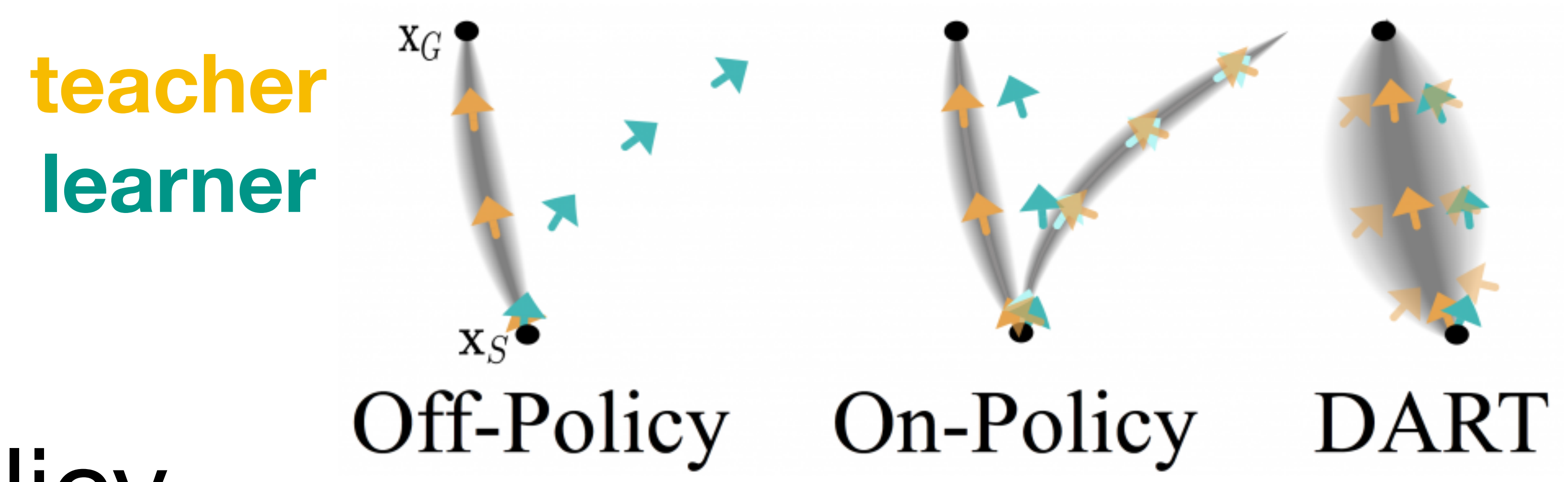
Goal-conditioned Behavior Cloning

- Can we train one policy to reach multiple goals? $\pi_\theta(a_t \mid s_t, g)$
  - Assume goal = state that the agent should reach
- How can we know the goal in demonstrations $\xi = s_0, a_0, s_1, a_1, \ldots$?
  - Require manual labeling?
- Hindsight: take each s_t as the goal of the trajectory leading to it

$$s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t = g$$

  - Supervised learning of $\pi(a \mid s, g)$ from data points $(s_t, a_t, s_{t'})$ for $t' > t$

