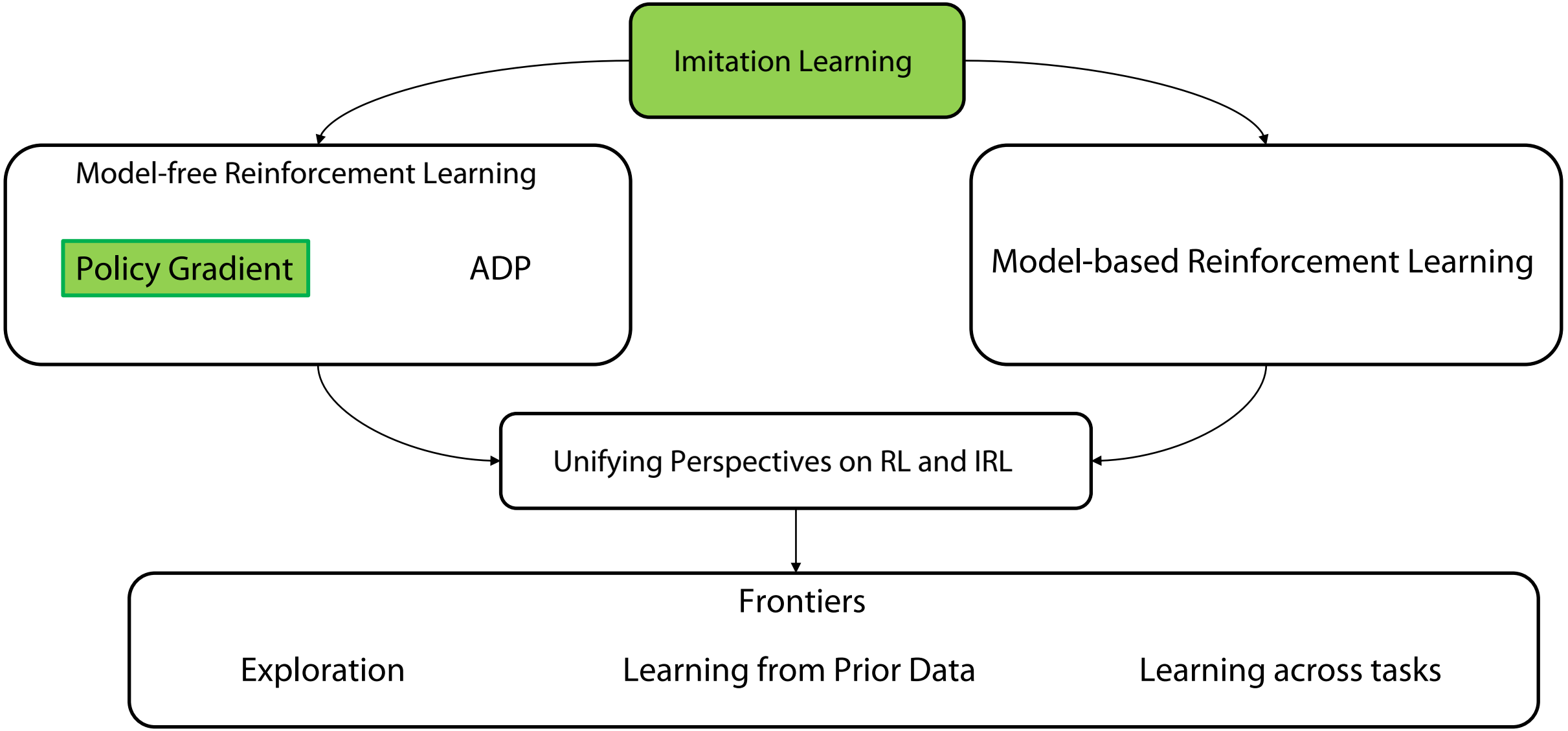
Reinforcement Learning
Spring 2026

Abhishek Gupta

TA: Mateo Guaman Castro

Class Structure



Lecture outline

Recap

A Closer Look at Convergence

Frontiers of Off-Policy RL

Model-based RL

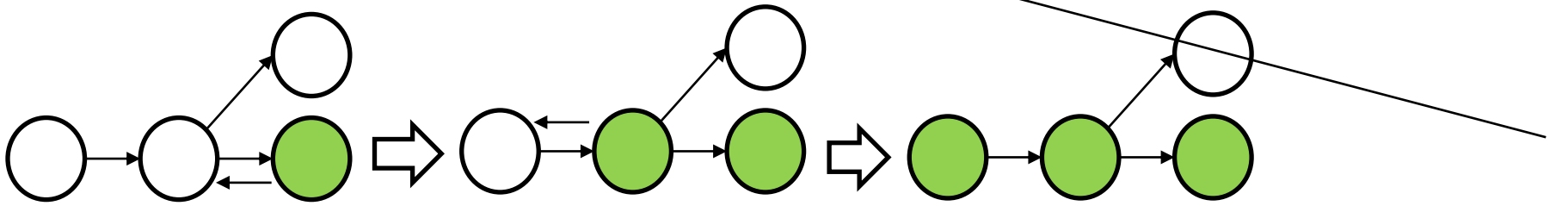
Recursive structure in Q functions directly

Q functions have special recursive structure themselves!

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s'_t, a'_t) | s_t, a_t\right]$$

$$= r(s_t, a_t) + \mathbb{E}_{\pi}\left[\sum_{t'=t+1} r(s_{t'}, a_{t'}) | s_{t+1}, a_{t+1} \sim \pi(.|s_{t+1})\right]$$

Bellman equation $$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{\substack{s_{t+1}\sim p(.|s_t,a_t) \\ a_{t+1}\sim\pi_\theta(.|s_{t+1})}}\left[Q^{\pi}(s_{t+1}, a_{t+1})\right]$$



Can be from different policies

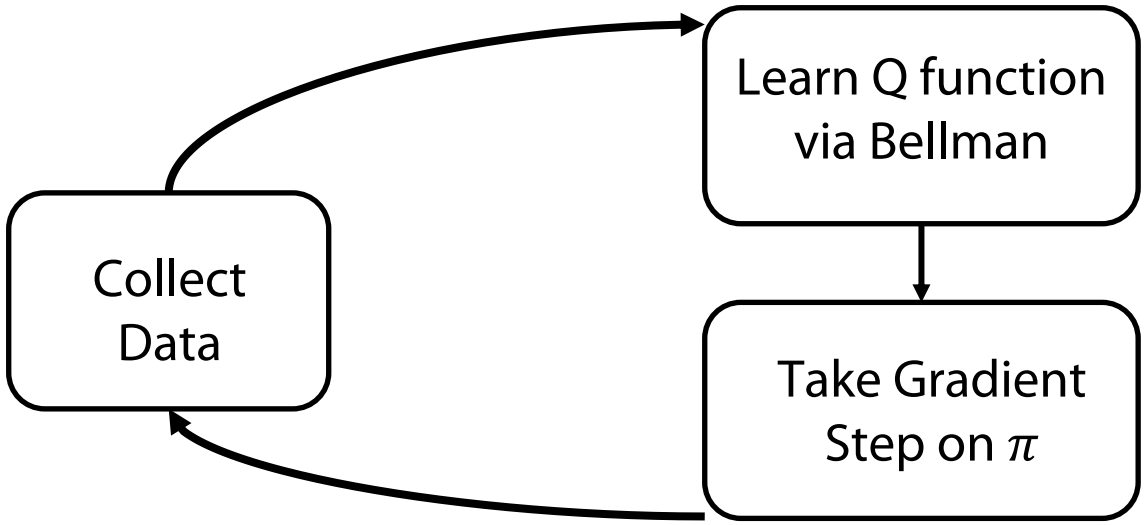
Decompose temporally via dynamic programming

Off-policy!

Actor-Critic: Policy Gradient in terms of Q functions

Critic: learned via the Bellman update (Policy Evaluation)

$$\min_{\phi} \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}} \left(Q^{\pi}_{\phi}(s_t,a_t) - \left(r(s_t,a_t) + Q^{\pi}_{\hat{\phi}}(s_{t+1},a_{t+1})\right)\right)^2 \quad a_{t+1} \sim \pi(\cdot|s_{t+1})$$

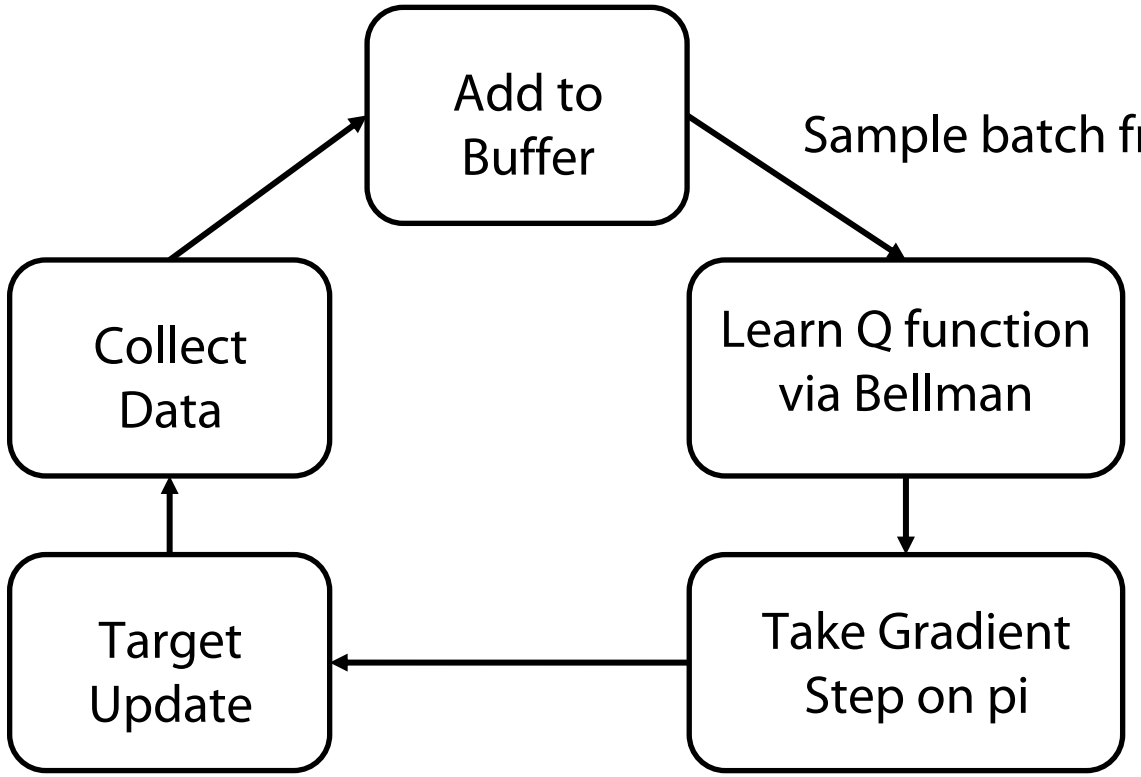


Lowers variance and is off-policy!

Actor: updated using learned critic (Policy Improvement)

$$\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}} \mathbb{E}_{a\sim\pi(.|s)} \left[Q^{\pi}(s,a)\right]$$

Targets and replay buffers



1 sample

Critic: 1 gradient step on Bellman error

Polyak Averaging

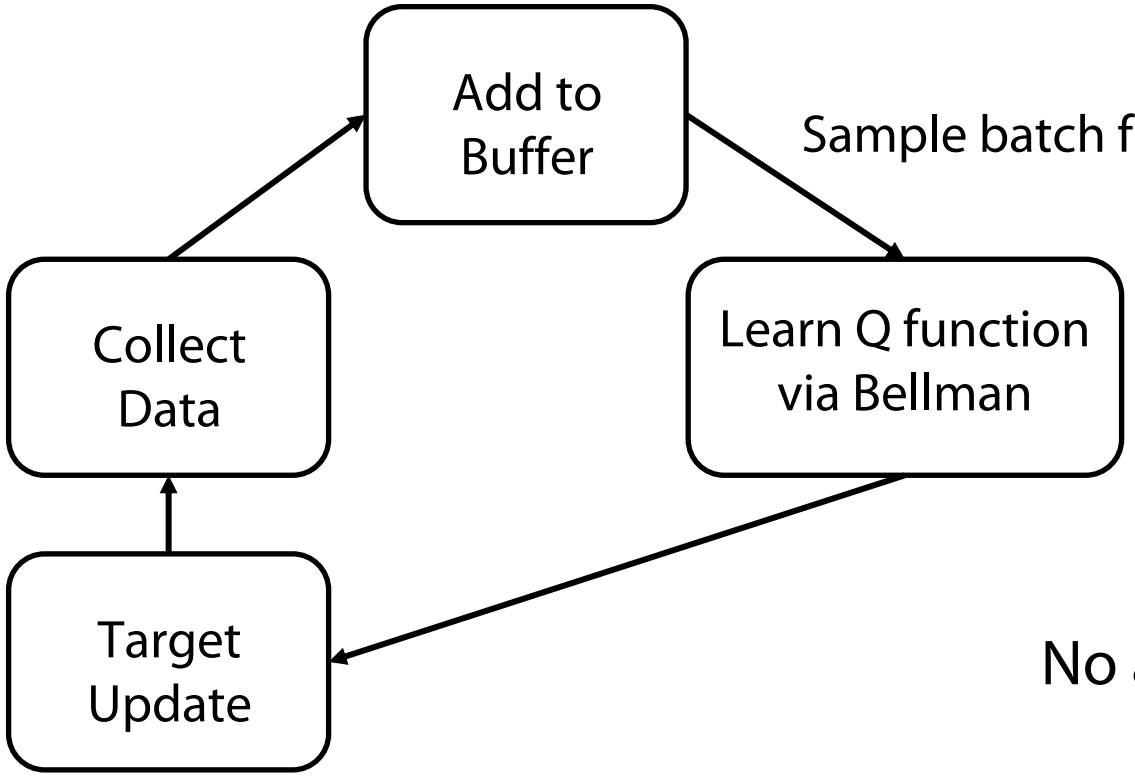
Actor: 1 gradient step on policy gradient

From actor-critic to Q-learning

$$\min_{\phi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left[Q_\phi^\pi(s_t, a_t) - \left(r(s_t, a_t) + \max_{a_{t+1}} \left[Q_{\bar{\phi}}(s_{t+1}, a_{t+1}) \right] \right) \right]^2 \right]$$

$$\pi(a|s) = \max_a Q(s, a)$$

Directly do max in the Bellman update



1 sample

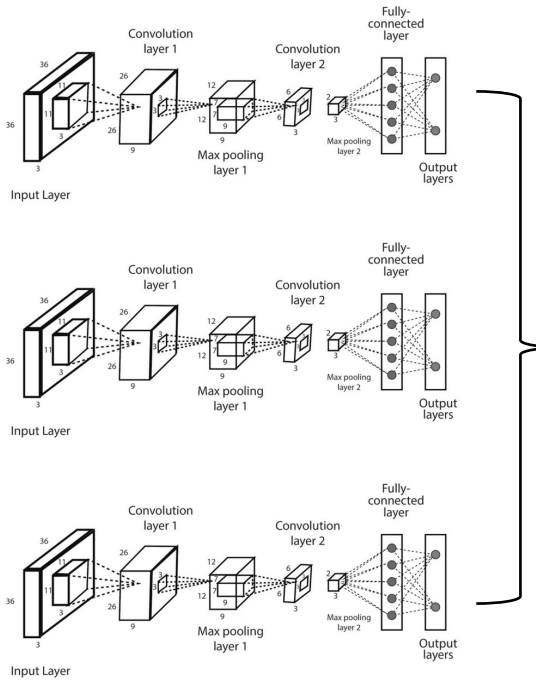
Critic: 1 gradient step on Bellman error

Polyak Averaging

No actor updates, just learn Q!

Overestimation Bias in Actor-Critic – Ensemble Q

Learn two (or N) independent measures of Q, take the minimum
→ pessimistic on random variable



Independent updates

Critic

$$y_j = r(s,a) + \gamma \min_{i=1,\dots,N} Q_{\bar{\phi}_i}(s', \pi_\theta(s'))$$

$$\min_{\phi_j} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left(Q_{\phi_j}(s,a) - y_j \right)^2 \right]$$

Actor

$$\max_{\theta_j} \mathbb{E}_{s\sim\mathcal{D}} \mathbb{E}_{a\sim\pi_{\theta_j}} \left[Q_{\phi_j}(s,a) \right]$$

Significantly improves overestimation and in turn sample efficiency!

Max-Ent Off-Policy RL

$$\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{T} \gamma^t (r(s_t, a_t) + \alpha \mathcal{H}(\pi(.|s_t)) \right]$$

Work through the recursion, same as with the regular Bellman

Critic – Policy Evaluation

$$\min_{\phi} \mathbb{E}_{\substack{(s_t, a_t, s_{t+1}) \sim \mathcal{D} \\ a_{t+1} \sim \pi(\cdot|s_{t+1})}} \left[(Q_{\phi}^{\pi}(s_t, a_t) - (r(s_t, a_t) + Q_{\hat{\phi}}^{\pi}(s_{t+1}, a_{t+1})) - \alpha \log \pi(a_{t+1}|s_{t+1}))^2 \right]$$

Actor – Policy Improvement

$$\max_{\pi} \mathbb{E}_{s \sim \mathcal{D}} \left[\mathbb{E}_{a \sim \pi(\cdot|s)} \left[Q_{\phi}^{\pi}(s, a) - \alpha \log \pi(a|s) \right] \right]$$

Lecture outline

Recap

A Closer Look at Convergence

Frontiers of Off-Policy RL

Model-based RL

Ok, so are off-policy algorithms perfect?

What makes off-policy RL hard?

Deadly triad:

1. Function Approximation
2. Bootstrapping
3. Off-policy learning

$$\min_{\phi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left[Q^{\pi}_{\phi}(s_t, a_t) - (r(s_t, a_t) + \max_{a_{t+1}} \left[Q_{\bar{\phi}}(s_{t+1}, a_{t+1}) \right]) \right]^2 \right]$$

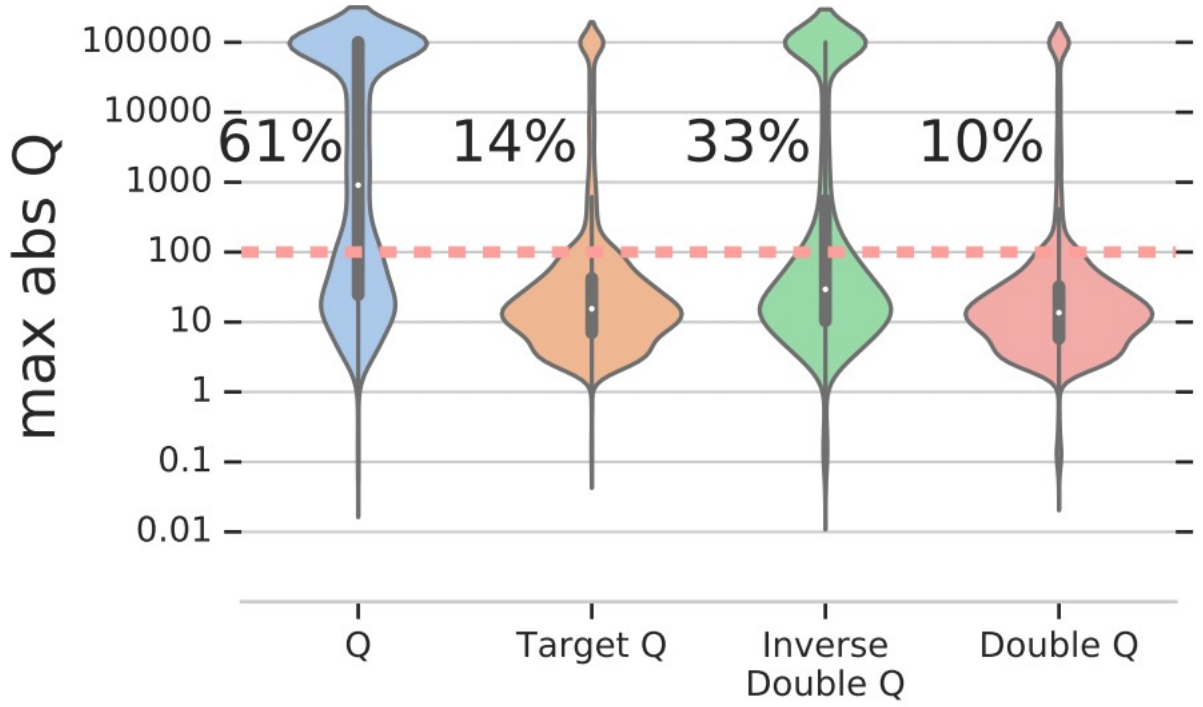
These in combination lead to many of the difficulties in stabilizing off-policy RL with function approximation

Zooming out – what makes off-policy RL hard?

Deadly triad:

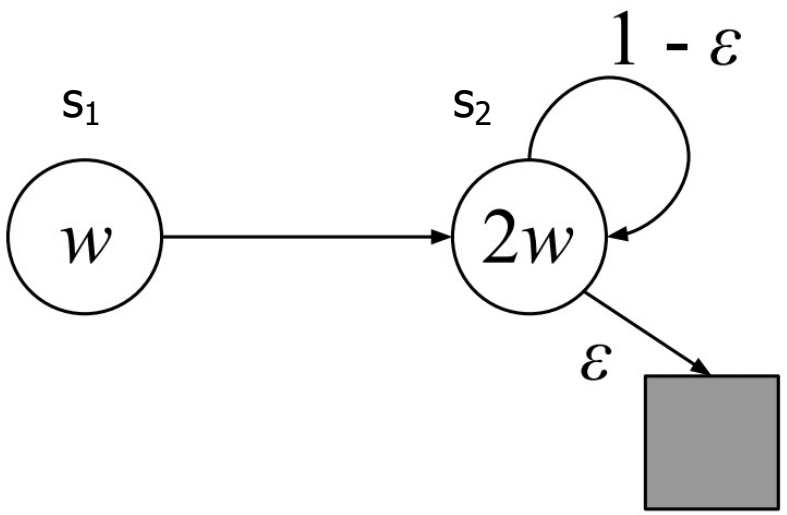
1. Function Approximation
2. Bootstrapping
3. Off-policy learning

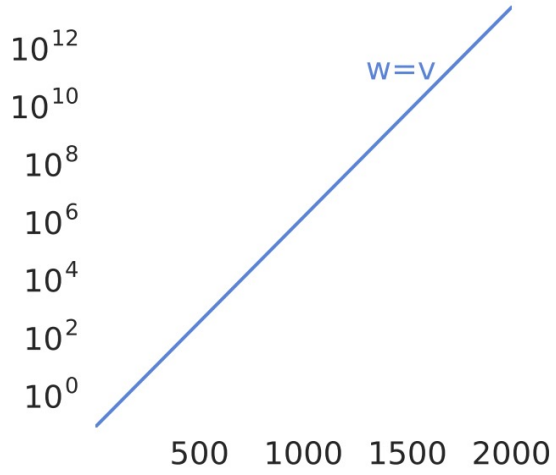
61% of runs show divergence of Q-values



Diverges even with linear function approximation, when off-policy + bootstrapping

Zooming out – what makes off-policy RL hard?





(b) $v(s) = w\phi(s)$ diverges.

Let's go to the whiteboard!

A Closer Look at the Value Bellman Equation

Bellman Update

$$\min_{\phi} \mathbb{E}_{(s_i, a_i, s_i{}') \sim \pi} \left[(V_\phi^\pi(s_i) - y_i)^2 \right]$$

$$y_i = r(s_i, a_i) + V(s_i{}')$$

Fixed-point iteration algorithm

Bellman equation

$$V^\pi(s_t) = \mathbb{E}_{\pi_\theta} \left[r(s_t, a_t) + V^\pi(s_{t+1}) \right]$$

Holds at convergence

$$x_{n+1} = f(x_n) \quad \Longrightarrow \quad x^* = f(x^*)$$

Fixed-point

Q: Does this update converge to the true value as a fixed point?

Does this converge?

Q: Does this update converge to the true value as a fixed point?

$$\min_{\phi} \mathbb{E}_{(s_i, a_i, s_i{}') \sim \pi} \left[(V_\phi^\pi(s_i) - y_i)^2 \right]$$

$$y_i = r(s_i, a_i) + V(s_i{}')$$

Banachs fixed point theorem

Let (M, d) be a complete metric space.

Let $f : M \to M$ be a contraction.

That is, there exists $q \in [0 \,..\, 1)$ such that for all $x, y \in M$:

$$d(f(x), f(y)) \leq q\, d(x, y)$$

Then there exists a unique fixed point of f.

Let's consider a simple version of this algorithm

$$V_{i+1}^\pi(s) \leftarrow \mathbb{E}_{\substack{s' \sim p(.|s,a) \\ a \sim \pi(\cdot|s')}} \left[r(s, a) + V_i^\pi(s') \right]$$

$$V_{i+1} \leftarrow B_p^\pi V_i^\pi$$

↑

Prove this is a contraction

Does this converge?

$$V_{i+1}^{\pi}(s) \leftarrow \mathbb{E}_{\substack{s' \sim p(.|s,a) \\ a \sim \pi(\cdot|s')}} [r(s,a) + V_i^{\pi}(s')]$$

$$V_{i+1} \leftarrow B_p^{\pi} V_i^{\pi}$$

Bellman operator

To prove: $d(f(x), f(y)) \leq q\, d(x,y)$

inf-norm — Value functions

$$V_{i+1} \leftarrow B_p^{\pi} V_i^{\pi} \qquad U_{i+1} \leftarrow B_p^{\pi} U_i^{\pi}$$

$$|V_{i+1} - U_{i+1}|_{\infty} \leq \gamma |V_i - U_i|_{\infty}$$

$$|V_{i+1} - U_{i+1}|_{\infty} = \max_s |V_{i+1}(s) - U_{i+1}(s)|$$

$$= \max_s \left| \left(\int \pi(a|s) \left(\int p(s'|s,a)(r(s,a) + \gamma U_i(s'))ds \right) da \right) - \left(\int \pi(a|s) \left(\int p(s'|s,a)(r(s,a) + \gamma V_i(s'))ds \right) da \right) \right|$$

$$= \gamma \max_s \left| \left(\int \pi(a|s) \left(\int p(s'|s,a)(U_i(s') - V_i(s'))ds \right) da \right) \right|$$

$$\leq \gamma \max_s \left| \left(\int \pi(a|s) \left(\int p(s'|s,a) \max_x (U_i(x) - V_i(x))ds \right) da \right) \right|$$

$$= \gamma \max_s \left| \left(\int \pi(a|s) \max_x (U_i(x) - V_i(x) da \right) \right|$$

$$= \gamma \max_x |U_i(x) - V_i(x)| = \gamma |U_i - V_i|_{\infty}$$

Contraction, hence converges to a fixed point

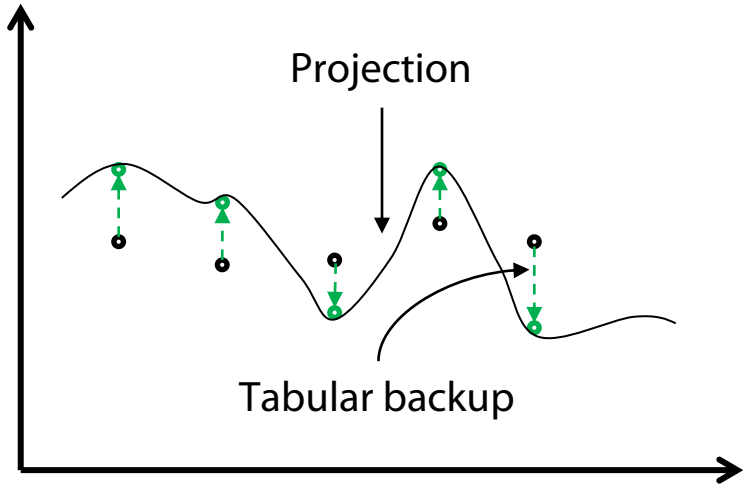
Does this converge for arbitrary function approximation?

For arbitrary function approximation, it is not just a Bellman backup

$$V_{i+1}^{\pi}(s) \leftarrow \mathbb{E}_{\substack{s' \sim p(.|s,a) \\ a \sim \pi(\cdot|s')}} [r(s,a) + V_i^{\pi}(s')] \qquad V_{i+1} \leftarrow B_p^{\pi} V_i^{\pi}$$

We perform a Bellman backup + a projection

Projection – find closest element of function class to approximate tabular values



$$\min_{\phi} \mathbb{E}_{(s_i, a_i, s_i{}') \sim \pi} \left[(V_{\phi}^{\pi}(s_i) - y_i)^2 \right]$$

$$y_i = r(s_i, a_i) + V(s_i{}')$$

Backup may be a contraction, but backup + projection may not be

Lecture outline

Recap

A Closer Look at Convergence

Frontiers of Off-Policy RL

Model-based RL

What should I work on?

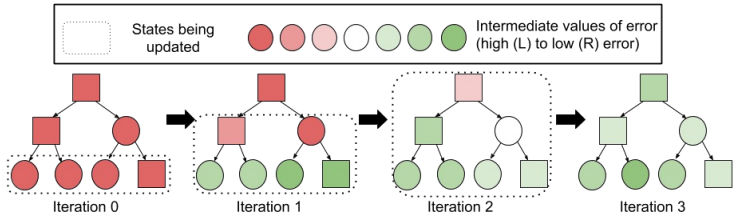
Where does the frontier of off-policy RL lie?

Off-policy is an extremely promising tool, but not quite plug and play like PG methods

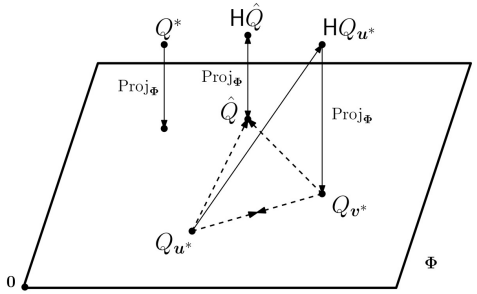
- Low variance, off-policy, avoids reconstruction, performs dynamic programming
- Has the potential to be **performant** and **sample efficient**

But in practice is often unstable, inefficient with high dimensional observations

Sampling



Theory



Exploration

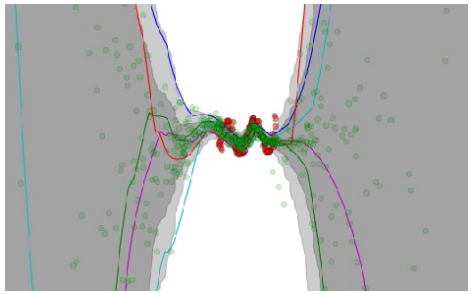
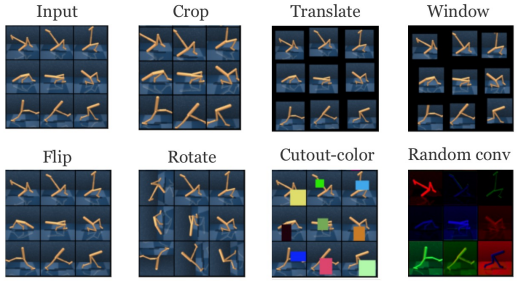
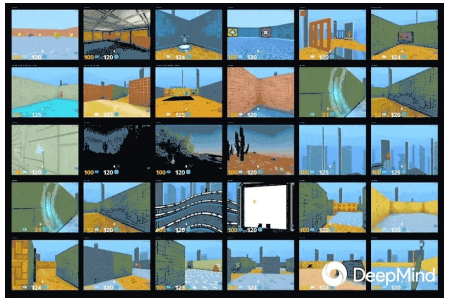


Image-based RL

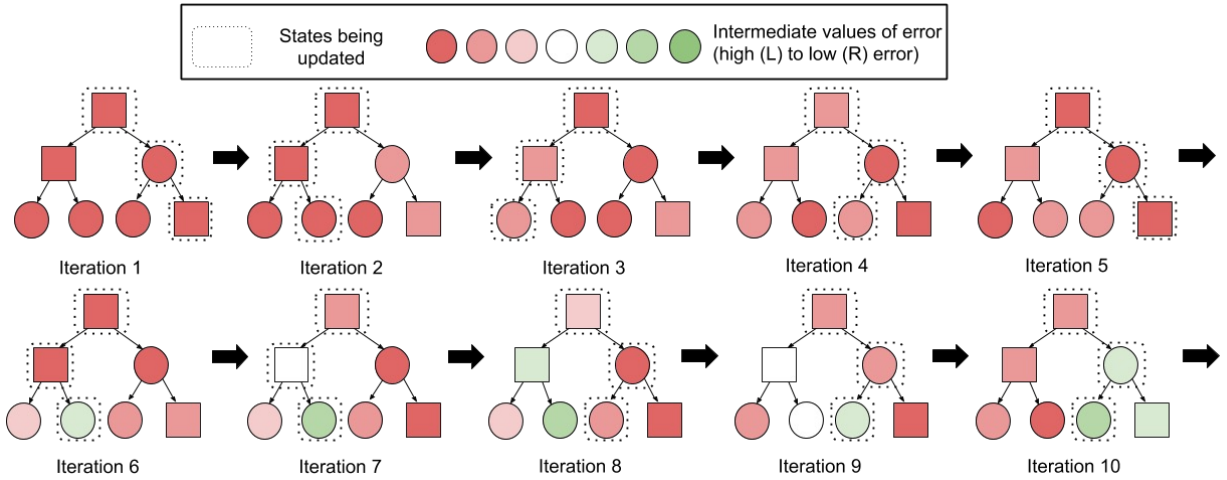


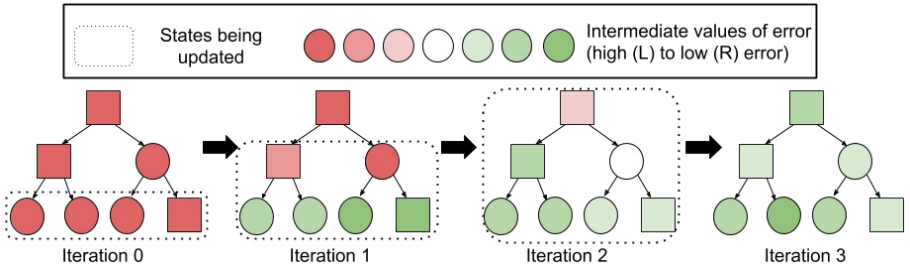
Partial Observability

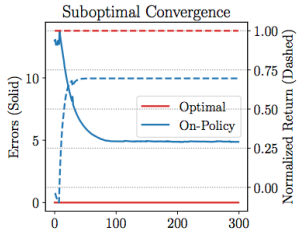


Prioritizing Experience

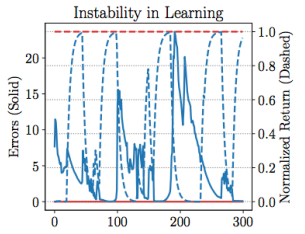
Performing uniform buffer TD updates can be catastrophically bad



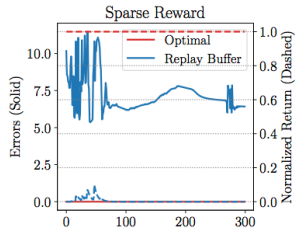




(a) Sub-optimal convergence for on-policy distributions: return (dashed) and value error (solid). Note that value error decreases rapidly at the start and finally converges to a nonzero value, leading to sub-optimal return.



(b) Instability for replay buffer distributions: return (dashed) and value error (solid) over training iterations. Note the rapid increase in value error at multiple points, which co-occurs with instabilities in returns.



(c) Error (left) and returns (right) for sparse reward MDP with replay buffer distributions. Note the inability to learn, low return, and highly unstable value error $\mathcal{E}_k$, often increasing sharply, destabilizing the learning process.

Need to prioritize updates to propagate good values

Theory/Convergence with Function Approximation

Significant body of work on learning dynamics with function approximation

Delusional Bias

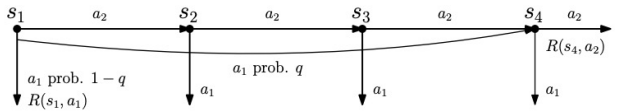


Figure 1: A simple MDP that illustrates delusional bias (see text for details).

Implicit regularization

$$\overline{\mathcal{R}}_{\text{exp}}(\theta) = \sum_{i \in \mathcal{D}} \phi(\mathbf{s}_i, \mathbf{a}_i)^\top \phi(\mathbf{s}'_i, \mathbf{a}'_i).$$

Bilinear classes

Framework	B-Rank	B-Complete	W-Rank	Bilinear Class (this work)
B-Rank	✓	✗	✓	✓
B-Complete	✗	✓	✗	✓
W-Rank	✗	✗	✓	✓
Bilinear Class (this work)	✗	✗	✗	✓

Practically Performant Optimization for Off-Policy RL

Well-conditioned deep RL can be hugely sample efficient and stable

XQC: Well-conditioned Optimization Accelerates Deep Reinforcement Learning

Daniel Palenicek[1,2] Florian Vogt[3] Joe V
[1]Technical University of Darmstadt [2]hessi
[5]German Research Center for AI (DFKI)
daniel.palenicek@tu-darmstad

FlashSAC: Fast and Stable Off-Policy Reinforcement Learning for High-Dimensional Robot Control

Holiday Robotics

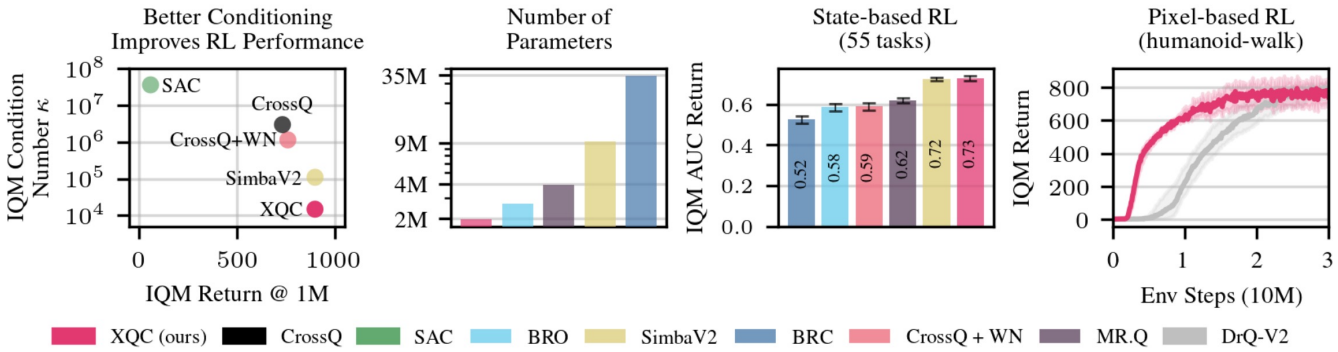
Donghu Kim[1,*] Youngdo Lee[2,3,*] Minho Park[2] Kinam Kim[2] I Made Aswin Nahendra[3] Takuma Seno[4]
Sehee Min[1] Daniel Palenicek[5,6] Florian Vogt[7] Danica Kragic[7] Jan Peters[5,6,8,9] Jaegul Choo[2,†] Hojoon Lee[1,†]

Hyperspherical Normalization for Scalable Deep Reinforcement Learning

Hojoon Lee[1*] Youngdo Lee[1*] Takuma Seno[2] Donghu Kim[1] Peter Stone[2 3] Jaegul Choo[1]

FlashSAC vs **PPO**

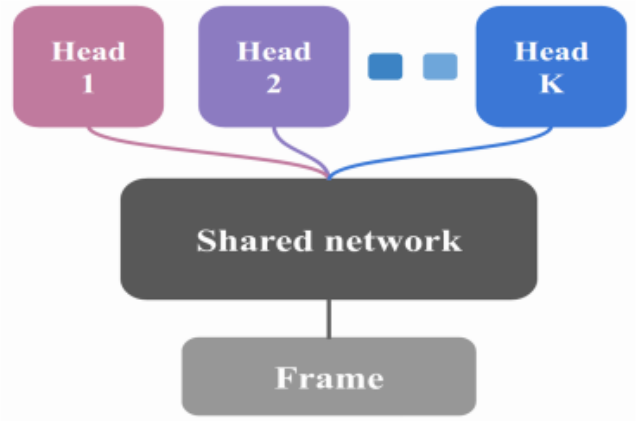
Training Unitree G1 to Walk



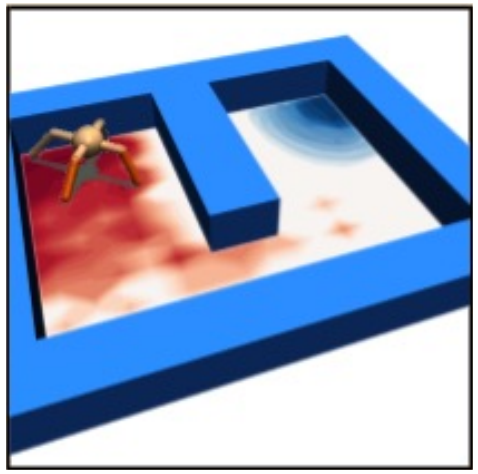
Exploration in Off-Policy RL

Better exploration methods

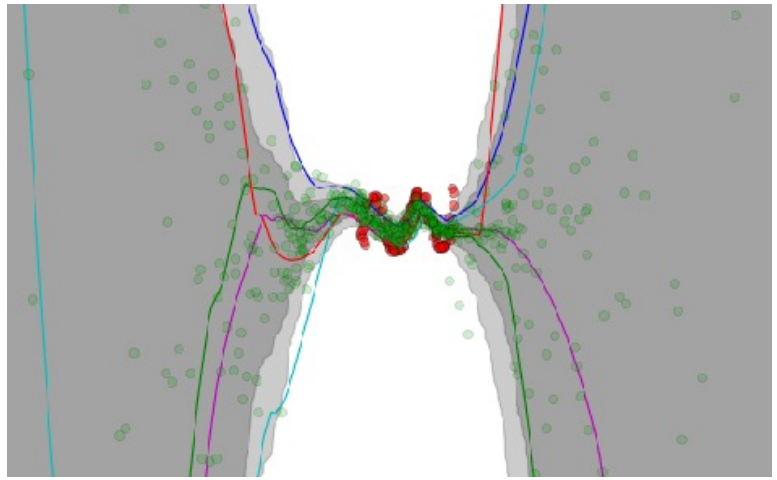
Uncertainty based methods



Count-based methods



Information gain methods

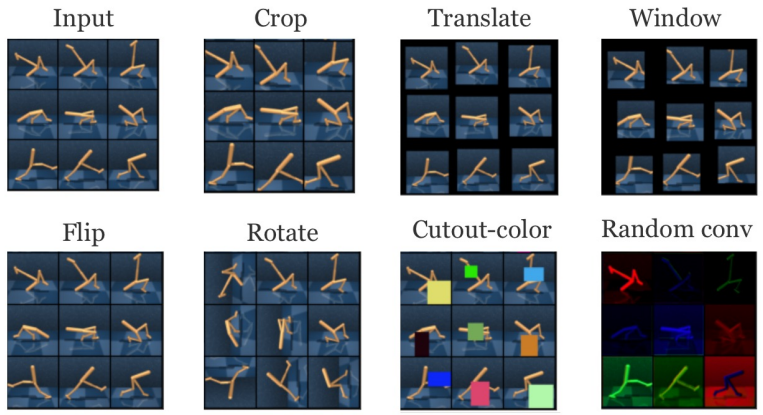


Often critical for getting algorithms to work!