DART: Disturbances Augmenting Robot Training



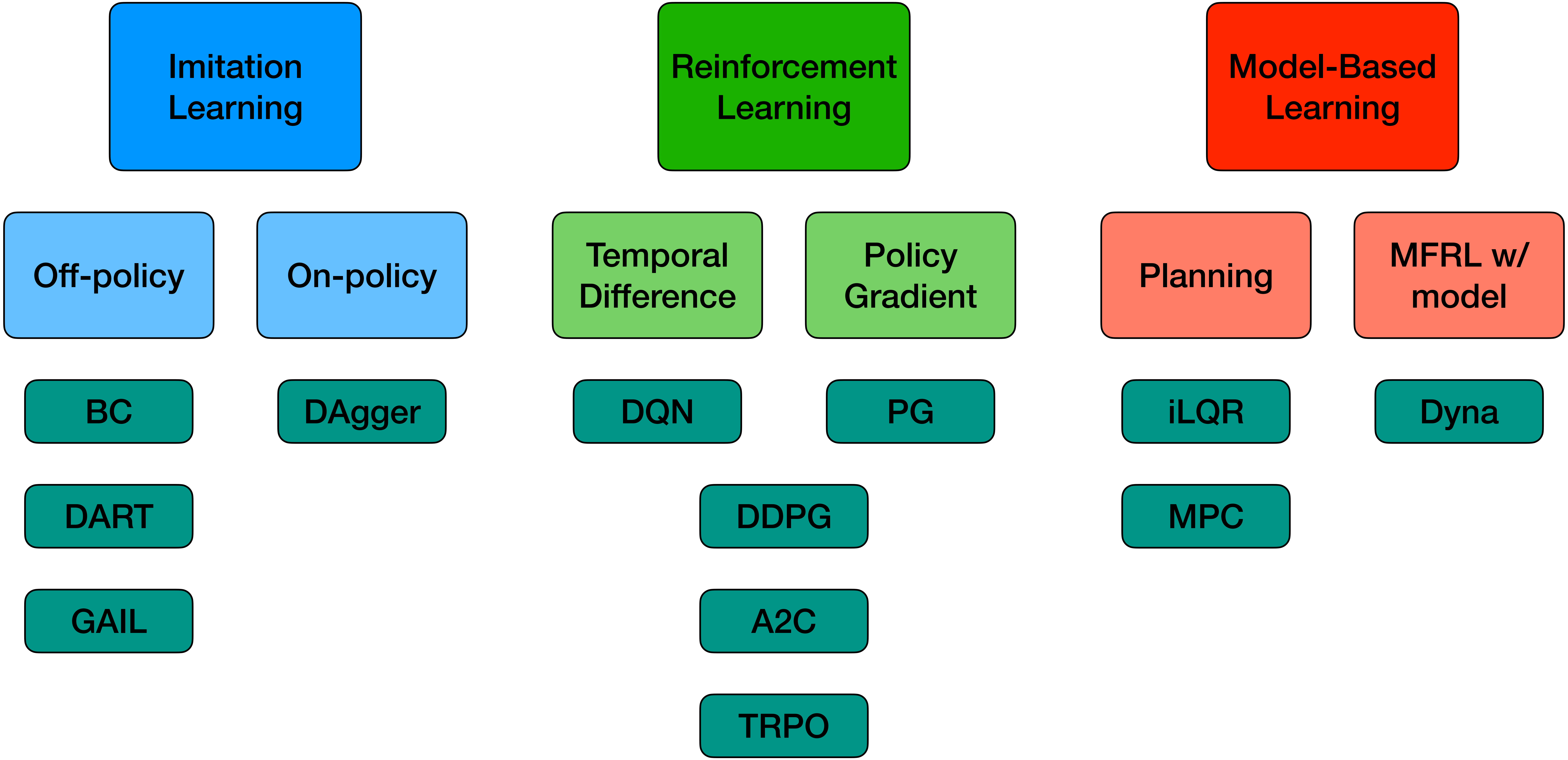
- Off-policy vs. on-policy
  - On-policy = data comes from the learner's current policy
  - Off-policy = data comes from another policy (another agent or past learner)
- In off-policy IL (e.g. BC) learner may go off the teacher's support
- In on-policy IL (e.g. DAgger) learner initially goes off, until corrected
- DART: increase the data support by injecting noise during demonstrations
  - Force teacher into slight-error states, to see how they are fixed

Image: Laskey et al. 2017

DART

- Pros:
  - Addresses covariate shift (to a point)
  - Safe (to a point)
  - Some theoretical guarantees
- Cons:
  - Burdens human teacher: how to provide good action under noise
  - Theoretically optimal noise level may be impossible for human, unsafe

Taxonomy


Imitation Learning
Reinforcement Learning
Model-Based Learning
Off-policy
On-policy
Temporal Difference
Policy Gradient
Planning
MFRL w/ model
BC
DAgger
DQN
PG
iLQR
Dyna
DART
DDPG
MPC
GAIL
A2C
TRPO

Reinforcement Learning

- Pros:
  - Supervisor only provides rewards (“Explicit what”)
  - Very general = lots of applications
- Cons:
  - Data-inefficient
  - Much theory still unknown
  - Much practice unreproducible, many hyperparameters
  - May be unsafe