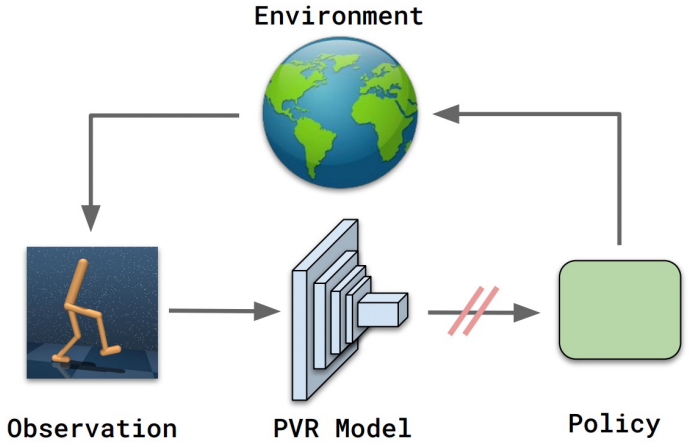
Image-based Off-Policy RL

Learning from high dimensional observations is unstable – images/point clouds

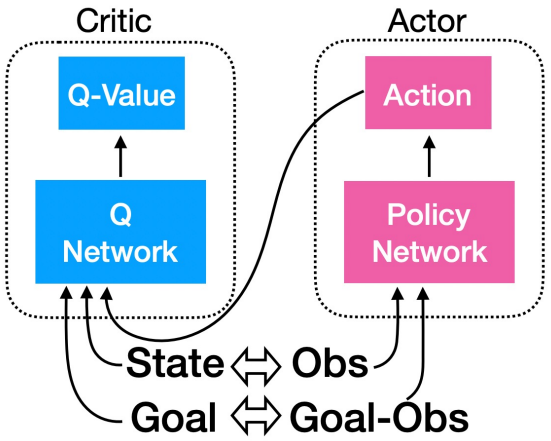
Data augmentations



Pre-trained representations



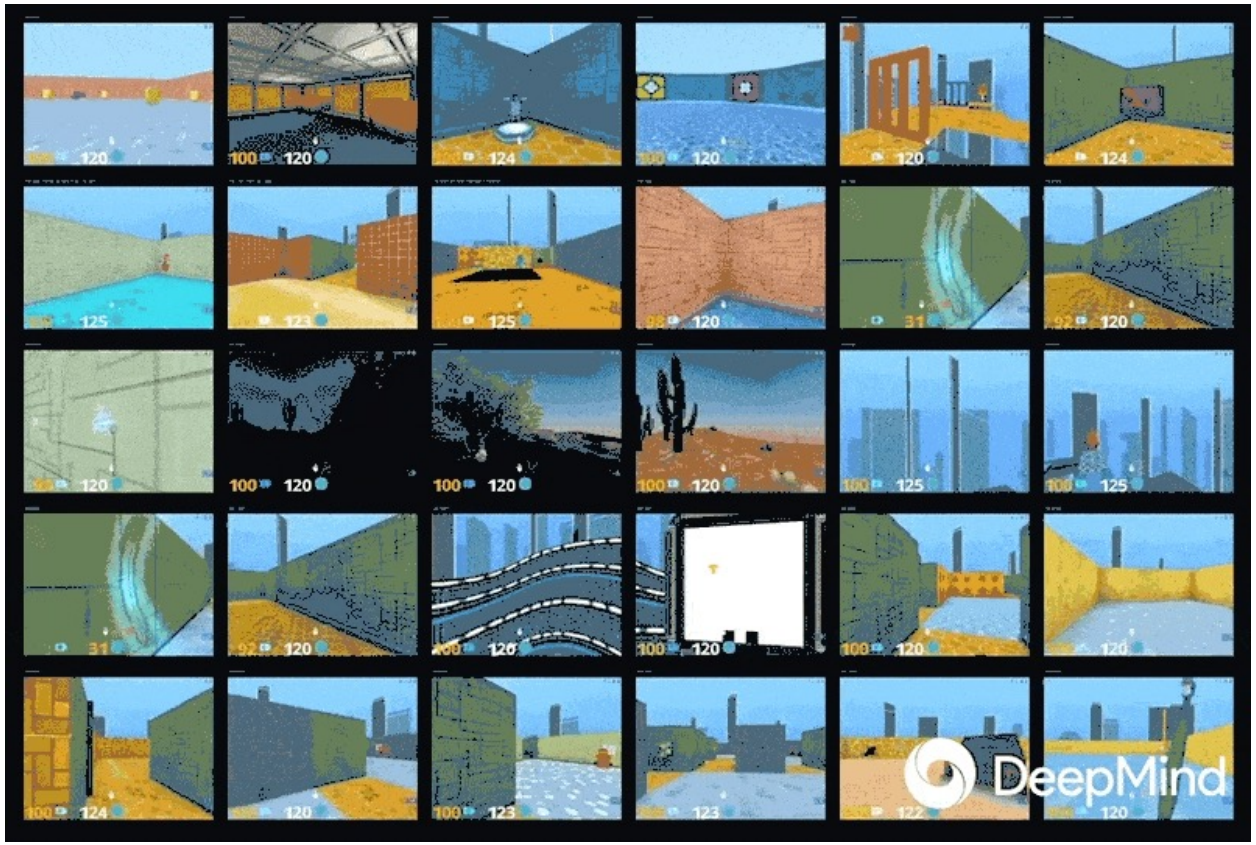
Student-teacher

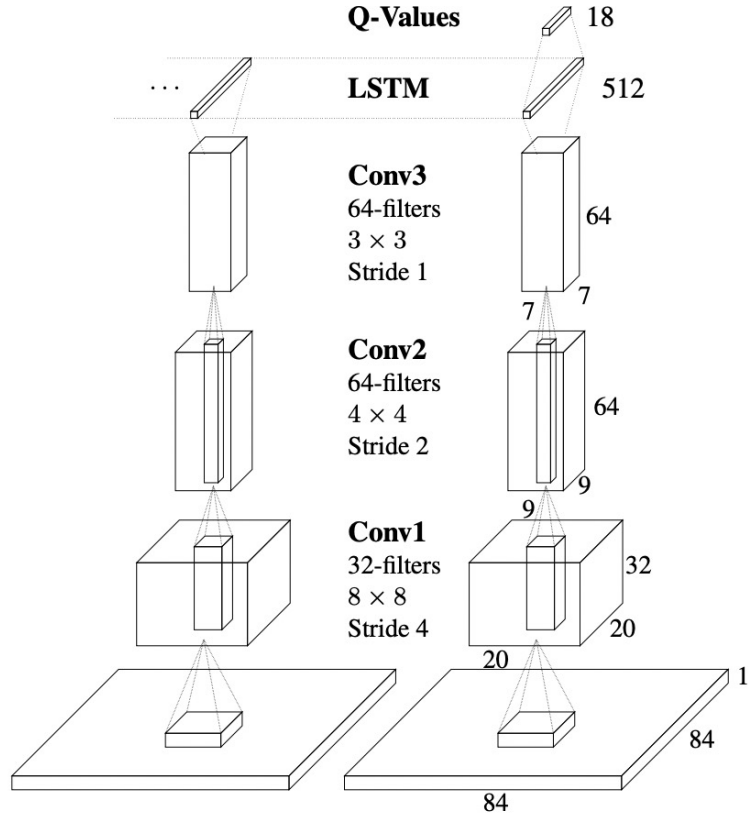


Still very unstable, lot of open research problems!

Partial Observability in Off-Policy RL

Off-policy methods critically depend on the Markov assumption



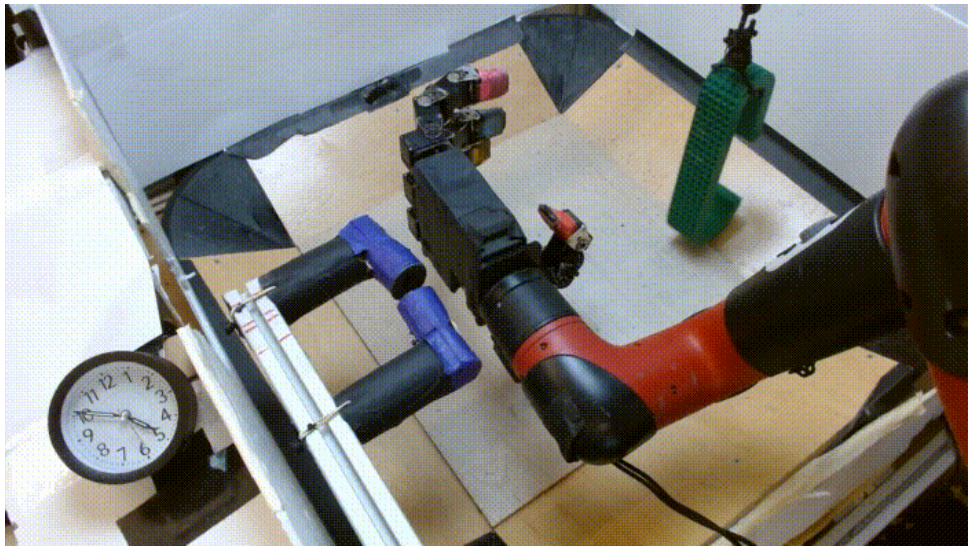


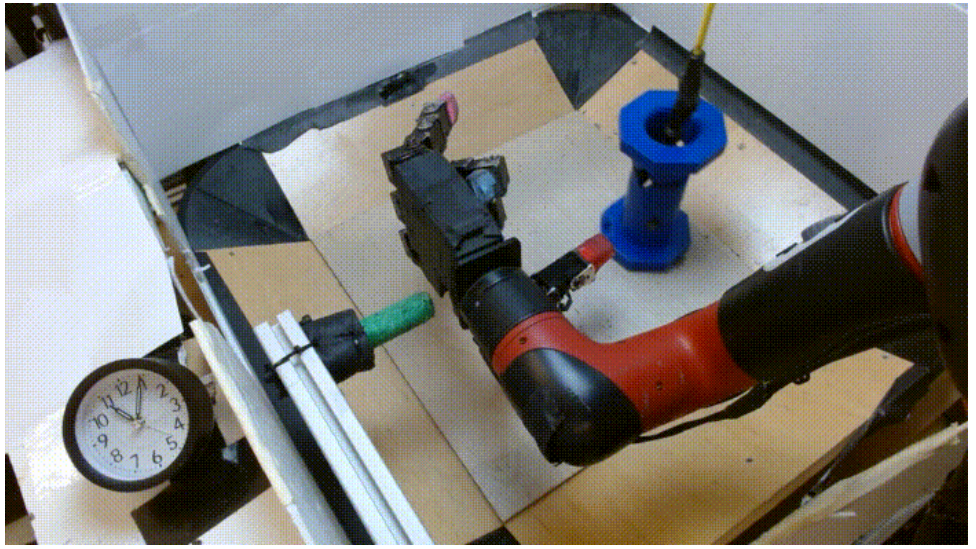
Learning history conditioned/recurrent Q-functions is an open area!

How has off-policy RL manifested in robotics?

Small changes – larger number of ensembles, more minibatch steps allow for training in < 20 mins

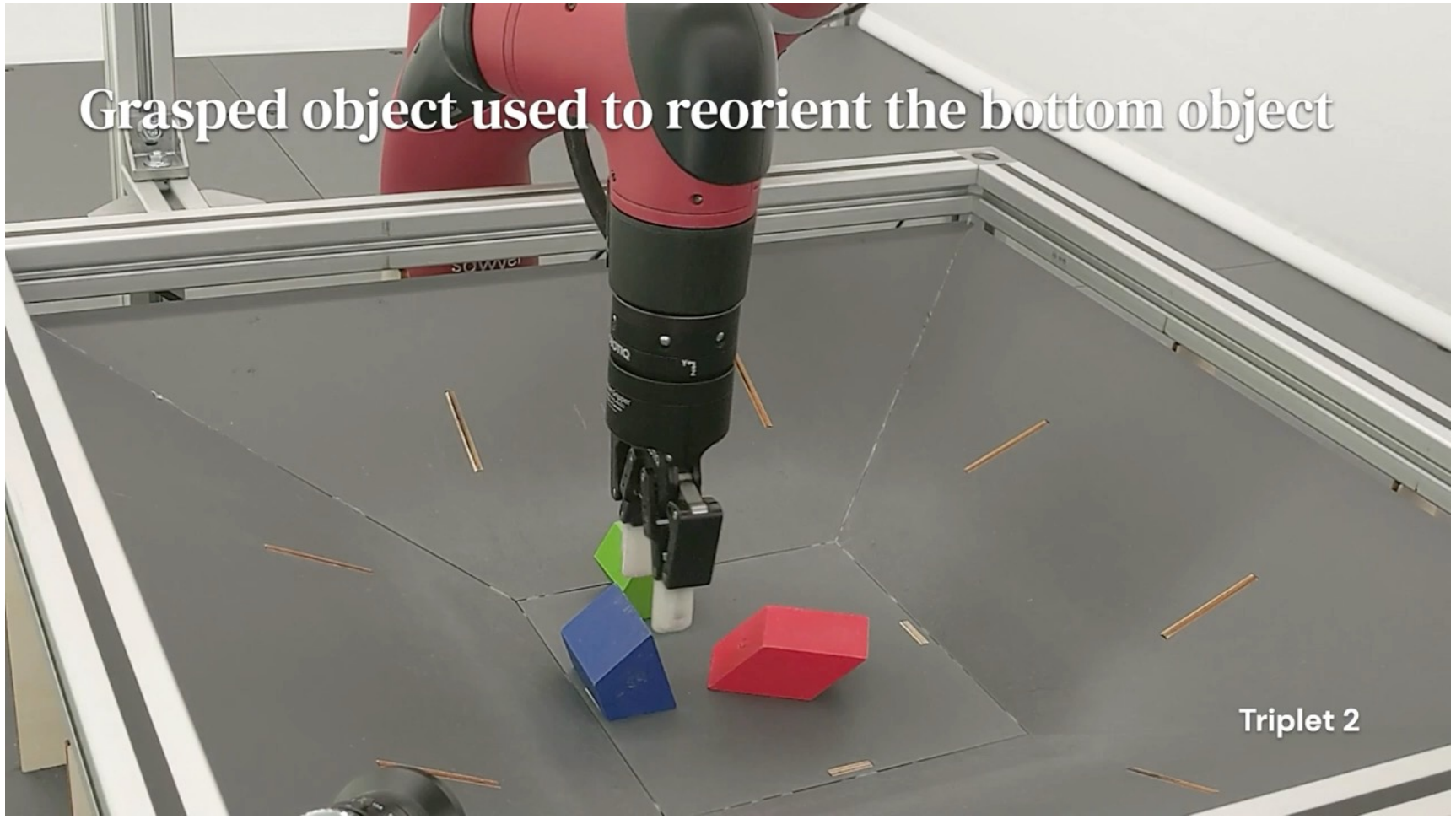
How has off-policy RL manifested in robotics?





How has off-policy RL manifested in robotics?

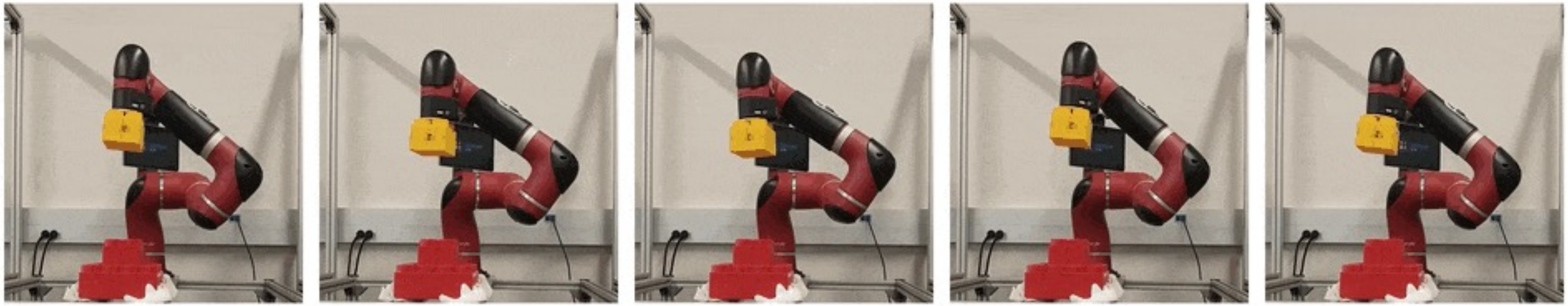
Uses MPO – a variant of actor critic with a supervised learning style actor update



How has off-policy RL manifested in robotics?

Bootstrapped with a few demonstrations

How has off-policy RL manifested in robotics?



Pros/Cons of Off-Policy Methods in Robotics

Pros:

1. Sample-efficient enough for real world
2. Can learn from images with suitable design choices
3. Off-policy, can incorporate prior data

Cons

1. Often unstable
2. Can achieve lower asymptotic performance
3. Requires significant storage

Lecture outline

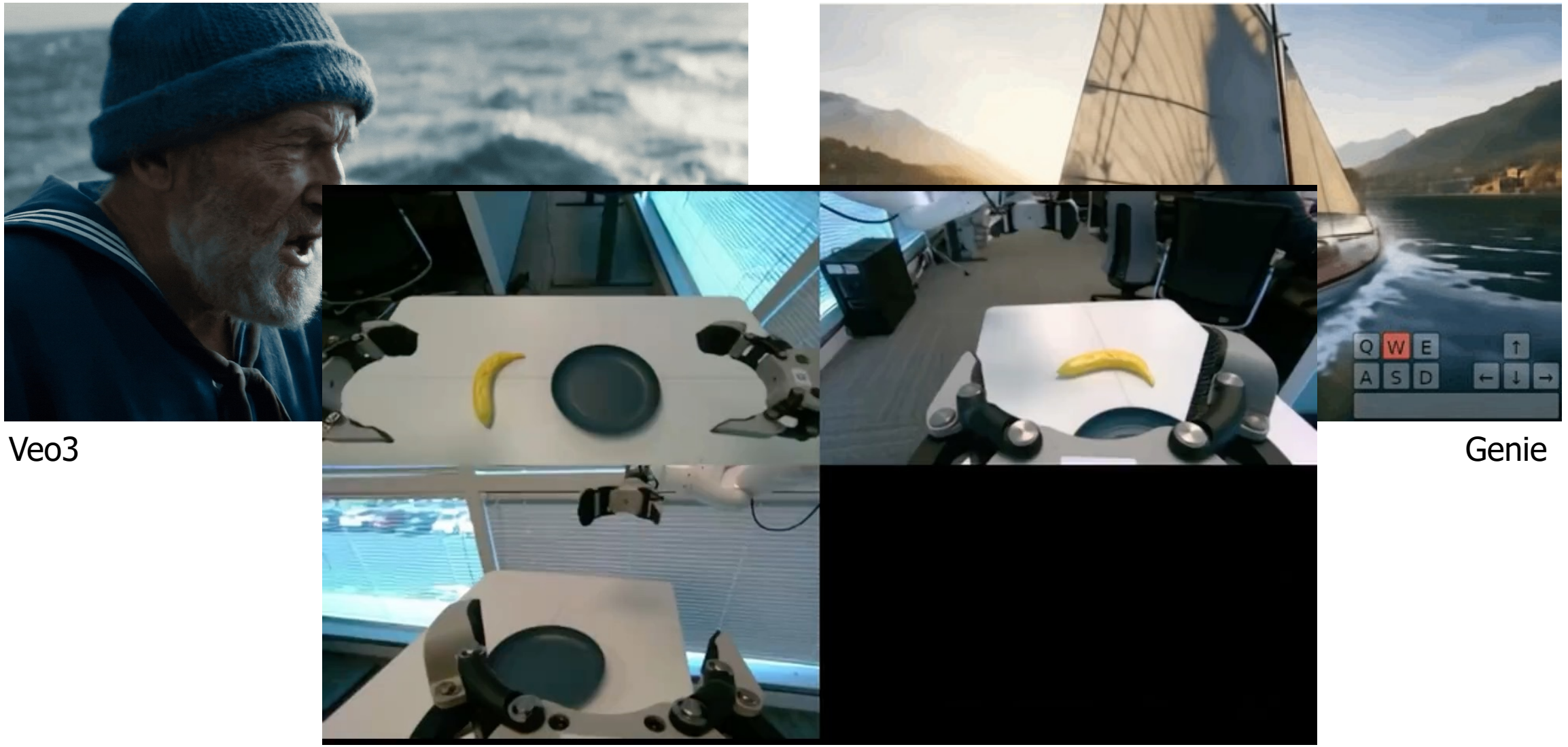
Recap

A Closer Look at Convergence

Frontiers of Off-Policy RL

Model-based RL

Let's think about models!

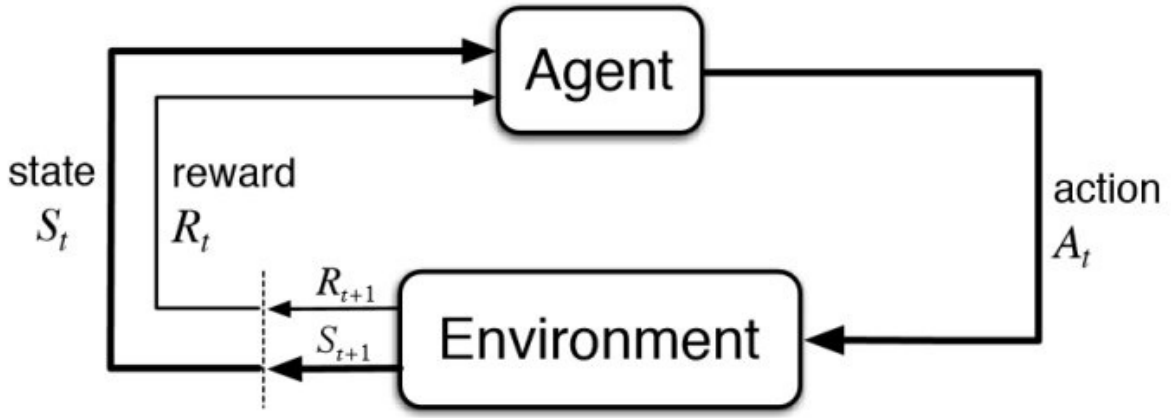


Veo3

Genie

DreamZero

How should we optimize this objective?



$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Gradient Ascent

Dynamic Programming

Model-Based Optimization

Each method has it's own +/-

Lecture outline

The Anatomy of Model-Based Reinforcement Learning

↓

Model based RL v0 → random shooting + MPC

↓

Model based RL v1 → MPPI + MPC

↓

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

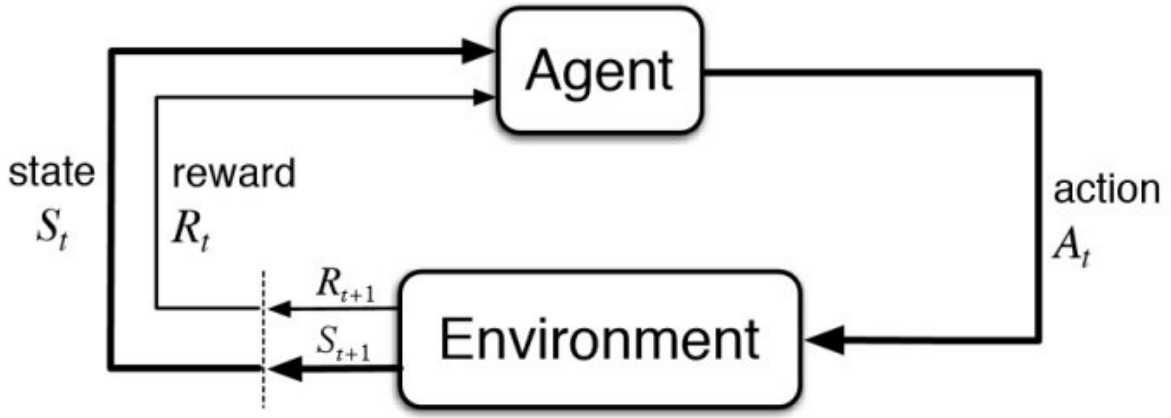
↓

Model based RL v4 → latent space models with images

↓

Model based RL v5 → From MPPI to MCTS

Landscape of Reinforcement Learning Algorithms



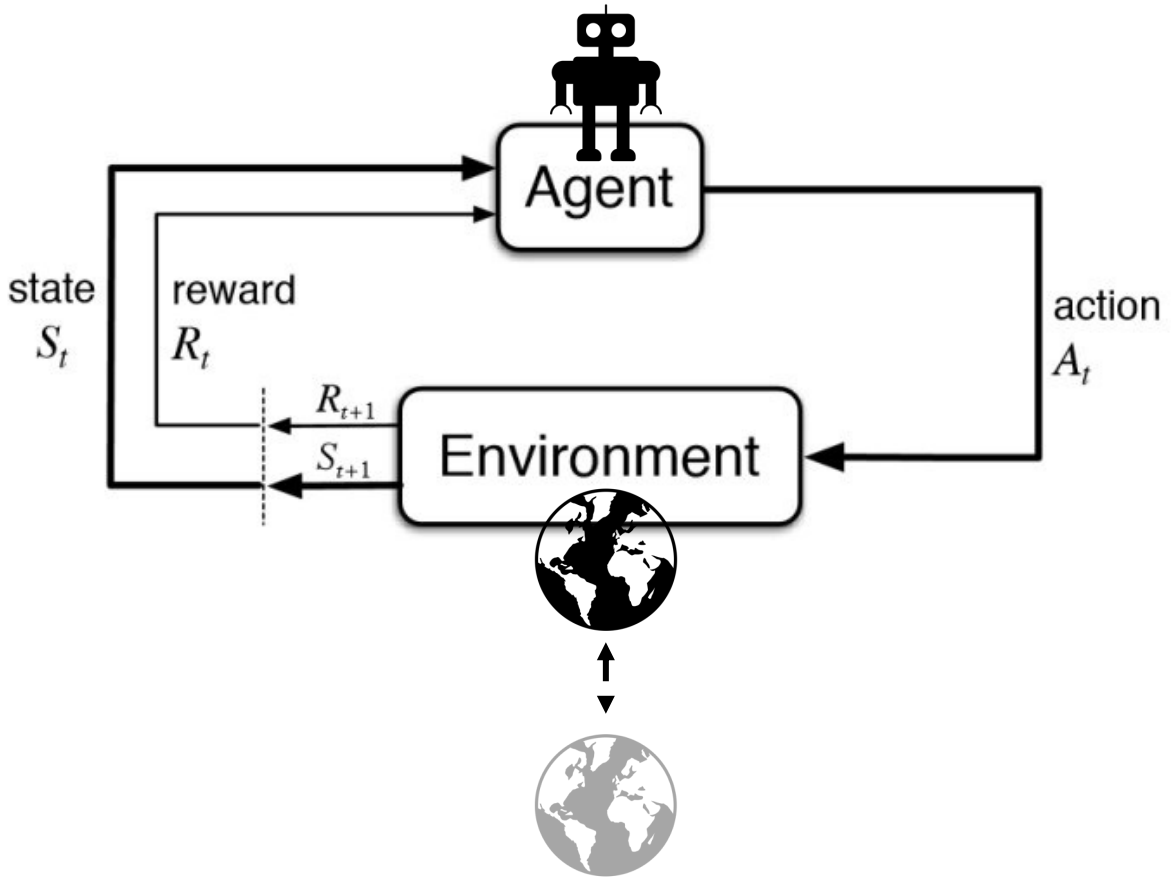
$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Gradient Ascent

Dynamic Programming

Model-Based Optimization

What if we just learned how the world worked?



$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

1. Learn a surrogate model of the transition dynamics from arbitrary off-policy data
2. Do reward maximization against this model

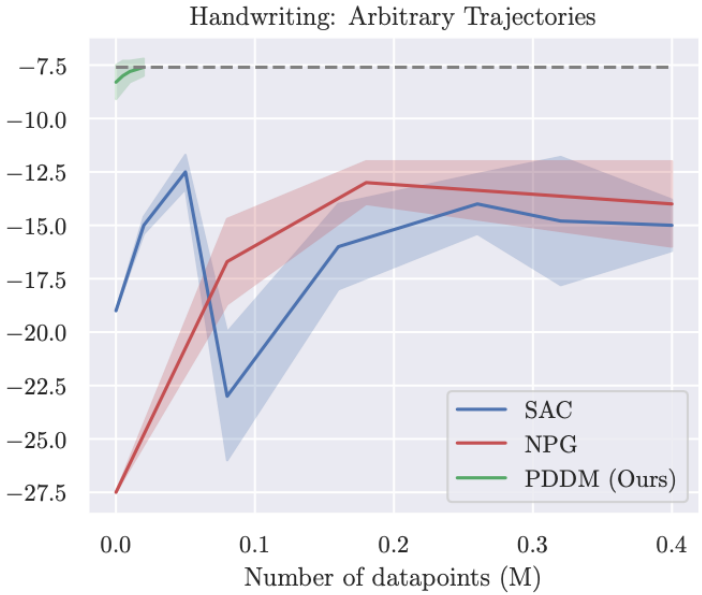
Intuitive: learn how the world works first and then plan in that model

Why do model-based RL?

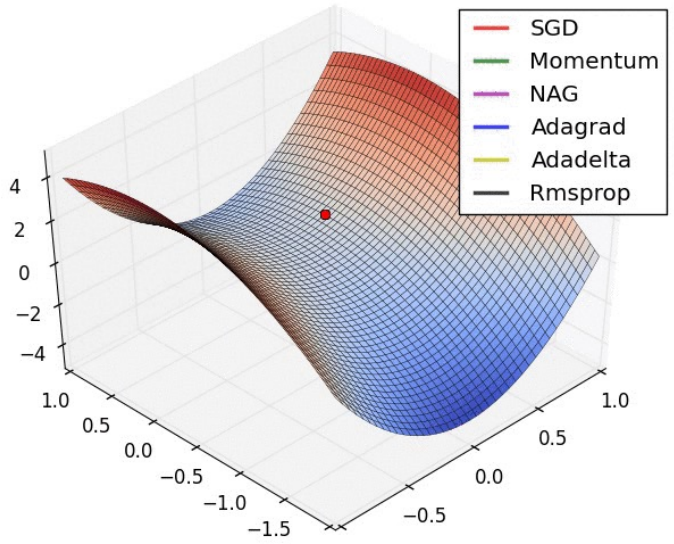
Why would we do this?

Transfer/Adaptive

Efficiency

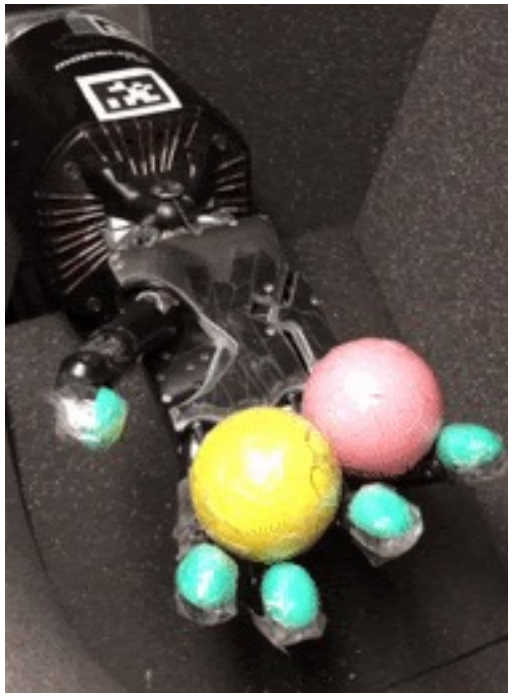


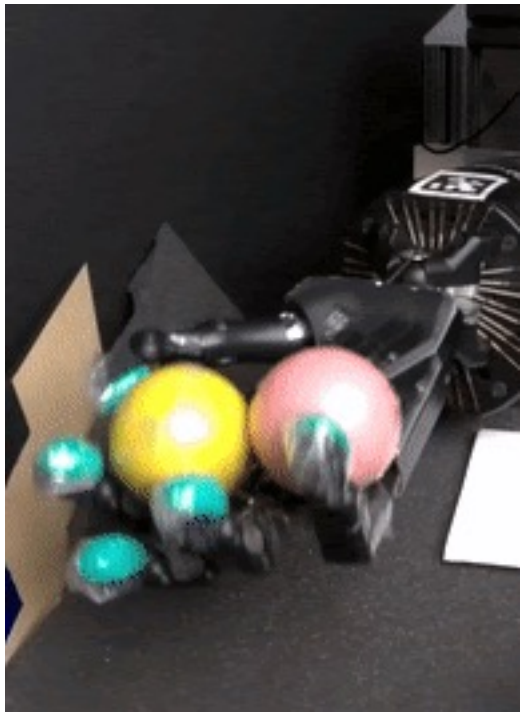
Simplicity



Naturally off-policy!

Why do model-based RL?

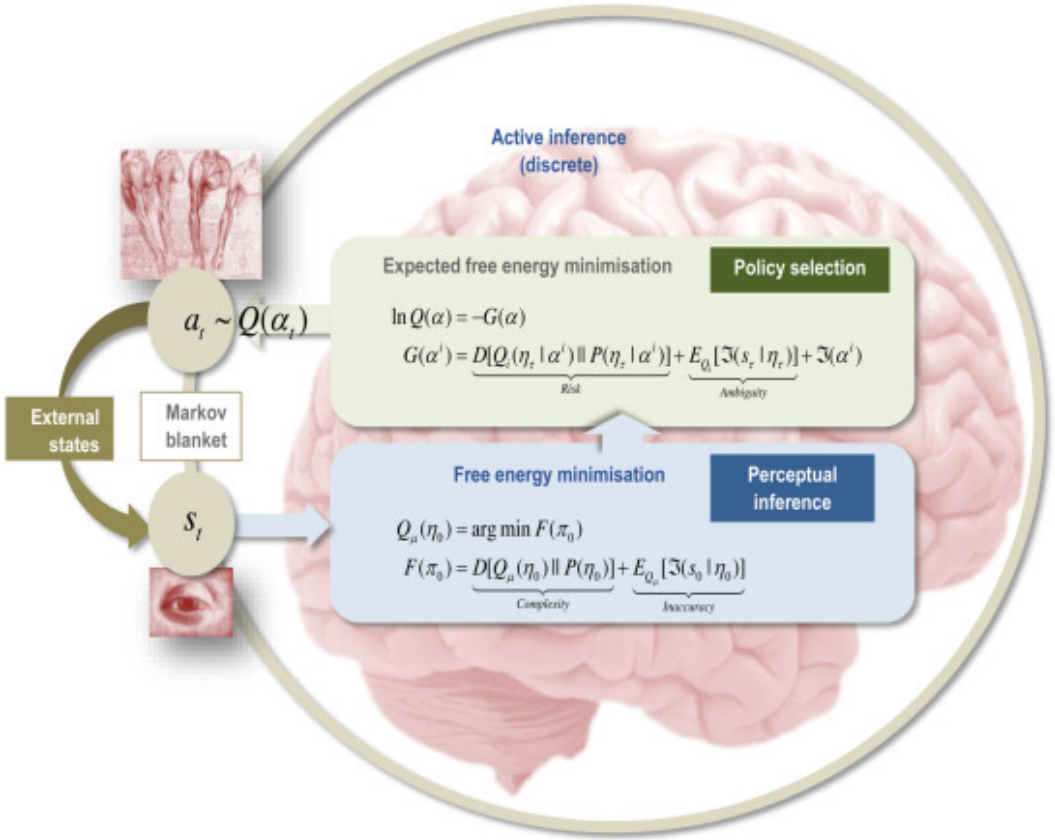


Just 2 hours of real robot training

Connections to Cognitive Science

Significant evidence for mechanisms for prediction of outcomes in neuro/cognitive science

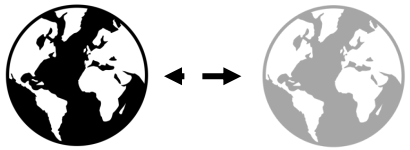


Reinforcement learning in the brain

Yael Niv
Psychology Department & Princeton Neuroscience Institute, Princeton University

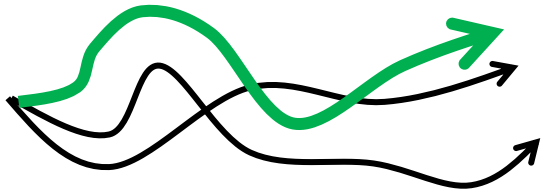
Model Based RL – Problem Statement

Model Learning



$$\hat{p}_\theta \leftarrow \arg\min_{\hat{p}_\theta} \mathcal{L}(\mathcal{D}, \hat{p}_\theta)$$

Planning



$$\arg\max_{\pi} \mathbb{E}_{\hat{p},\pi} \left[\sum_t r(s_t, a_t) \right]$$

Can also just be a single trajectory

How should we instantiate these?

What will we not cover today?

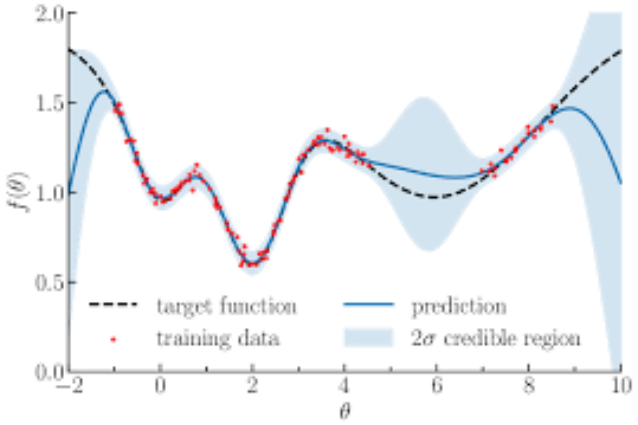
iLQR/iLQG

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \text{ s.t. } \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$$

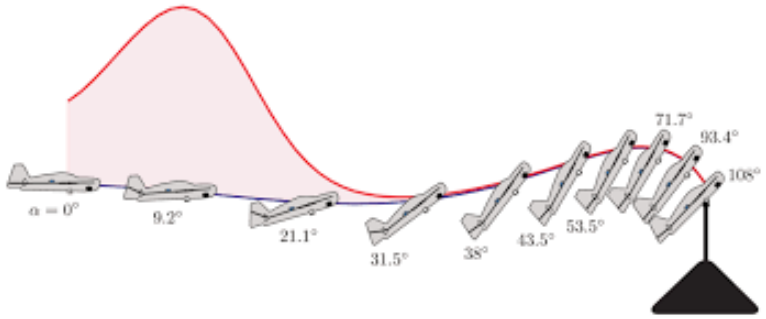
$$f(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{F}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \mathbf{f}_t$$

$$c(\mathbf{x}_t, \mathbf{u}_t) = \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{C}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{c}_t$$

MBRL with GPs/Non-Parametrics



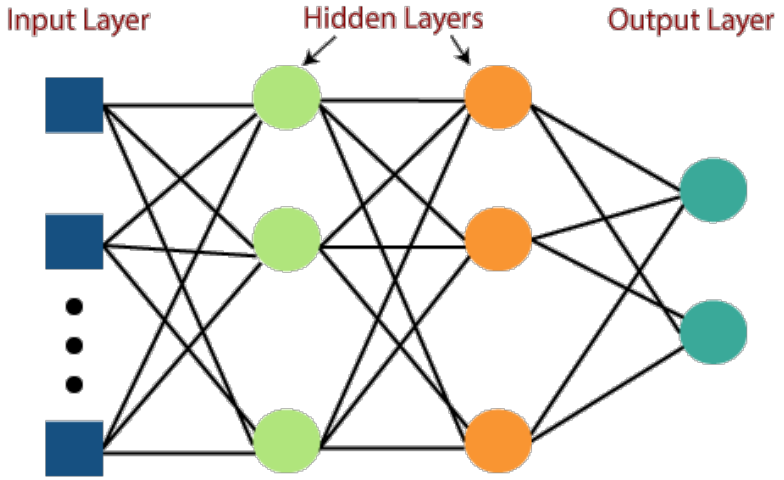
Non-linear TrajOpt



Byron's lectures do a wonderful job, do go watch them!

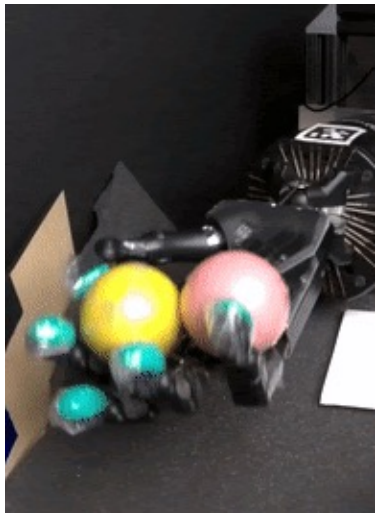
What will we cover today?

Use neural networks as our model!

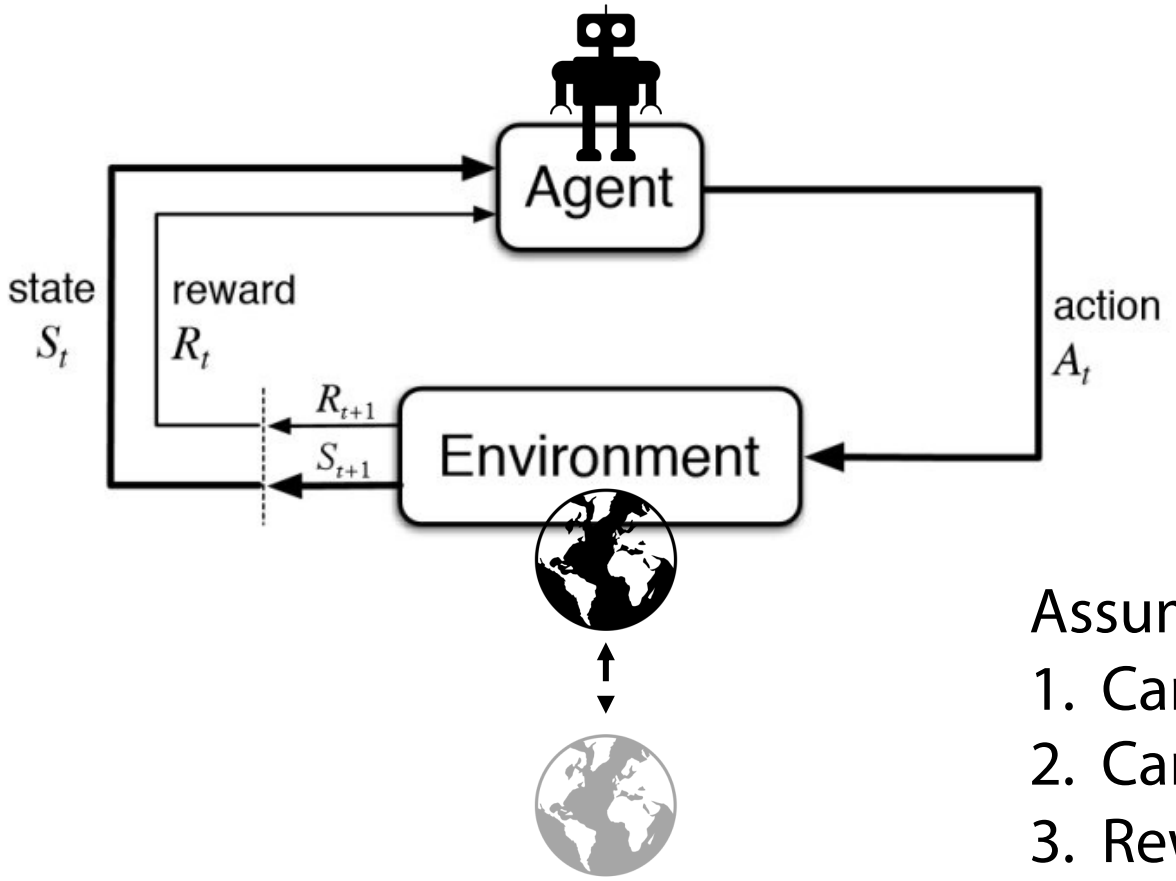


$$\hat{p}_\theta \leftarrow \arg\min_{\hat{p}_\theta} \mathcal{L}(\mathcal{D}, \hat{p}_\theta)$$

$$\arg\max_\pi \mathbb{E}_{\hat{p},\pi} \left[\sum_t r(s_t, a_t) \right]$$



Model Based RL – Assumptions



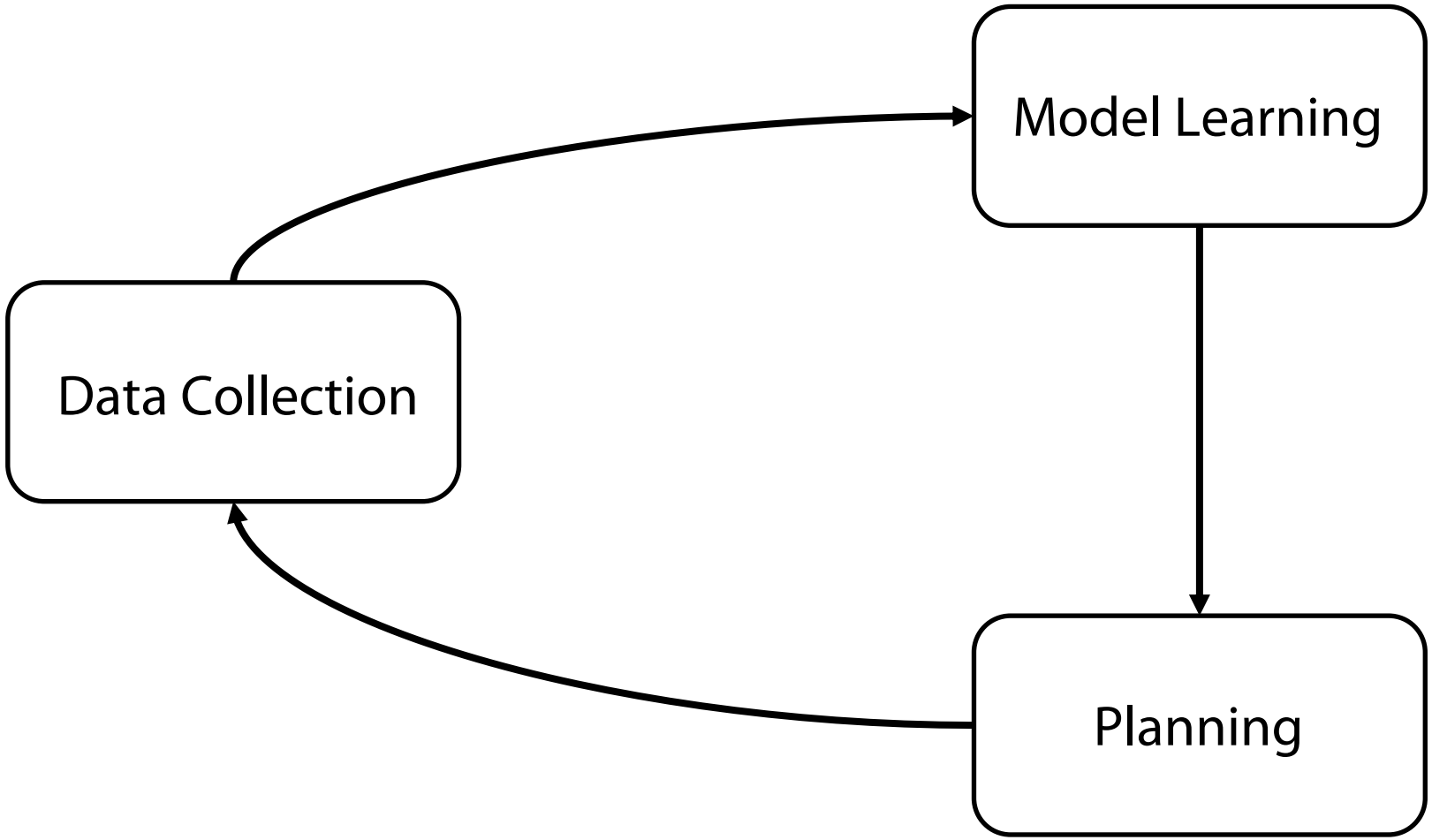
$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Assumptions:

1. Can only **sample** from dynamics
2. Can **reset** the environment
3. Reward function is **known**

We will get into this in a later lecture!