Recap: policy evaluation

	model-based	model-free
Monte Carlo (MC)	$V_\pi(s_0) = \mathbb{E}_{\xi \sim p_\pi}[R \mid s_0]$	$\xi \sim p_\pi \qquad V(s_0) \to R(\xi)$
Dynamic Programming (DP) / Temporal Difference (TD) (on-policy)	$V_\pi(s) = \mathbb{E}_{a\|s \sim \pi}[r(s,a) + \gamma \mathbb{E}_{s'\|s,a \sim p}[V_\pi(s')]]$	$s, a, r, s' \sim p_\pi \qquad V(s) \to r + \gamma V(s')$
Dynamic Programming (DP) / Temporal Difference (TD) (off-policy)	$Q_\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{\substack{s'\|s,a \sim p \\ a'\|s' \sim \pi}}[Q_\pi(s',a')]]$	$s, a, r, s' \sim p_{\pi'} \quad Q(s,a) \to r + \gamma \mathbb{E}_{a'\|s' \sim \pi}[Q(s',a')]$

Recap: policy ~~evaluation~~ improvement

	model-based	model-free
Monte Carlo (MC)	$V_\pi(s_0) = \mathbb{E}_{\xi \sim p_\pi}[R \mid s_0]$	$\xi \sim p_\pi \qquad V(s_0) \to R(\xi)$
Dynamic Programming (DP) / Temporal Difference (TD) (on-policy)	$V_\pi(s) = \overset{\max\limits_{a}}{\cancel{\mathbb{E}_{a\mid s\sim\pi}}}[r(s,a) + \gamma \mathbb{E}_{s'\mid s,a\sim p}[V_\pi(s')]]$ **Value Iteration**	$s, a, r, s' \sim p_\pi \qquad V(s) \to r + \gamma V(s')$
Dynamic Programming (DP) / Temporal Difference (TD) (off-policy)	$Q_\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{s'\mid s,a\sim p,\ \cancel{a'\mid s'\sim\pi}}[Q_\pi(s',a')]]$ $\max\limits_{a'}$	$s, a, r, s' \sim p_{\pi'} \qquad Q(s,a) \to r + \gamma \overset{\max\limits_{a'}}{\cancel{\mathbb{E}_{a'\mid s'\sim\pi}}}[Q(s',a')]$ **Q-learning**

Deep TD reinforcement learning

- Deep Q Learning (historically called DQN):

$$\mathscr{L}_\theta(s, a, r, s') = (r + \gamma \max_{a'} Q_{\bar{\theta}}(s', a') - Q_\theta(s, a))^2$$

- This algorithm should work off-policy, so we can keep past experience
  - Replay buffer = data set of recent past experience of learner policy at that time
  - Variants differ on
    - How to add experience to the buffer
    - How to sample from the buffer

Temporal Difference

- Pros:
  - Data-efficient (Dynamic Programming)
  - Off-policy
  - Biologically inspired, some evidence in neuroscience (dopamine $\approx$ Bellman error)
- Cons:
  - Learns value function, not a policy
  - Much is unknown about optimal exploration

Policy Gradient (PG)

- Unlike minimizing $\mathscr{L}_\theta(\mathscr{D})$ in general ML, in RL we maximize $\mathscr{J}_\theta = \mathbb{E}_{\xi \sim p_{\pi_\theta}}[R]$
- This is harder since the “data” distribution depends on θ
- But there's a trick: $\nabla_\theta \log p_\theta(\xi) = \dfrac{1}{p_\theta(\xi)} \nabla_\theta p_\theta(\xi)$
- Log-derivative / score-function / REINFORCE trick: estimate gradient using samples of $p_\theta(\xi)$

$$
\begin{aligned}
\nabla_\theta \mathscr{J}_\theta &= \nabla \theta \int d\xi p_\theta(\xi) R(\xi) \\
&= \int d\xi p_\theta(\xi) \nabla_\theta \log p_\theta(\xi) R(\xi) \\
&= \mathbb{E}_{\xi \sim p_\theta}[\nabla_\theta \log p_\theta(\xi) R]
\end{aligned}
$$

REINFORCE (1992 !)