Model Based RL – A template



Lecture outline

The Anatomy of Model-Based Reinforcement Learning

↓

Model based RL v0 → random shooting + MPC

↓

Model based RL v1 → MPPI + MPC

↓

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

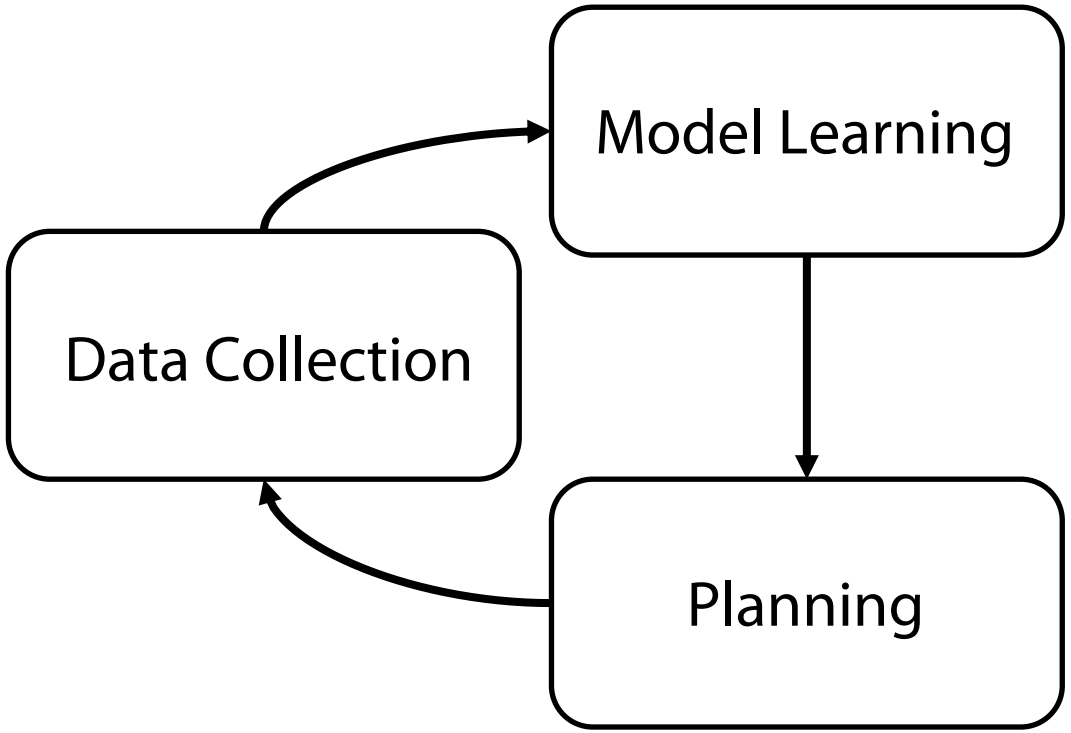
↓

Model based RL v4 → latent space models with images

↓

Model based RL v5 → From MPPI to MCTS

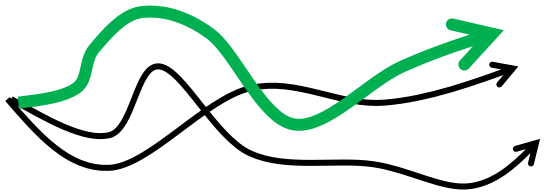
Model Based RL – Naïve Algorithm (v0)



Maximum likelihood supervised Learning

$$\max_{\theta} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right]$$

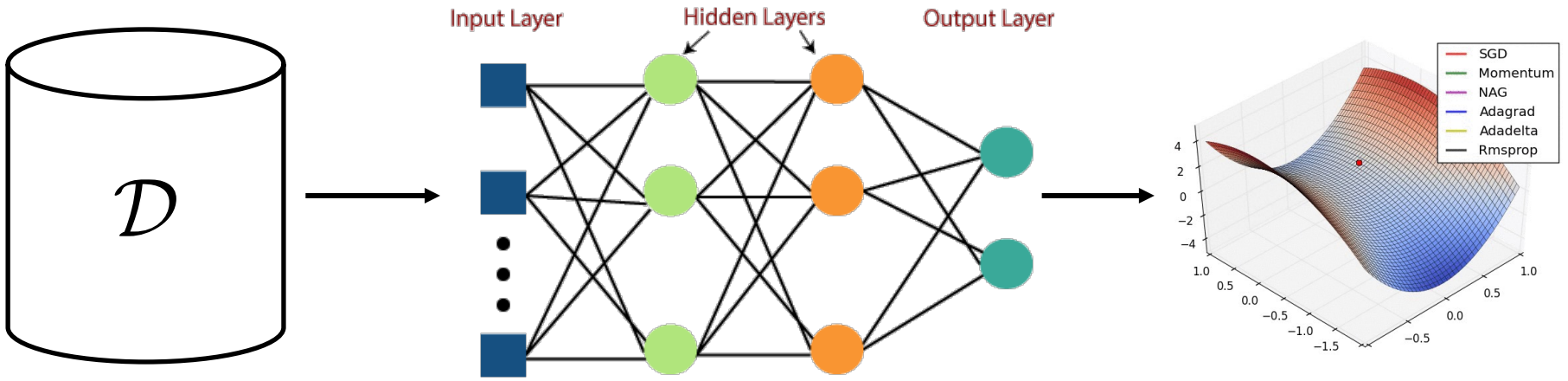
Random Search



Model Based RL – Naïve Algorithm (Model Learning) (v0)

$$\max_{\theta} \mathbb{E}_{(s,a,s') \sim \mathcal{D}} \left[\log \hat{p}_\theta(s' | s, a) \right]$$

Fit 1-step models



More expressive may be better, at the risk of overfitting

Choice of $\hat{p}_\theta$ distribution determines the loss function:

1. Gaussian → L_2
2. Energy Based Model → Contrastive Divergence
3. Diffusion Model → Score Matching

Trick: Model Residual's (s' –s)

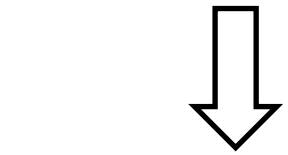
Model Based RL – Naïve Algorithm (Planning)

Planning

$$\max_{a_0, a_1, \ldots, a_T} \sum_{t=0}^{T} r(\hat{s}_t, a_t)$$

$$\hat{s}_{t+1} \sim \hat{p}_\theta(s_{t+1} | \hat{s}_t, a_t)$$

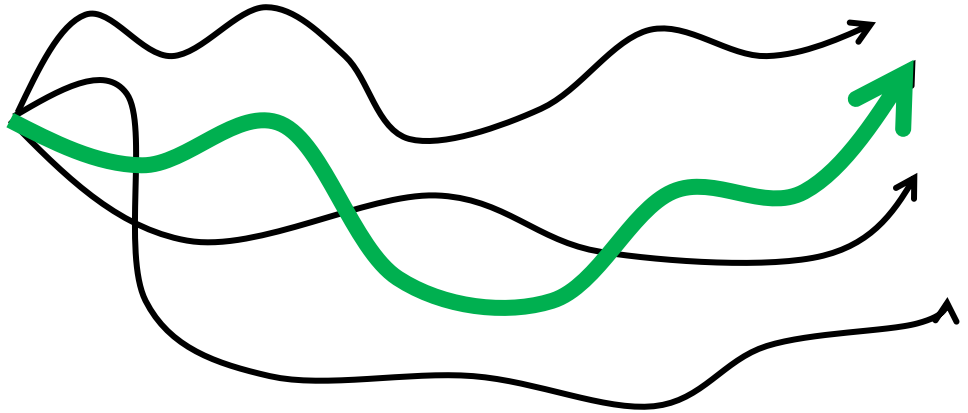
$$\hat{s}_1 \sim \hat{p}_\theta(s_{t+1} | s_0, a_0)$$



Just do random search!

$$\arg \max_{a_0^j, a_1^j, \ldots, a_T^j} \sum_{t=0}^{T} r(\hat{s}_t^j, a_t^j)$$

$$\hat{s}_{t+1}^j \sim \hat{p}_\theta(.|\hat{s}_t^j, a_t^j)$$

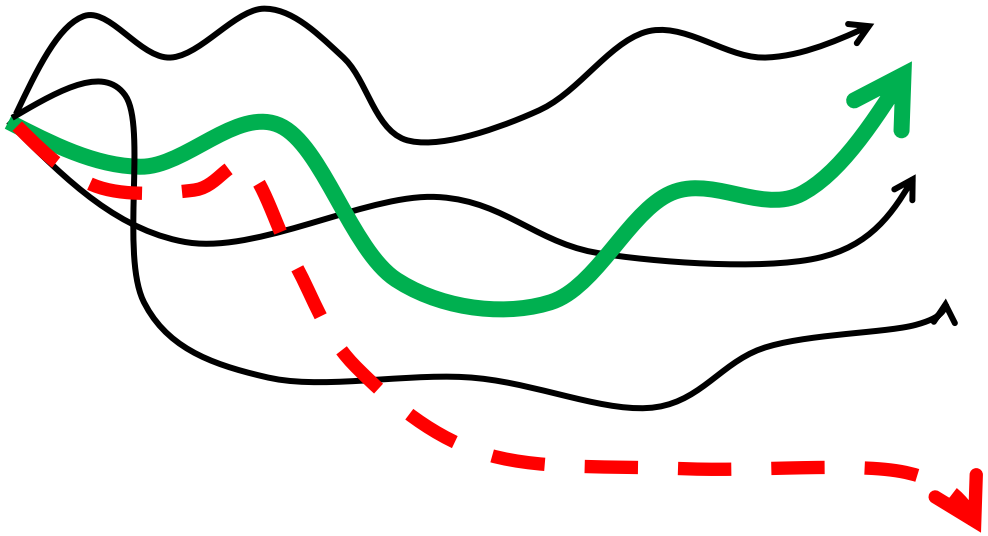


Just execute actions open loop!

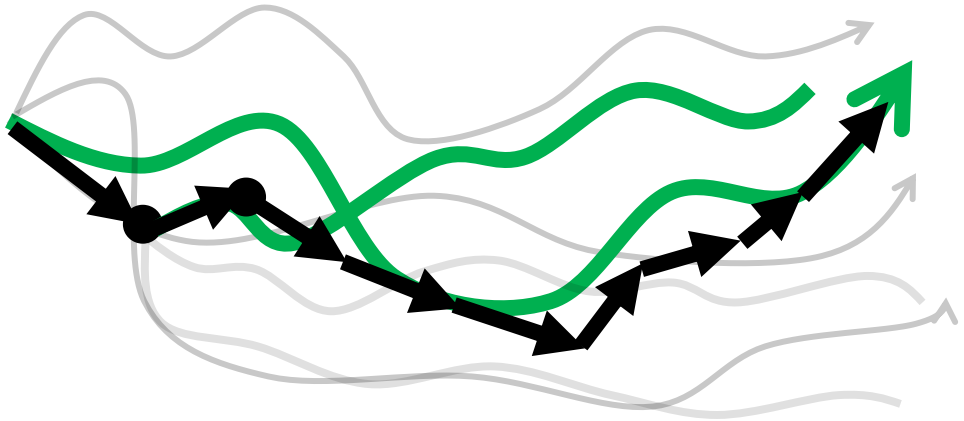
Can soften by taking softmax rather than argmax

Model Based RL – Naïve Algorithm (MPC)

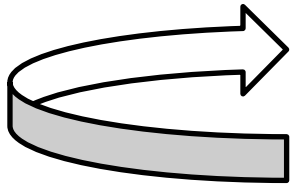
Without feedback, an open loop controller can diverge even for minimal noise



Replanning can help with divergence

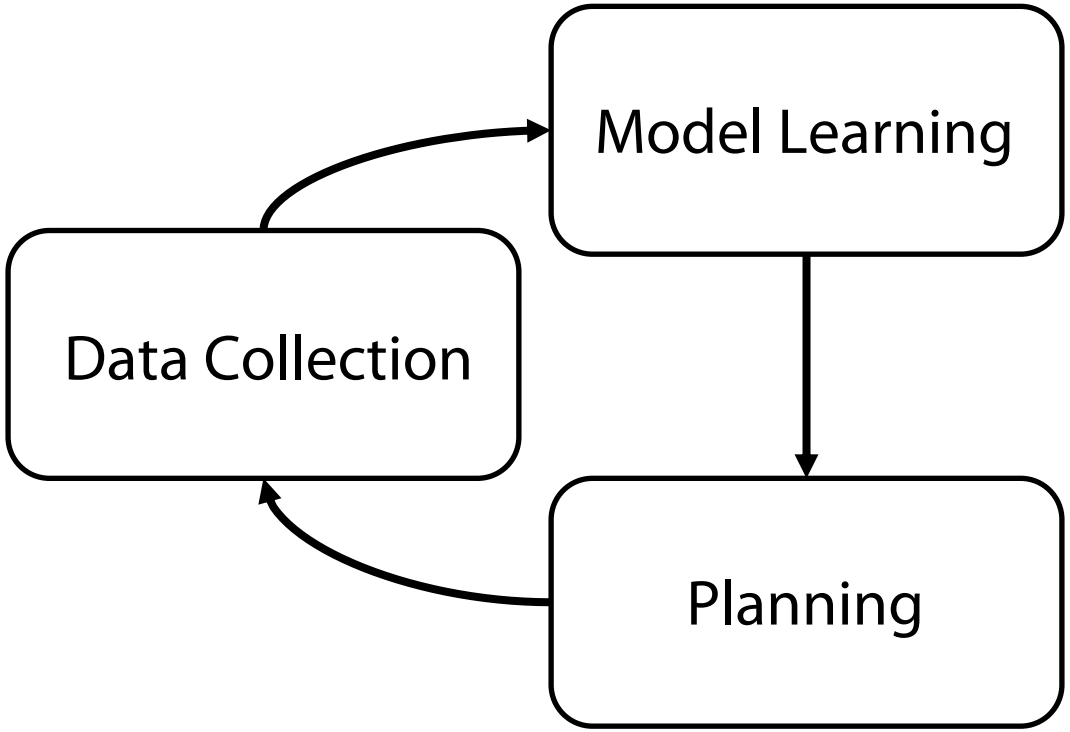


Model-Predictive/Receding Horizon Control



1. Plan with random shooting from s_t
2. Execute the first action a_0 and reach s_{t+1}

Model Based RL – Naïve Algorithm (v0)

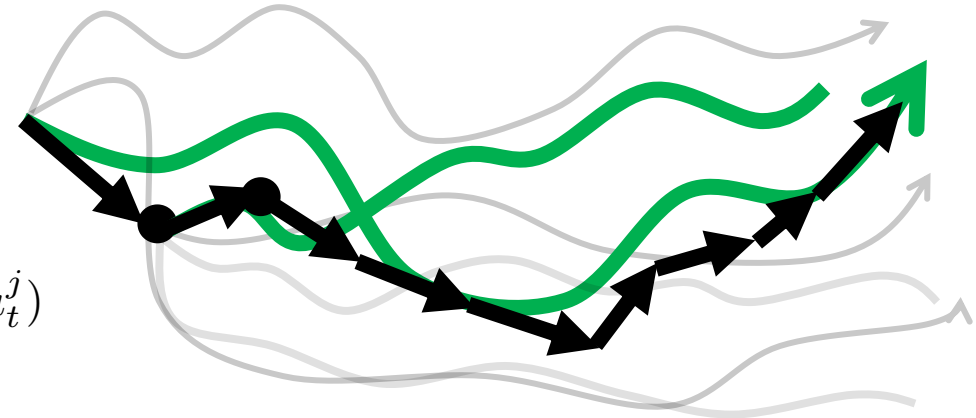


Data collected from planner

Maximum likelihood supervised Learning

$$\max_{\theta} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right]$$

Planning with Shooting + MPC

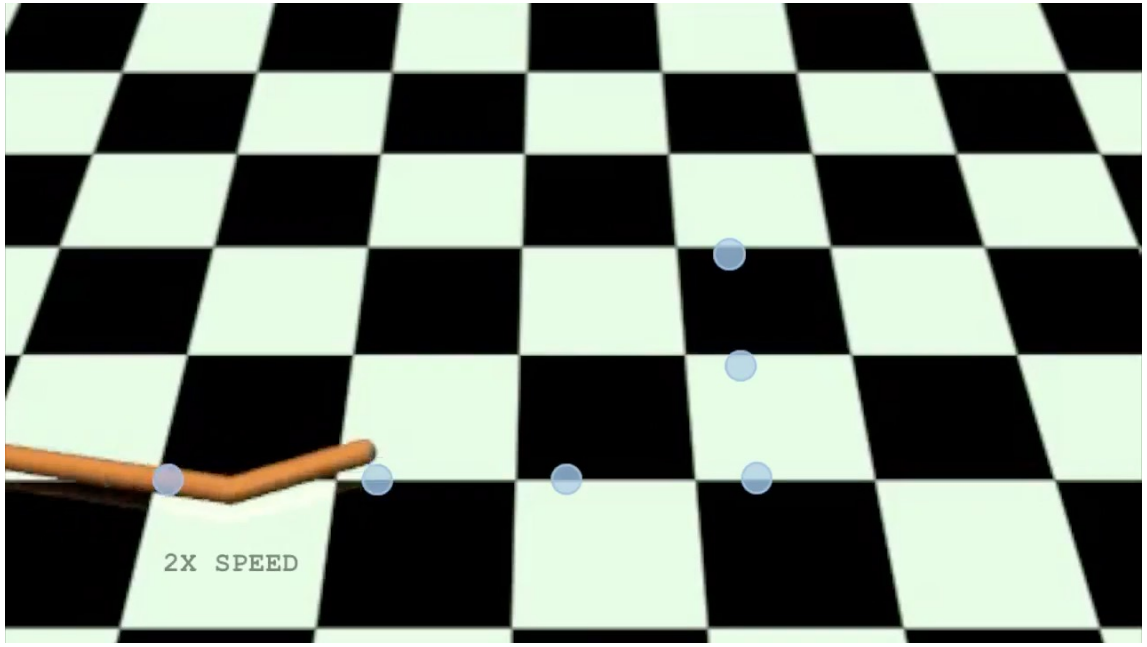


$$\arg \max_{a_0^j, a_1^j, \ldots, a_T^j} \sum_{t=0}^{T} r(\hat{s}_t^j, a_t^j)$$

$$\hat{s}_{t+1}^j \sim \hat{p}_\theta(.|\hat{s}_t^j, a_t^j)$$

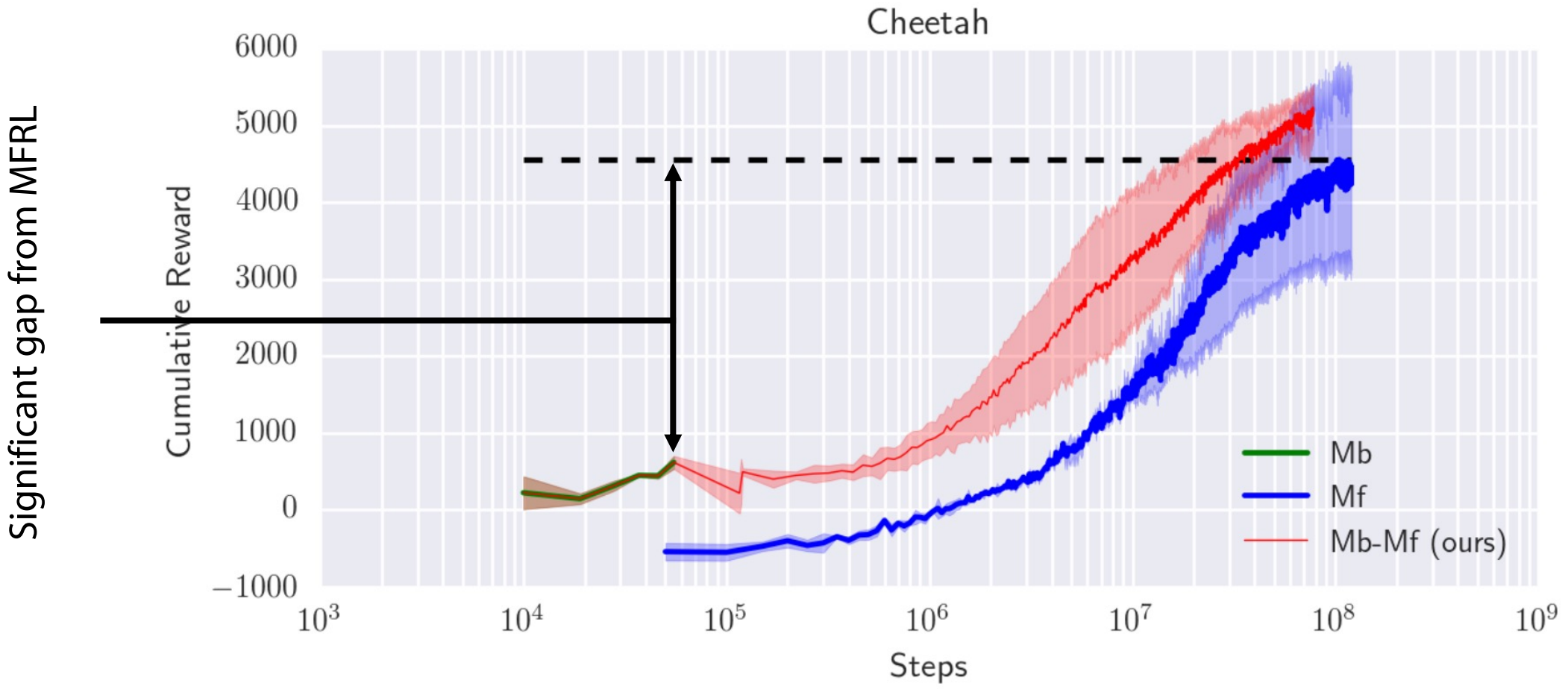
Better than open loop planning because of feedback

Does it work?