- Roll out π_θ to sample $\xi \sim p_\theta$
- Compute $R(\xi)$ and

$$\nabla_\theta \log p_\theta(\xi) = \nabla_\theta(\log p(s_0) + \sum_t (\log \pi_\theta(a_t \mid s_t) + \log p(s_{t+1} \mid s_t, a_t)))$$

- Take a gradient step with $\nabla_\theta \log p_\theta(\xi) R$
- Repeat

- This is model-free! but on-policy, + high variance of the gradient estimator

Baselines

- Constant shifts in return shouldn't matter for optimal policy

$$0 = \nabla_\theta \mathbb{E}_{\xi \sim p_\theta}[b] = \mathbb{E}_{\xi \sim p_\theta}[\nabla_\theta \log p_\theta(\xi) b]$$

- Can we use that to reduce variance without adding bias?

- Using the average return works pretty well in practice

$$\nabla_\theta \mathcal{J}_\theta \approx \frac{1}{N} \sum_i \nabla_\theta \log p_\theta(\xi_i)(R_i - b)$$

- With $b = \frac{1}{N} \sum_i R_i$

Don't let the past distract you

$$\nabla_\theta \mathcal{J}_\theta = \mathbb{E}_{\xi \sim p_\theta}\left[\left(\sum_t \nabla_\theta \log \pi_\theta(a_t | s_t)\right) R\right] = \sum_t \mathbb{E}_{s_t \sim p_\theta}[\nabla_\theta \mathbb{E}_{a_t | s_t \sim \pi_\theta}[R]]$$

- In our case, $R_{\geq t} = \sum_{t' \geq t} \gamma^{t'} r(s_{t'}, a_{t'})$ is a sufficient statistic of R for $\mathcal{J}_\theta$
- Therefore, a lower-variance gradient estimator:

$$\sum_t \gamma^t \mathbb{E}_{s_t \sim p_\theta}[\mathbb{E}_{a_t | s_t \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a_t | s_t) R_{\geq t}]]$$

Policy-Gradient Theorem

$$\nabla_\theta \mathcal{J}_\theta = \sum_t \gamma^t \mathbb{E}_{s_t \sim p_\theta}[\mathbb{E}_{a_t|s_t \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t) R_{\geq t}]]$$

$$\stackrel{!}{=} \sum_t \gamma^t \mathbb{E}_{s_t \sim p_\theta}[\mathbb{E}_{a_t|s_t \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t) Q_{\pi_\theta}(s_t, a_t)]]$$

$$\nabla_\theta V_{\pi_\theta}(s) = \nabla_\theta \mathbb{E}_{a|s \sim \pi_\theta}[Q_{\pi_\theta}(s, a)]$$

$$= \sum_a (\nabla_\theta \pi_\theta(a|s) Q_{\pi_\theta}(s, a) + \pi_\theta(a|s) \nabla_\theta Q_{\pi_\theta}(s, a))$$

$$= \mathbb{E}_{a|s \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q_{\pi_\theta}(s, a) + \nabla_\theta(r(s, a) + \gamma \mathbb{E}_{s'|s,a \sim p}[V_{\pi_\theta}(s')])]$$

$$= \mathbb{E}_{a|s \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q_{\pi_\theta}(s, a) + \gamma \mathbb{E}_{s'|s,a}[\nabla_\theta V_{\pi_\theta}(s')]]$$