Just 20 minutes of training time with random data!

Does it work?


Significant gap from MFRL
Cheetah
Cumulative Reward
6000
5000
4000
3000
2000
1000
0
−1000
10^3
10^4
10^5
10^6
10^7
10^8
10^9
Steps
Mb
Mf
Mb-Mf (ours)

Lecture outline

The Anatomy of Model-Based Reinforcement Learning

↓

Model based RL v0 → random shooting + MPC

↓

Model based RL v1 → MPPI + MPC

↓

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

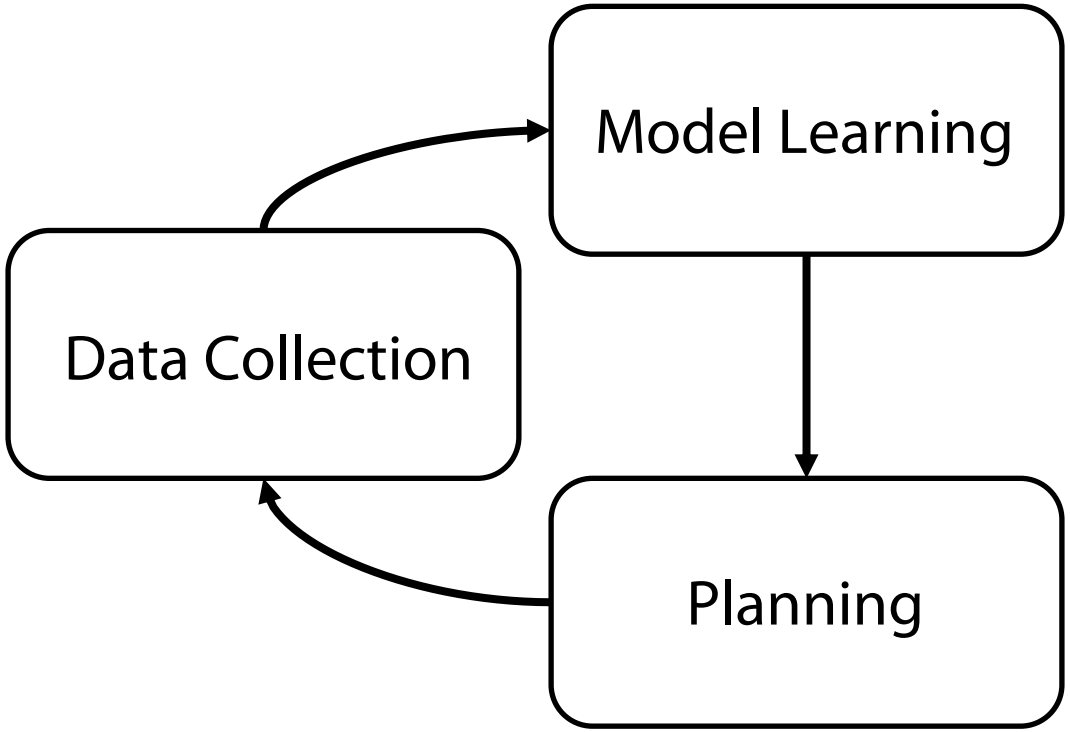
↓

Model based RL v4 → latent space models with images

↓

Model based RL v5 → From MPPI to MCTS

What might be the issue?



Data collected from planner

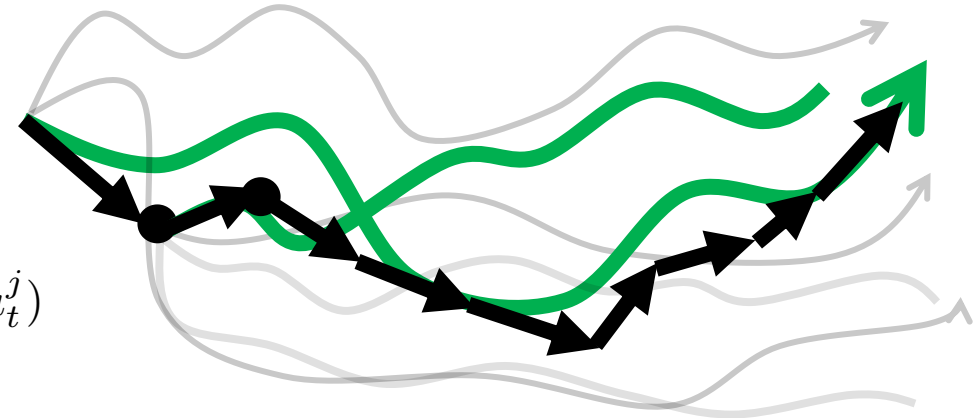
Maximum likelihood supervised Learning

$$\max_{\theta} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right]$$

Planning with Shooting + MPC

$$\arg \max_{a_0^j, a_1^j, \ldots, a_T^j} \sum_{t=0}^{T} r(\hat{s}_t^j, a_t^j)$$

$$\hat{s}_{t+1}^j \sim \hat{p}_\theta(.|\hat{s}_t^j, a_t^j)$$



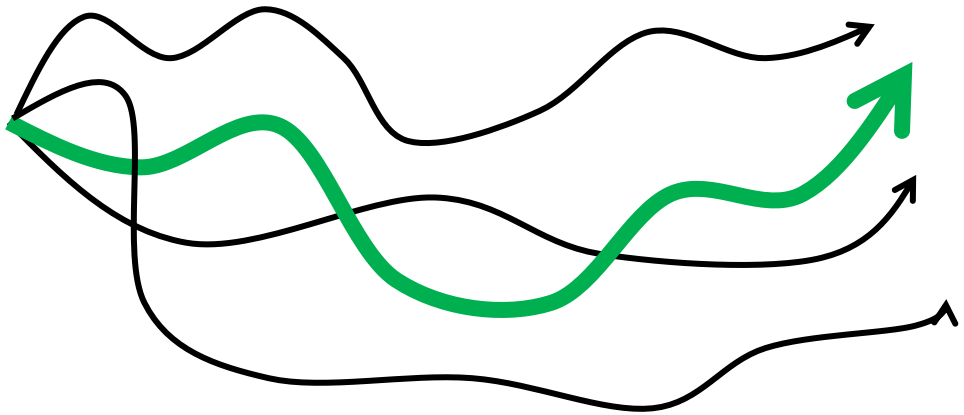
Searching for a needle in a haystack by random shooting, high variance!

Better Sampling Techniques for Shooting

Sampled from stationary uniform/gaussian distribution

$$\arg \max_{a_0^j, a_1^j, \ldots, a_T^j} \sum_{t=0}^{T} r(\hat{s}_t^j, a_t^j)$$

$$\hat{s}_{t+1}^j \sim \hat{p}_\theta(.|\hat{s}_t^j, a_t^j)$$

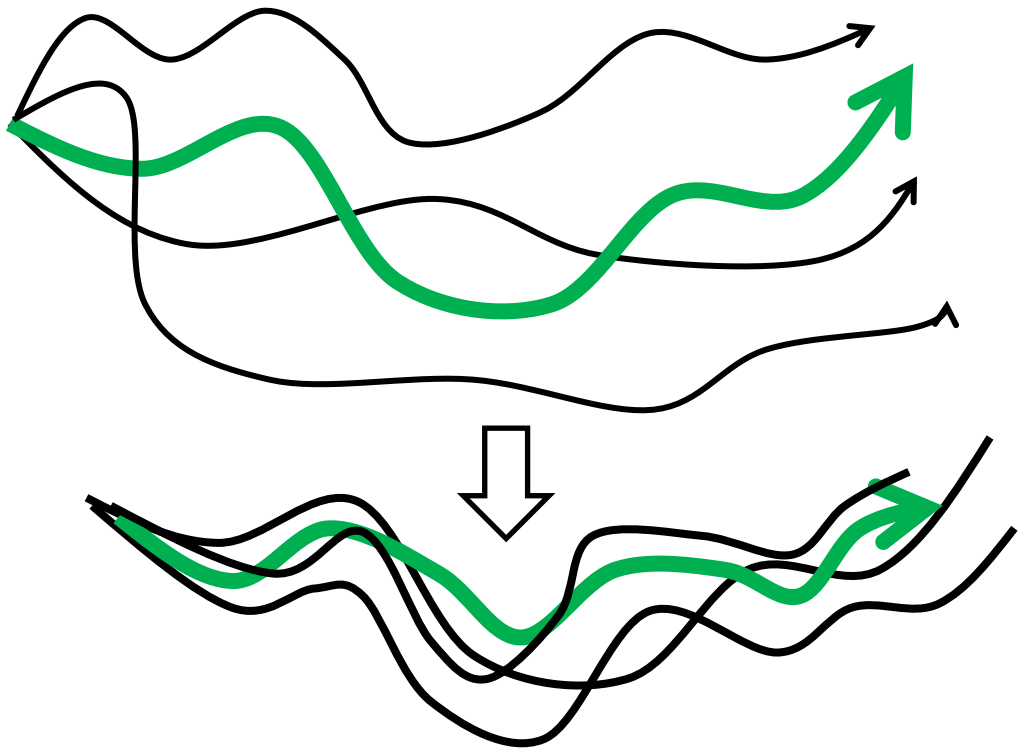


Can we inform the sampling function with the reward function?

Idea: Iteratively upweight sampling distribution around the things that are higher returns

Better Sampling Techniques for Shooting - MPPI

Idea: Iteratively upweight sampling distribution around the things that are higher returns



Referred to as **MPPI**, lower variance!

Sample N action sequences

$$(a_0^j, a_1^j, \ldots, a_T^j)_{j=1}^N \sim p(a)$$

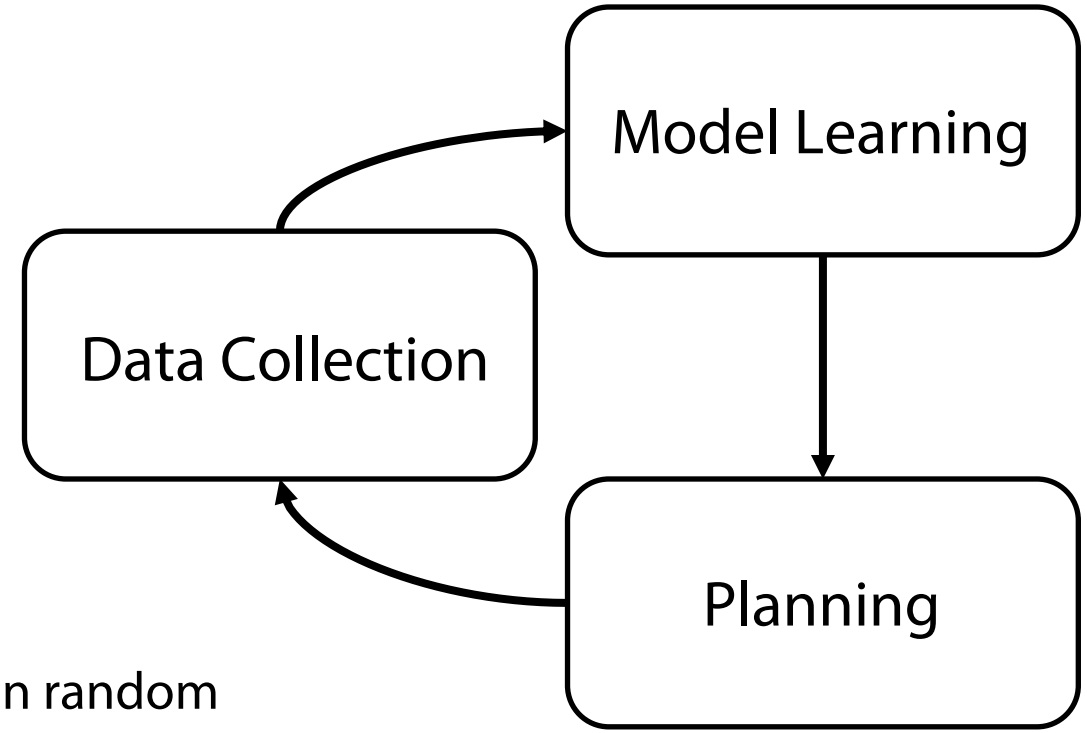
Sample trajectories using these action sequences with the model $\hat{p}_\theta$

$$\hat{s}_{t+1} \sim \hat{p}_\theta(.|\hat{s}_t, a_t)$$

Update action sampler by upweighting high return actions

$$p(a) \leftarrow p(a)\frac{\exp(\sum_t r(s_t, a_t))}{Z}$$

Model Based RL – Better Sampling Methods (v1)

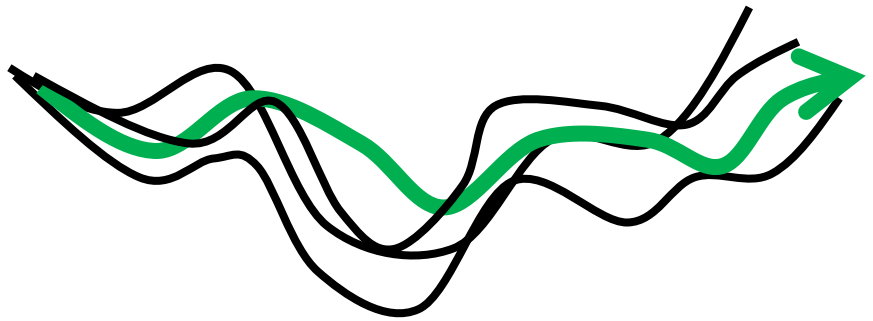


Data collected from planner

Maximum likelihood supervised Learning

$$\max_{\theta} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right]$$

Planning with MPPI + MPC



Better than random shooting + MPC, since lower variance!

Aside: Can derive this update trying to bring sampling distribution close to optimal distribution

$$\arg \max_{a_0^j, a_1^j, \ldots, a_T^j} \sum_{t=0}^{T} r(\hat{s}_t^j, a_t^j)$$

$$\hat{s}_{t+1}^j \sim \hat{p}_\theta(.|\hat{s}_t^j, a_t^j)$$

$$p(a) \leftarrow p(a) \frac{\exp(\sum_t r(s_t, a_t))}{Z}$$

Does it work?

Lecture outline

The Anatomy of Model-Based Reinforcement Learning

↓

Model based RL v0 → random shooting + MPC

↓

Model based RL v1 → MPPI + MPC

↓

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

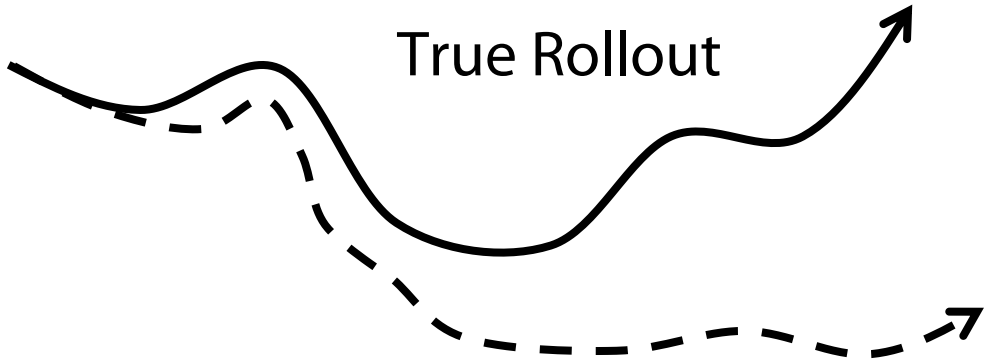
↓

Model based RL v5 → From MPPI to MCTS

What might be the issue?

Rollouts under learned model != Rollouts under true model

→ Model bias/compounding error



Why does this happen? → lack of data

1. Errors in state go to OOD next states
2. Deviations in actions go to OOD next states

↓

Model is bad on OOD states!

Most trained deep models can only roll out for 5-10 steps maximum!

How might we deal with compounding error?

Idea 1: Change the training objective of the model to directly account for this!

Equation error – 1 step prediction error

$$\max_{\theta} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right]$$

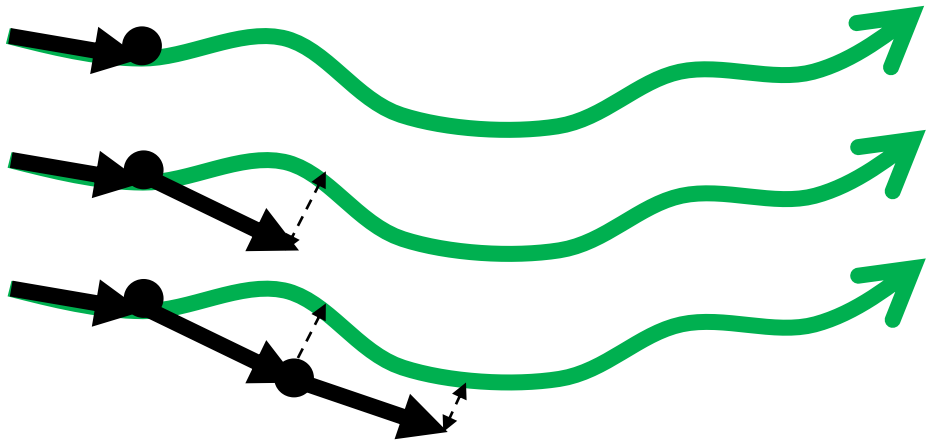
Simulation error – K step prediction error

$$\max_{\theta} \sum_{t} \log \hat{p}_\theta(s_{t+1}|\hat{s}_t, a_t)$$

$$\hat{s}_t \sim \hat{p}_\theta(.|\hat{s}_{t-1}, a_{t-1})$$

Model error under learned mode $\hat{p}_\theta$ rather under true model

Can be a challenging non-convex optimization!

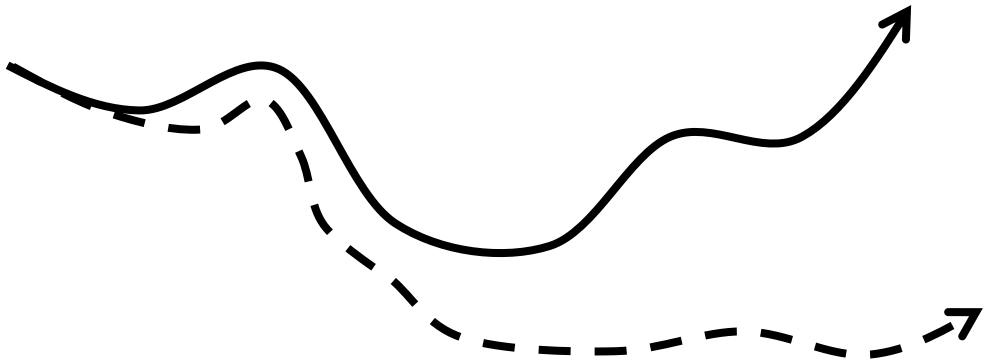


How might we deal with compounding error?

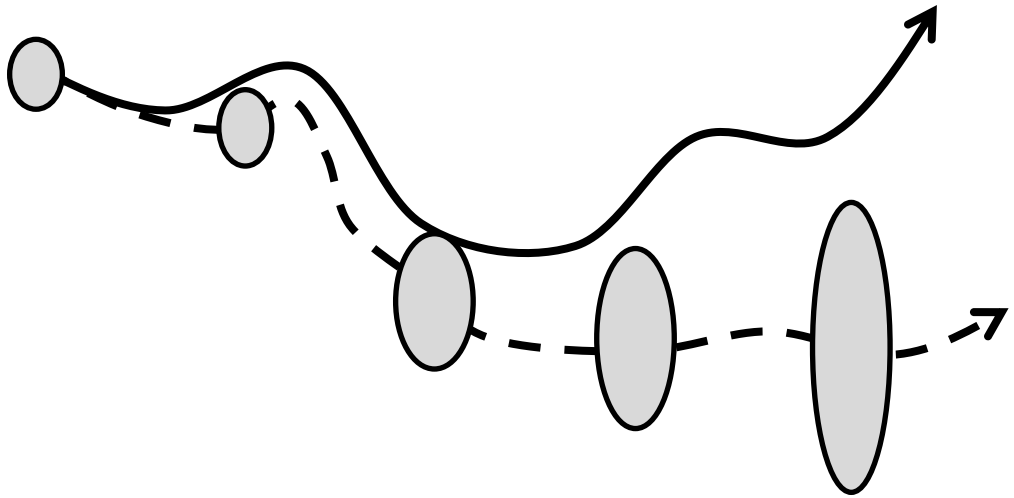
Idea 2: Estimate when OOD and account for it

→ Measure uncertainty!