- Here back-propagating gradients is like a Bellman recursion

DDPG

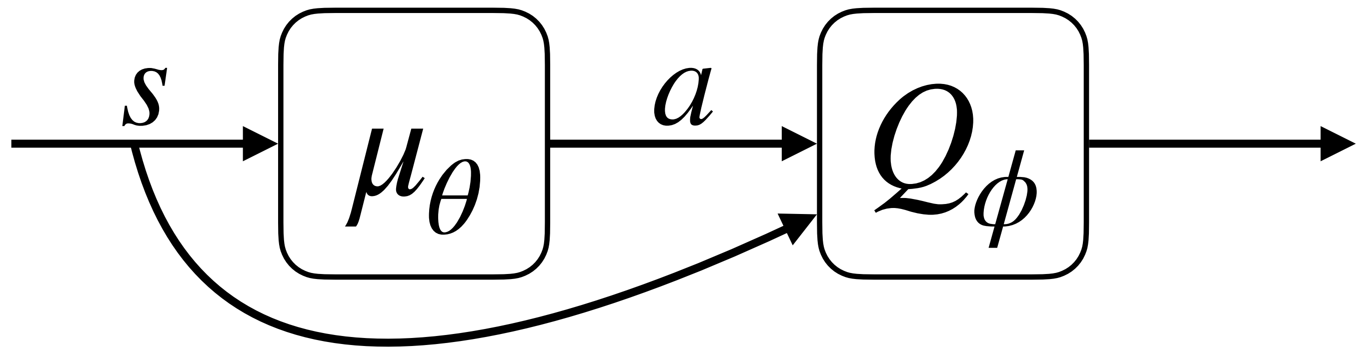
- Previous methods: represent a Q maximizer or train one ad-hoc
- More general method: let a deterministic $\mu_\theta(s)$ learn to maximize $Q_\phi(s, a)$
  - This makes it an Actor–Critic method; critic trained with any TD method
- Policy Gradient Theorem:

$$\nabla_\theta \mathcal{J}_\theta = \mathbb{E}_{s,a \sim p_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q_{\pi_\theta}(s, a)]$$

- Deterministic Policy Gradient Theorem:

$$\nabla_\theta \mathcal{J}_\theta = \mathbb{E}_{s \sim p_\theta}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q_{\mu_\theta}(s, a)\Big|_{a=\mu_\theta(s)}\right]$$

$\leftarrow \nabla_\theta \mu_\theta(s) \quad \nabla_a Q_\phi(s, a) \leftarrow$



Baselines

$$\nabla_\theta \mathcal{J}_\theta = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s)(Q_{\pi_\theta}(s,a) - b) | s, a]$$

- b can be any variable independent of a given s
  - Can depend on the past, but not the future
- Previously, we used the average as baseline $b = \frac{1}{N}\sum_i R_i$
- This suggests using the state value as baseline $b = V_{\pi_\theta}(s)$

Advantage estimation

$$\nabla_\theta \mathcal{J}_\theta = \mathbb{E}[\nabla_\theta \log \pi_\theta(a \mid s) A_{\pi_\theta}(s, a) \mid s, a]$$

- How to estimate $A_{\pi_\theta}(s, a)$?

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s) = r(s, a) + \gamma \mathbb{E}_{s' \mid s, a \sim p}[V_\pi(s')] - V_\pi(s)$$

- With value estimation $V_\phi(s)$, estimate the advantage:

$$\hat{A}(s, a) \approx r + \gamma V_\phi(s') - V_\phi(s)$$

An Actor–Critic algorithm

Algorithm 1 Actor–Critic

get on-policy sample (s, a, r, s')
take gradient step on $\mathcal{L}_\phi = (r + \gamma V_{\bar{\phi}}(s') - V_\phi(s))^2$
compute $\hat{A}(s, a) = r + \gamma V_\phi(s') - V_\phi(s)$
take gradient step $\nabla_\theta \log \pi_\theta(a|s) \hat{A}(s, a)$
repeat

Policy Gradient algorithms

- REINFORCE: $\nabla_\theta \mathcal{J}_\theta \approx \nabla_\theta \log \pi_\theta(a_t | s_t) R$
- Baselines (= control variates): $\nabla_\theta \mathcal{J}_\theta \approx \nabla_\theta \log \pi_\theta(a_t | s_t)(R - b)$
- Future return (Rao–Blackwell): $\nabla_\theta \mathcal{J}_\theta \approx \nabla_\theta \log \pi_\theta(a_t | s_t) R_{\geq t}$
- Policy Gradient Theorem: $\nabla_\theta \mathcal{J}_\theta \approx \left(\nabla_\theta \log \pi_\theta(a_t | s_t)\right) Q_\theta(s_t, a_t)$
- Actor–Critic: $\nabla_\theta \mathcal{J}_\theta \approx \nabla_\theta \log \pi_\theta(a_t | s_t) \hat{A}(s_t, a_t)$
  - Algorithms differ by how they estimate advantages

Comparing advantage estimators

	bias	variance
$\nabla_\theta \mathcal{J}_\theta \approx \nabla_\theta \log \pi_\theta(a\,|\,s)\left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} - b\right)$	none	high one grad per traj
$\nabla_\theta \mathcal{J}_\theta \approx \nabla_\theta \log \pi_\theta(a\,|\,s)(r + \gamma V_\phi(s') - V_\phi(s))$	some approx value	lower
$\nabla_\theta \mathcal{J}_\theta \approx \nabla_\theta \log \pi_\theta(a\,|\,s)\left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} - V_\phi(s)\right)$	none	mid state-dependent baseline

n-step TD

- 1-step TD: $\hat{A}_t^1 = r_t + \gamma V(s_{t+1}) - V(s_t)$
- 2-step TD: $\hat{A}_t^2 = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t)$

...

- n-step TD: $\hat{A}_t^n = r_t + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n}) - V(s_t)$
- In the limit (MC): $\hat{A}_t^\infty = r_t + \gamma r_{t+1} + \cdots - V(s_t)$

Generalized Advantage Estimation (GAE(λ))

$$\nabla_\theta \mathcal{J}_\theta \approx \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{t'} (\lambda\gamma)^{t'} \hat{A}^1_{t+t'}$$

Generalized Advantage Estimation (GAE(λ))

$$\nabla_\theta \mathcal{J}_\theta \approx \nabla_\theta \log \pi_\theta(a_t | s_t) \sum_{t'} (\lambda\gamma)^{t'} \hat{A}^1_{t+t'}$$

$$\hat{A}^1_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

- GAE(0) = 1-step; GAE(1) = MC

Off-policy MC advantage estimation

- MC advantage estimation: $\sum_t \gamma^t r(s_t, a_t) - V_{\pi_\theta}(s)$

$$= \sum_t \gamma^t (r(s_t, a_t) + \gamma V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s_t)) = \sum_t \gamma^t \hat{A}^1_{\pi_\theta}(s_t, a_t)$$

- What if estimate the advantage of our current π_θ under a proposed new policy $\pi_{\theta'}$?

we want to maximize this!

$$\mathbb{E}_{\xi \sim p_{\theta'}}\left[\sum_t \gamma^t \hat{A}^1_{\pi_\theta}(s_t, a_t)\right] = \mathbb{E}_{\xi \sim p_{\theta'}}\left[\sum_t \gamma^t r(s_t, a_t) - V_{\pi_\theta}(s_0)\right] = \mathcal{J}_{\theta'} - \mathcal{J}_\theta$$

$$= \sum_t \gamma^t \mathbb{E}_{s_t, a_t \sim p_{\theta'}}[\hat{A}^1_{\pi_\theta}(s_t, a_t)]$$

$$= \sum_t \gamma^t \mathbb{E}_{s_t \sim p_{\theta'}}\left[\mathbb{E}_{a_t | s_t \sim \pi_\theta}\left[\frac{\pi_{\theta'}(a_t | s_t)}{\pi_\theta(a_t | s_t)} \hat{A}^1_{\pi_\theta}(s_t, a_t)\right]\right]$$

  - We can't estimate this empirically, because of $s_t \sim p_{\theta'}$; what's the difference if we just use $s_t \sim p_\theta$?

Trust-Region Policy Optimization (TRPO)

$$\max_{\theta'} \sum_t \gamma^t \mathbb{E}_{s_t \sim p_\theta} \left[\mathbb{E}_{a_t|s_t \sim \pi_\theta} \left[\frac{\pi_{\theta'}(a_t \,|\, s_t)}{\pi_\theta(a_t \,|\, s_t)} \hat{A}^1_{\pi_\theta}(s_t, a_t) \right] \right]$$

s.t. $\mathbb{D}[\pi_{\theta'} \| \pi_\theta] \leq \epsilon$

- For small enough ϵ, the objective is close to $\mathcal{J}_{\theta'} - \mathcal{J}_\theta$
  - Guarantees improvement of our objective; in practice, good ϵ is too large
- TRPO loss:

importance weight

advantage estimate

Lagrange multiplier for KL constraint

$$\mathcal{L}_\theta(s, a, r, s') = -\frac{\pi_\theta(a \,|\, s)}{\pi_{\bar{\theta}}(a \,|\, s)}(r + \gamma V_\phi(s') - V_\phi(s)) + \lambda \mathbb{D}[\pi_\theta(\,\cdot\,|\,s) \| \pi_{\bar{\theta}}(\,\cdot\,|\,s)]$$