Maximum likelihood models



Uncertainty-aware models



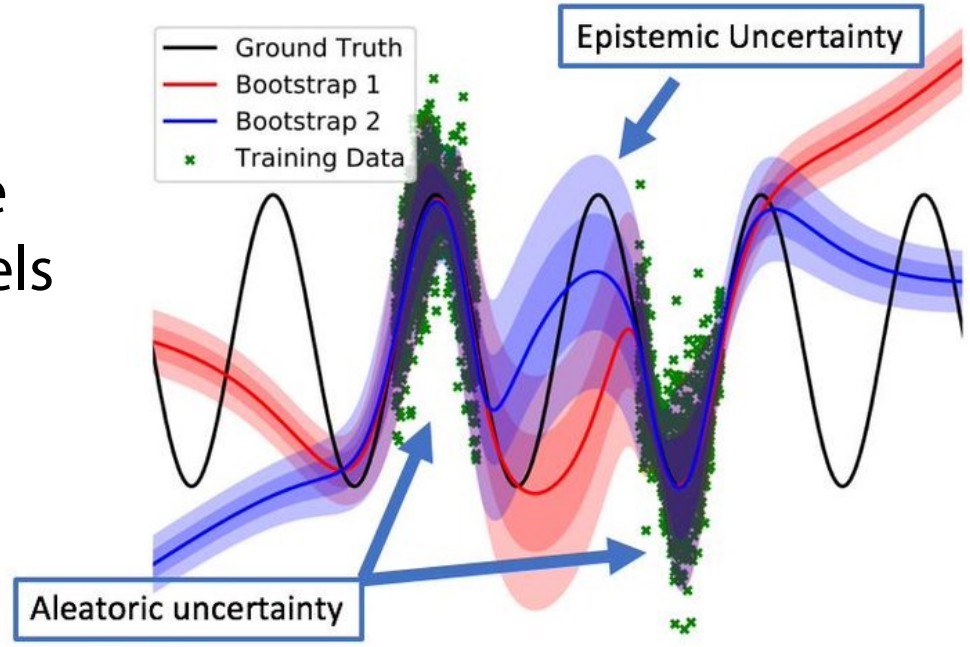
Being aware of uncertainty allows us to account for the effects of model bias!

What is uncertainty?

Alleatoric Uncertainty

(environment stochasticity)

Easier, can use stochastic models



Epistemic Uncertainty

(Lack of data)

More challenging, need to compute posterior

Let's largely focus on epistemic uncertainty

How might we measure uncertainty?

$$p(\theta|\mathcal{D})$$

Difficult to estimate directly!

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta')p(\theta')d\theta'}$$

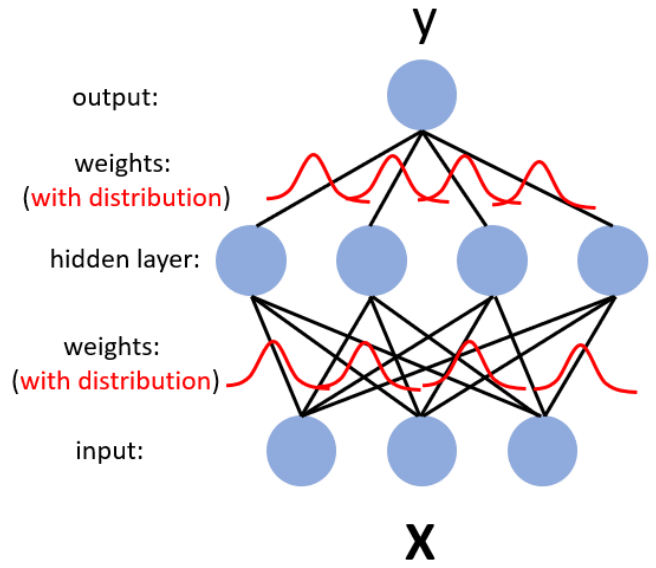
1. Bayesian neural networks
2. Ensemble methods
3. …

Directly model posterior distribution

Use variational inference to avoid computing partition function

$$\min_{q(\theta|\mathcal{D})} D_{KL}(q(\theta|\mathcal{D}) \,||\, p(\theta|\mathcal{D}))$$

Challenge: can be difficult to express rich distributions



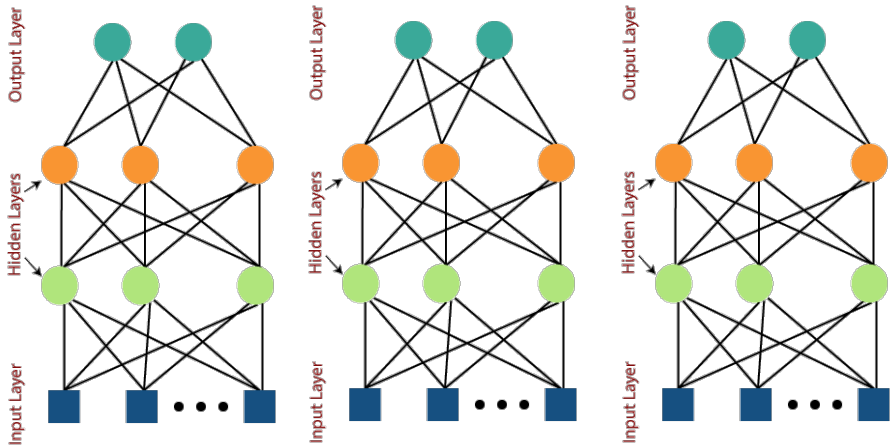
How might we measure uncertainty?

$$p(\theta|\mathcal{D})$$

Difficult to estimate directly!

1. Bayesian neural networks
2. Ensemble methods
3. …

Learn an ensemble of models



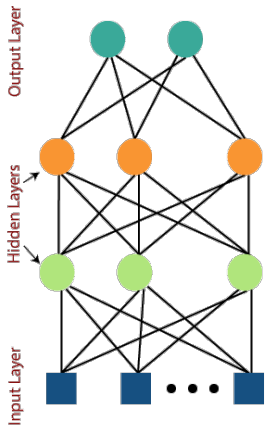
Low data regime → high ensemble variance

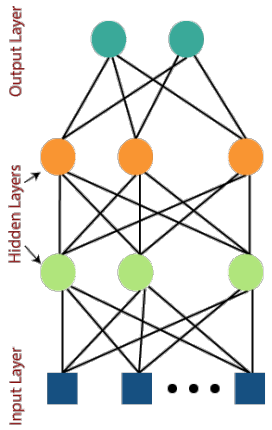
Approximate posterior

Easier and more expressive than BNNs!

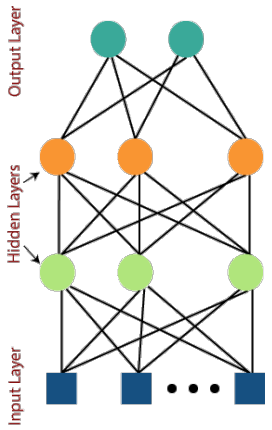
Model Based RL – Learning Ensembles of Dynamics Models

Learn ensembles of dynamics models with MLE rather than a single model





...

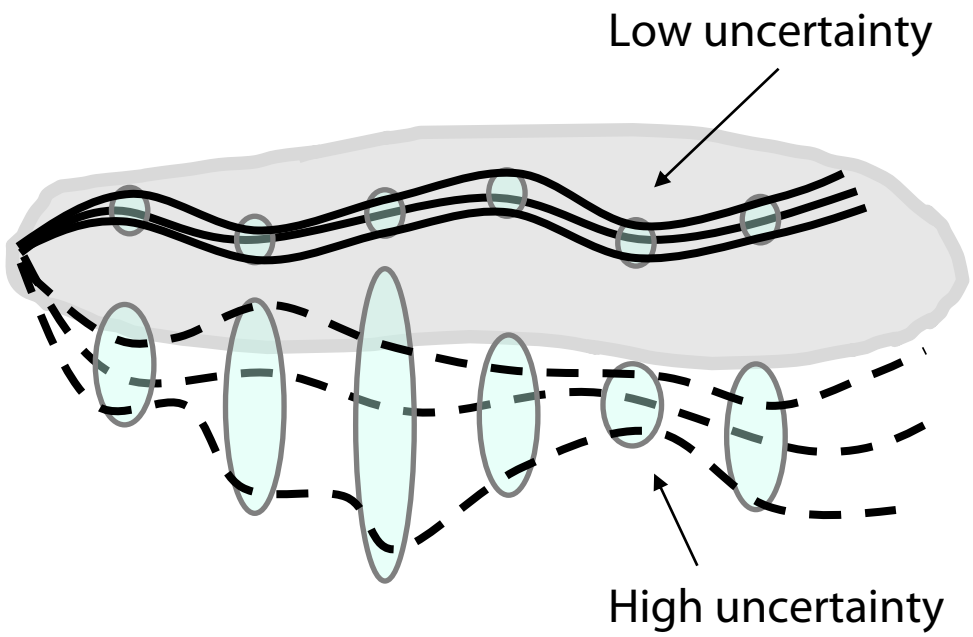


$$\max_\theta \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right] \quad \max_\theta \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right] \quad \max_\theta \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right]$$

Learn ensembles by either subsampling the data or having different initializations

Model Based RL – Integrating Uncertainty into MBRL (v2)

Take expected value under the uncertain dynamics



Expected value over ensemble

$$\arg \max_{(a_0^j, a_1^j, \ldots, a_T^j)_{j=1}^N} \sum_{i=1}^{K} \sum_{t=0}^{T} r((\hat{s}_t^j)^i, a_t^j)$$

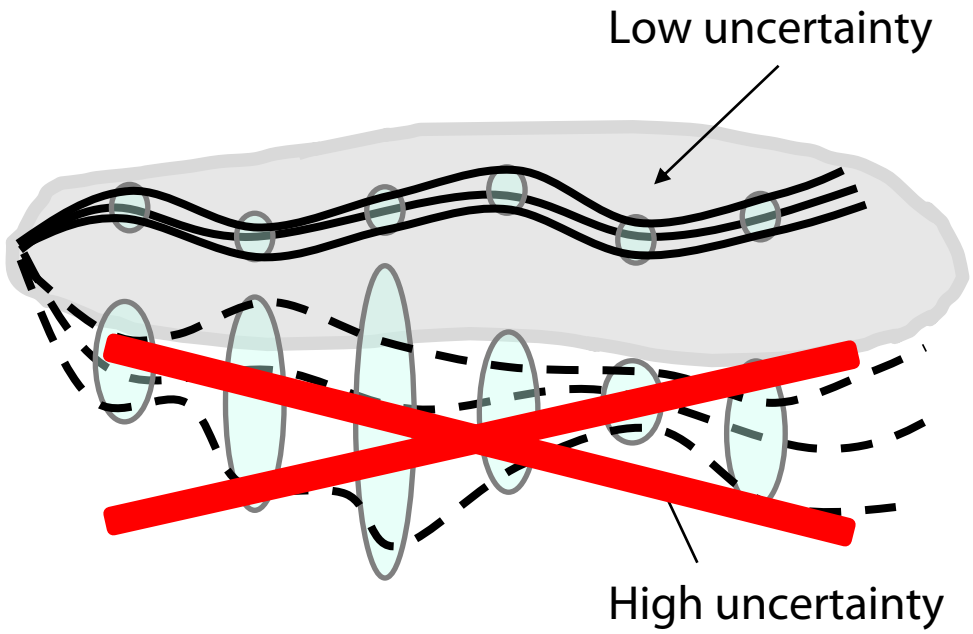
$$(\hat{s}_{t+1}^j)^i \sim \hat{p}_{\theta_i}(.|(\hat{s}_t^j)^i, a_t^j)$$

Can also swap which ensemble element is propagated at every step or just pick randomly amongst them

Avoids overly OOD settings since the expected reward is affected by uncertainty

Model Based RL – Integrating Uncertainty into MBRL (v2)

Take **pessimistic** value under the uncertain dynamics



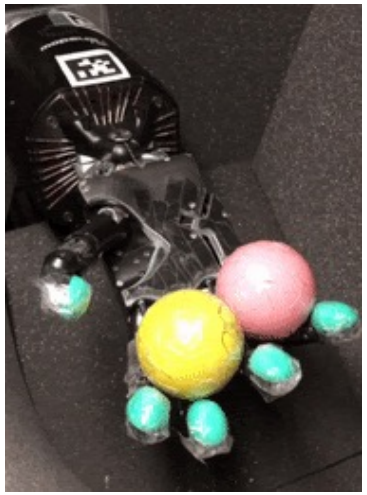
Penalize ensemble variance

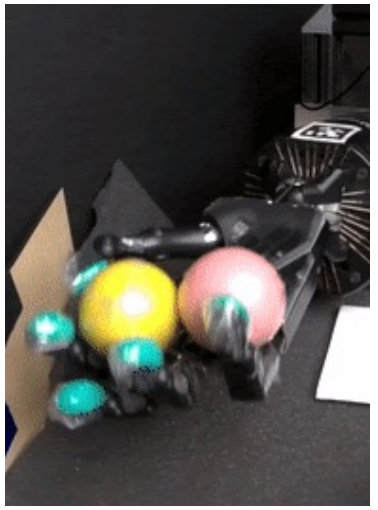
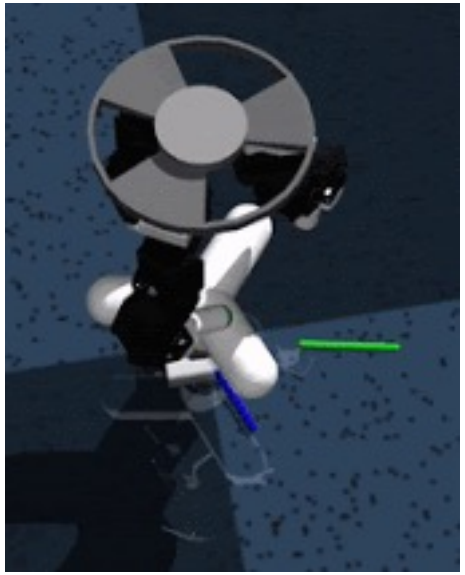
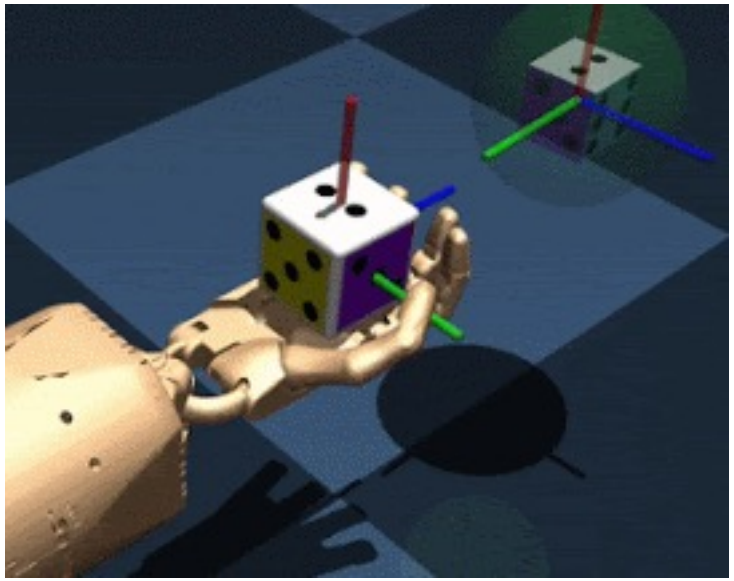
$$\arg \max_{(a_0^j, a_1^j, \ldots, a_T^j)_{j=1}^N} \sum_{i=1}^{K} \sum_{t=0}^{T} r((\hat{s}_t^j)^i, a_t^j) - \lambda \mathrm{Var}((\hat{s}_t^j)^i)$$

$$(\hat{s}_{t+1}^j)^i \sim \hat{p}_{\theta_i}(.|(\hat{s}_t^j)^i, a_t^j)$$

Avoids overly OOD settings since these states are explicitly penalized

Does this work?



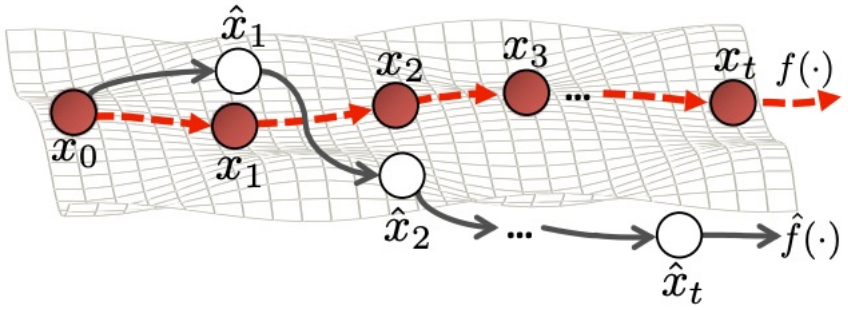






How might we deal with compounding error?

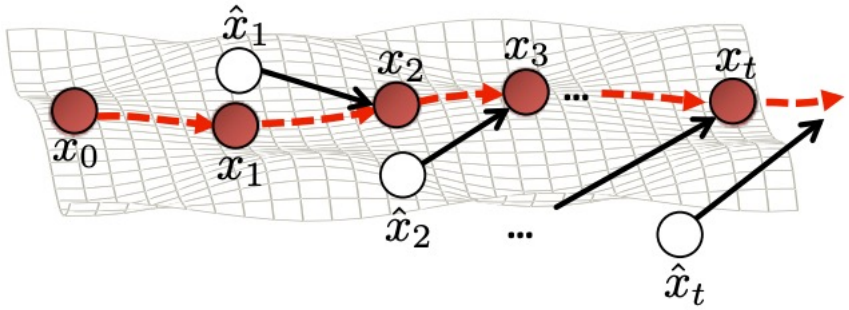
Idea 3: Cast this as an imitation learning problem

→ Reuse ideas from DAgger!

Compounding error



Synthetically generative corrective labels



Can help to correct model predictions with “feedback”

Can run into issues if the synthetic labels conflict with true data

Lecture outline

The Anatomy of Model-Based Reinforcement Learning

↓

Model based RL v0 → random shooting + MPC

↓

Model based RL v1 → MPPI + MPC

↓

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

↓

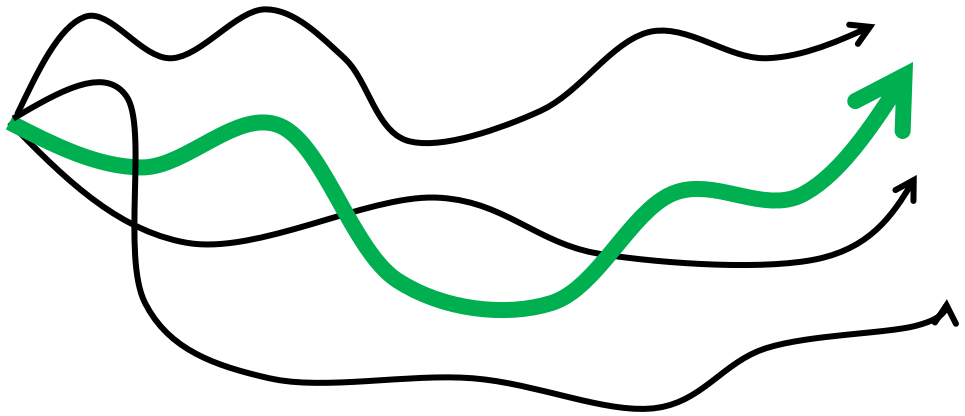
Model based RL v4 → latent space models with images

↓

Model based RL v5 → From MPPI to MCTS

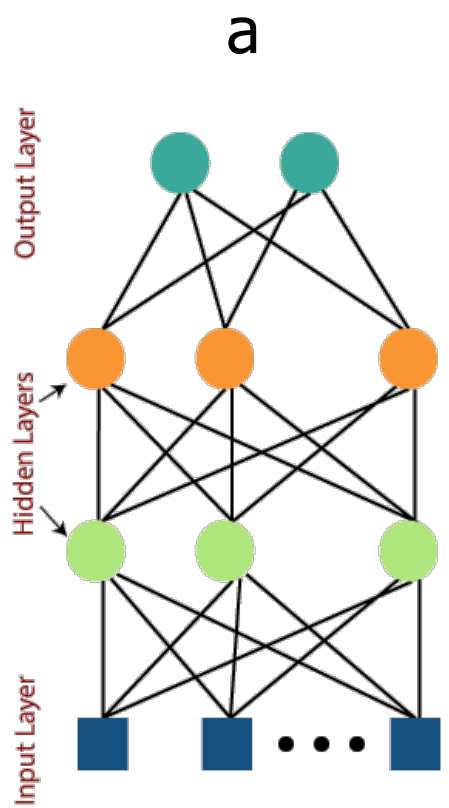
What might be the issue?

Huge number of samples
needed to reduce variance



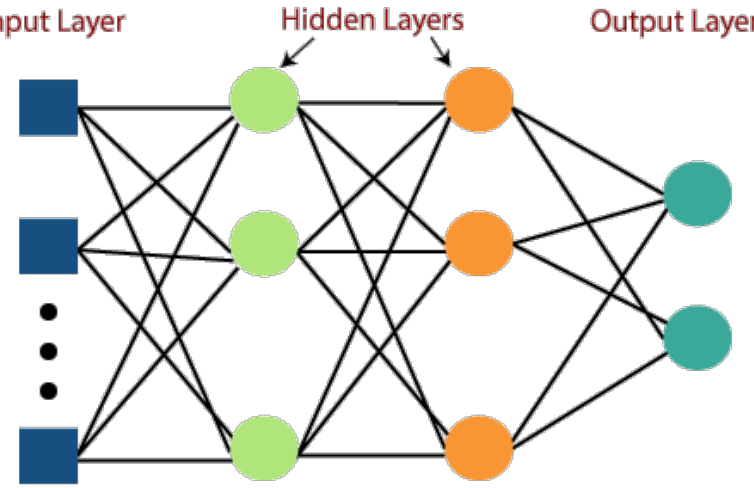
Extremely slow, hard to run in real time

Amortize planning
into a policy

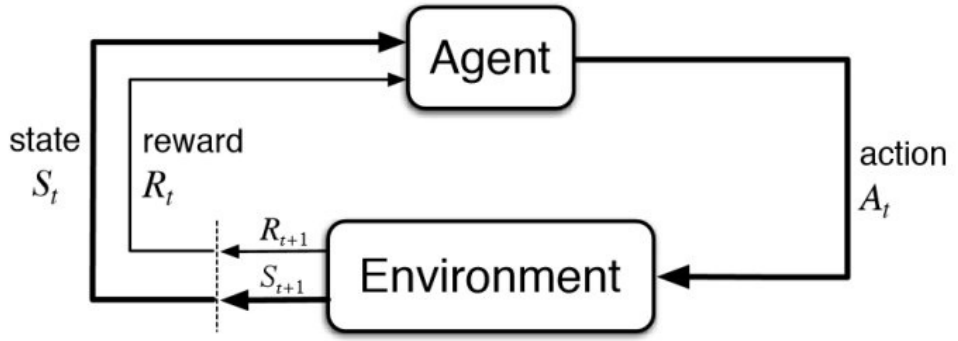


Speeding Up Model-Based Planning

$$\max_{\theta} \mathbb{E}_{(s,a,s') \sim \mathcal{D}} \left[\log \hat{p}_\theta(s' | s, a) \right]$$



Use model(s) to generate data for policy optimization



Can use PG or off-policy!