- The actual algorithm is more complicated; simpler variant: PPO

Taxonomy

- Imitation Learning
  - Off-policy
    - BC
    - DART
    - GAIL
  - On-policy
    - DAgger
- Reinforcement Learning
  - Temporal Difference
    - DQN
  - Policy Gradient
    - PG
  - DDPG
  - A2C
  - TRPO
- Model-Based Learning
  - Planning
    - iLQR
    - MPC
  - MFRL w/ model
    - Dyna

Iterative LQR (iLQR)

Algorithm 1 iLQR

linearize dynamics around current trajectory $(\hat{x}, \hat{u})$

compute $A, B \leftarrow \nabla_x \hat{f}_t, \nabla_u \hat{f}_t$

quadratic cost approximation around $(\hat{x}, \hat{u})$

compute $Q, R, N, q, r \leftarrow \nabla_x^2 \hat{c}_t, \nabla_u^2 \hat{c}_t, \nabla_{xu} \hat{c}_t, \nabla_x \hat{c}_t, \nabla_u \hat{c}_t$

$\hat{L}_t, \hat{\ell}_t \leftarrow$ LQR on $\delta x_t = x_t - \hat{x}_t$, $\delta u_t = u_t - \hat{u}_t$ ← solve with LQR

$\delta x^*, \delta u^* \leftarrow$ execute policy $\delta u_t = \hat{L}_t \delta x_t + \hat{\ell}_t$ in the simulator / environment

$\hat{x} \leftarrow \hat{x} + \delta x^*$, $\hat{u} \leftarrow \hat{u} + \delta u^*$

roll out to get new trajectory $(\hat{x}, \hat{u})$

repeat to convergence

Model-free RL with a model

- General scheme for using a model for model-free RL:

$$
\begin{aligned}
&\text{collect data} \\
&\text{train model } \hat{p}, \hat{r} \\
&\textbf{repeat} \\
&\quad \text{sample } s \text{ from the replay buffer} \\
&\quad \text{sample } a|s \text{ from the learner's policy (or anything else)} \\
&\quad \text{simulate } r = \hat{r}(s,a) \text{ and } s'|s,a \sim \hat{p} \\
&\quad \text{perform model-free RL with } (s,a,r,s')
\end{aligned}
$$

interaction with environment (random policy) (collect data)

supervised learning (train model)

seeded by initial interaction
may interact more as learner improves (replay buffer)

use model as simulator (simulate)

- Benefit: get diverse off-policy s, and fresh on-policy a

Issues with approximate models

- Model inaccuracy accumulates
  - If $|p_\phi(s'|s,a) - p(s'|s,a)|_1 \leq \epsilon$ then $|p_\phi(s_t) - p(s_t)|_1 \leq \epsilon t$
  - We have to plan far enough ahead to realize the consequences of actions
  - But we don't have to execute those plans far ahead!
- Model Predictive Control (MPC):

$\mathcal{D} \leftarrow$ collect data
repeat
    $\hat{\mathcal{M}} \leftarrow$ train model $\hat{p}, \hat{r}$ from $\mathcal{D}$
    repeat
        $\pi \leftarrow$ plan in $\hat{\mathcal{M}}$ from current state s to horizon H
        take *one action* a according to π
        add empirical (s, a, r, s') to $\mathcal{D}$

Taxonomy

- Imitation Learning
  - Off-policy
    - BC
    - DART
    - GAIL
  - On-policy
    - DAgger
- Reinforcement Learning
  - Temporal Difference
    - DQN
  - Policy Gradient
    - PG
  - DDPG
  - A2C
  - TRPO
- Model-Based Learning
  - Planning
    - iLQR
    - MPC
  - MFRL w/ model
    - Dyna