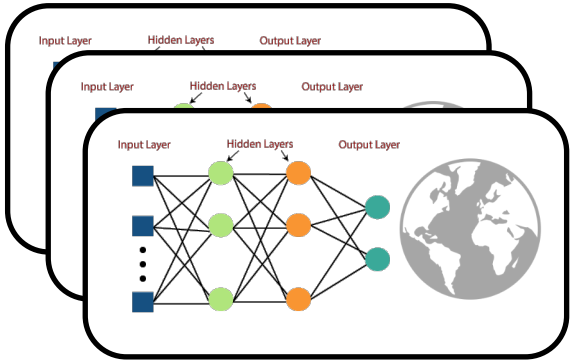
Generating Data for Policy Optimization

Learn models

Add Fake Sampled Data to Buffer

Policy Optimization

Train time



$\mathcal{D}$

$$\min_{\phi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left[Q_\phi^\pi(s_t, a_t) - (r(s_t, a_t) + \max_{a_{t+1}} \left[Q_{\bar{\phi}}(s_{t+1}, a_{t+1}) \right]) \right]^2 \right]$$

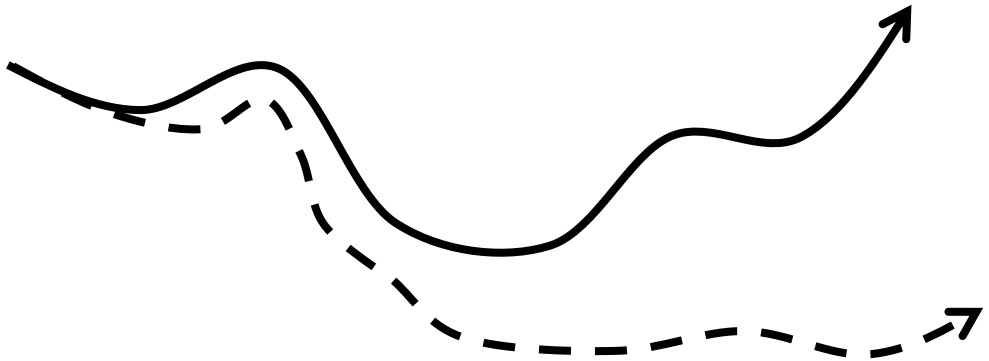
Rollout in environment

π_θ

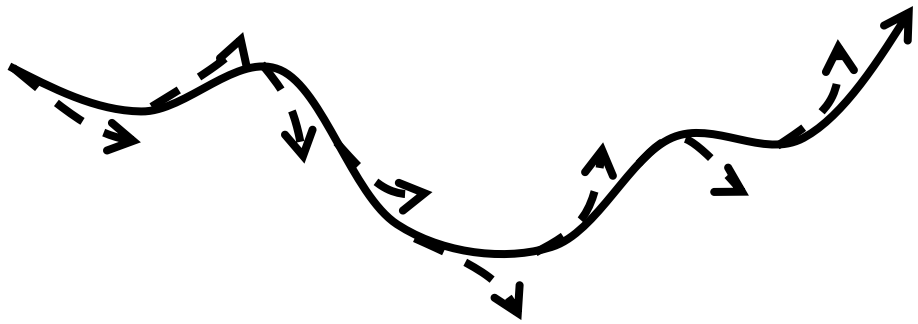
Test time

π_θ

What matters in generating data from models?



Long horizon rollouts can deviate

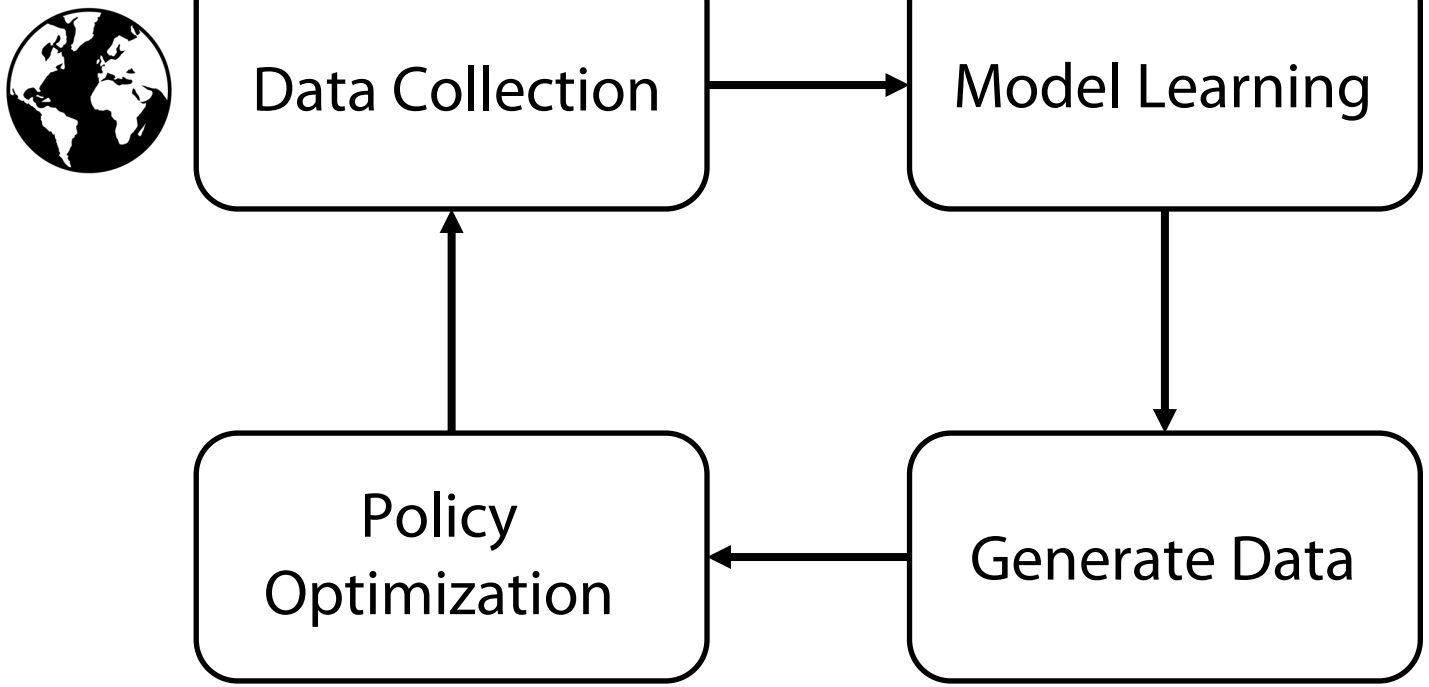


Short horizon rollouts deviate far less

Balance between off-policy coverage and compounding error

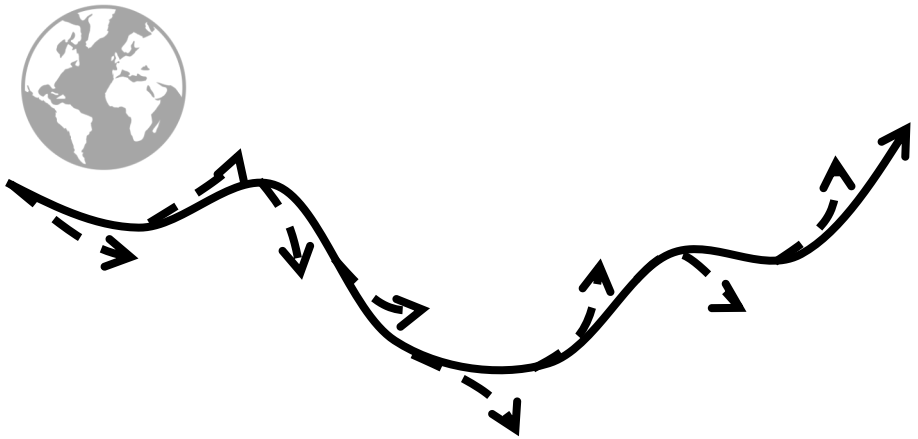
More at https://arxiv.org/abs/1906.08253

Model Based RL – Using Models for Policy Optimization (v3)



Maximum likelihood supervised Learning

$$\max_{\theta} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right]$$



$$\min_{\phi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left[Q^{\pi}_{\phi}(s_t,a_t) - (r(s_t,a_t) + \max_{a_{t+1}} \left[Q_{\bar{\phi}}(s_{t+1},a_{t+1})\right])\right]^2\right]$$

More expensive/harder at training time, faster at test time

Does this work?

Lecture outline

The Anatomy of Model-Based Reinforcement Learning

↓

Model based RL v0 → random shooting + MPC

↓

Model based RL v1 → MPPI + MPC

↓

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

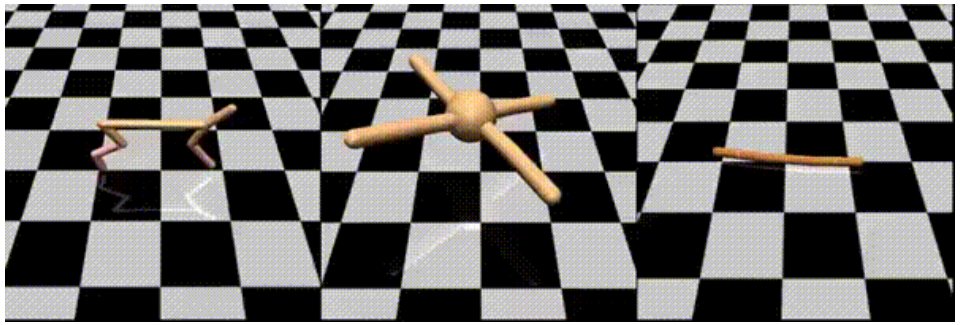
↓

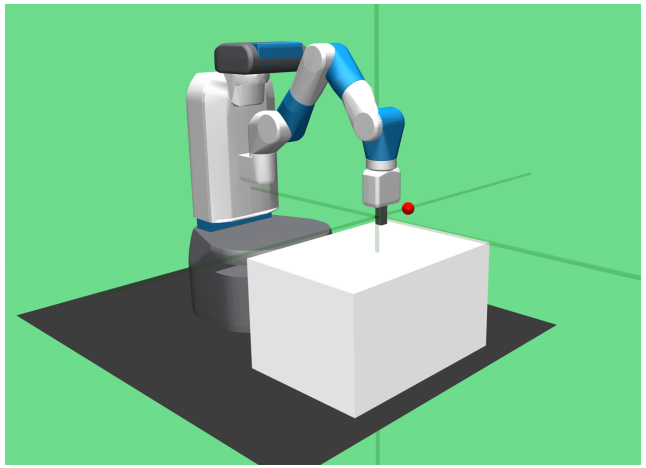
Model based RL v4 → latent space models with images

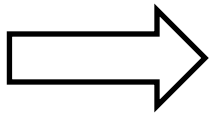
↓

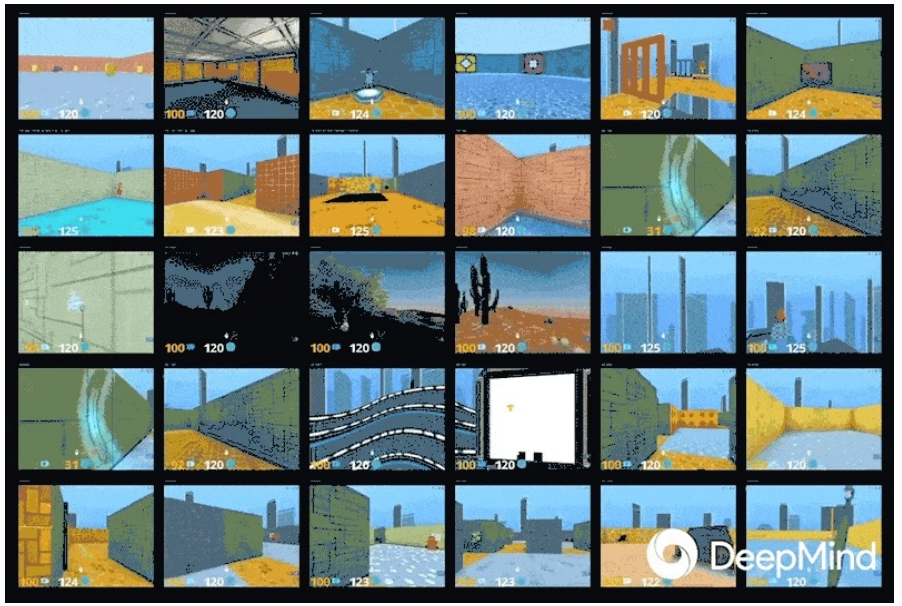
Model based RL v5 → From MPPI to MCTS

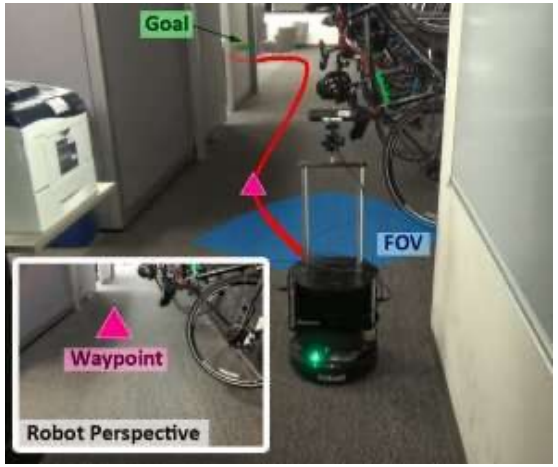
What about images?









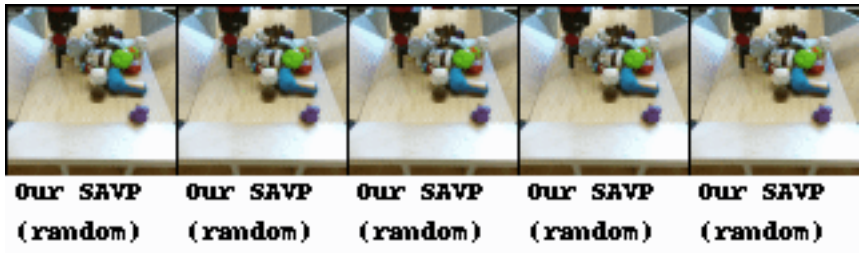


State based domains

Image based domains

Why is learning from images hard?

Generative modeling is videos, challenging to model multimodal correlated predictions

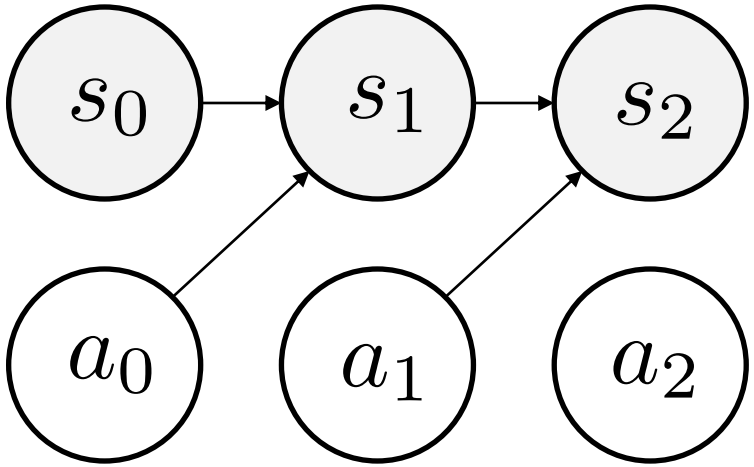


Partially observable!

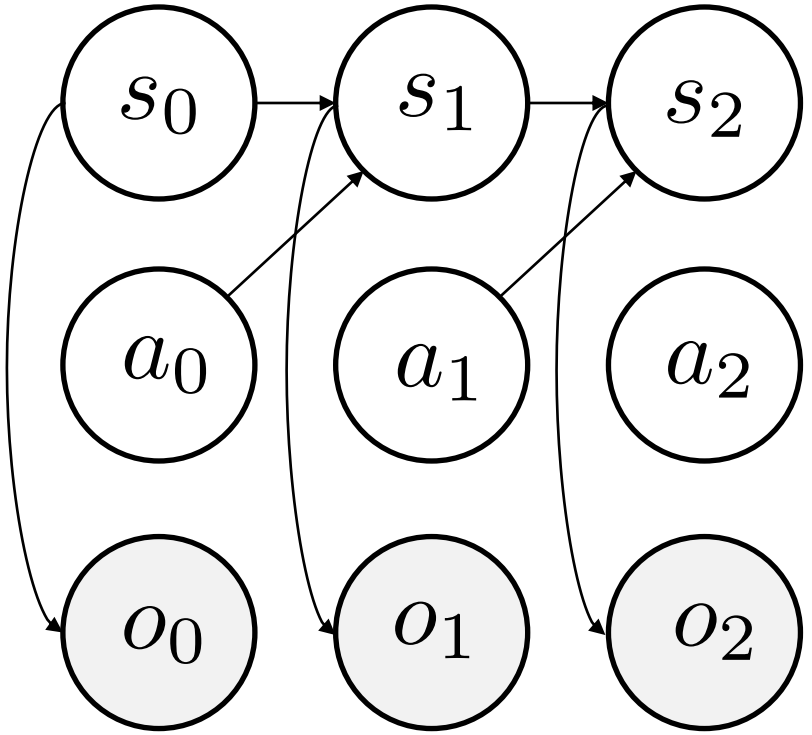
Long horizon predictions in video space can be challenging!

Model Based RL – Latent Space Models for Image Based RL (v4)

Fully observed – Markovian case



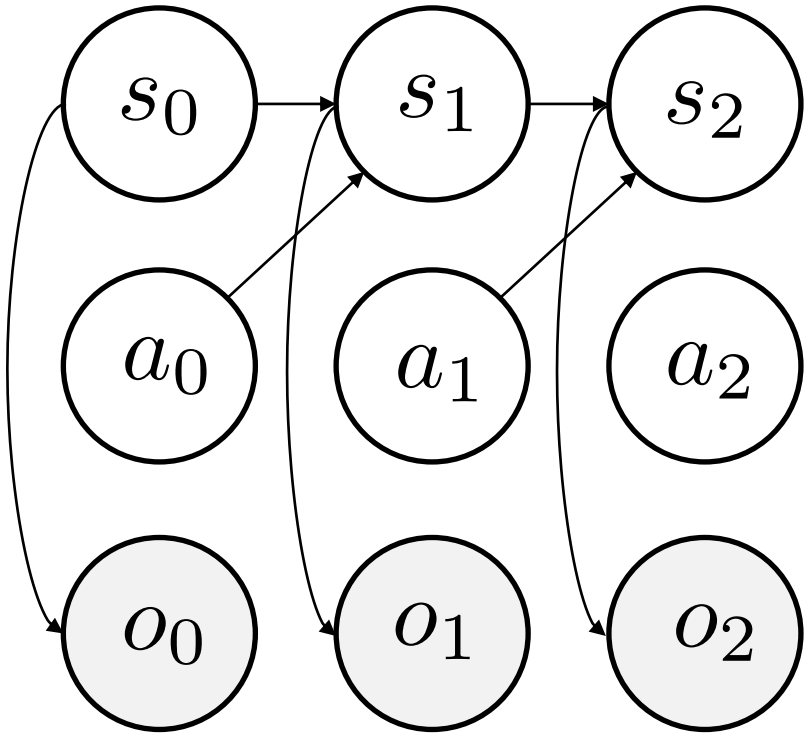
Partially observed – Non-Markovian case



If we can infer latent state and learn dynamics, then we can plan in a much smaller space

How do we infer latent state and learn dynamics in this space?

How do we **train** latent space models?



Learn latent encoder to infer latent state from observations $q_\phi(s_t|o_{1:t})$

Learn action conditioned latent transition model $p_\eta(s_{t+1}|s_t, a_t)$

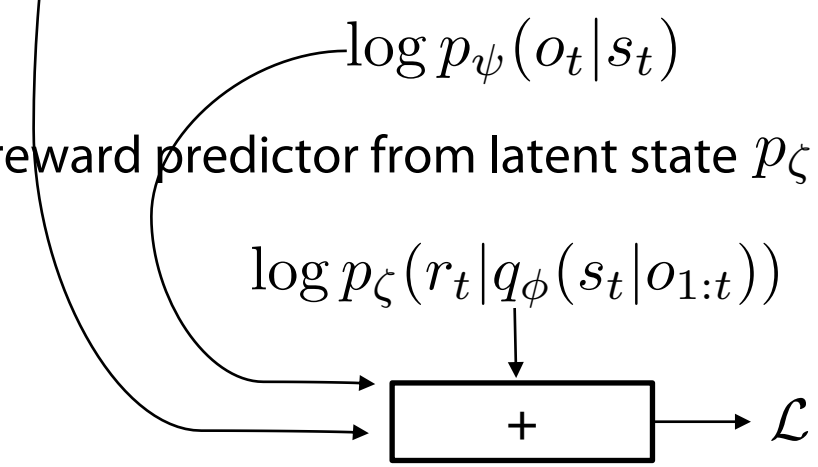
$$\log p_\eta(q_\phi(s_{t+1}|o_{1:t+1})|q_\phi(s_t|o_{1:t}), a_t)$$

Learn latent decoder to reconstruct observations $p_\psi(o_t|s_t)$

$$\log p_\psi(o_t|s_t)$$

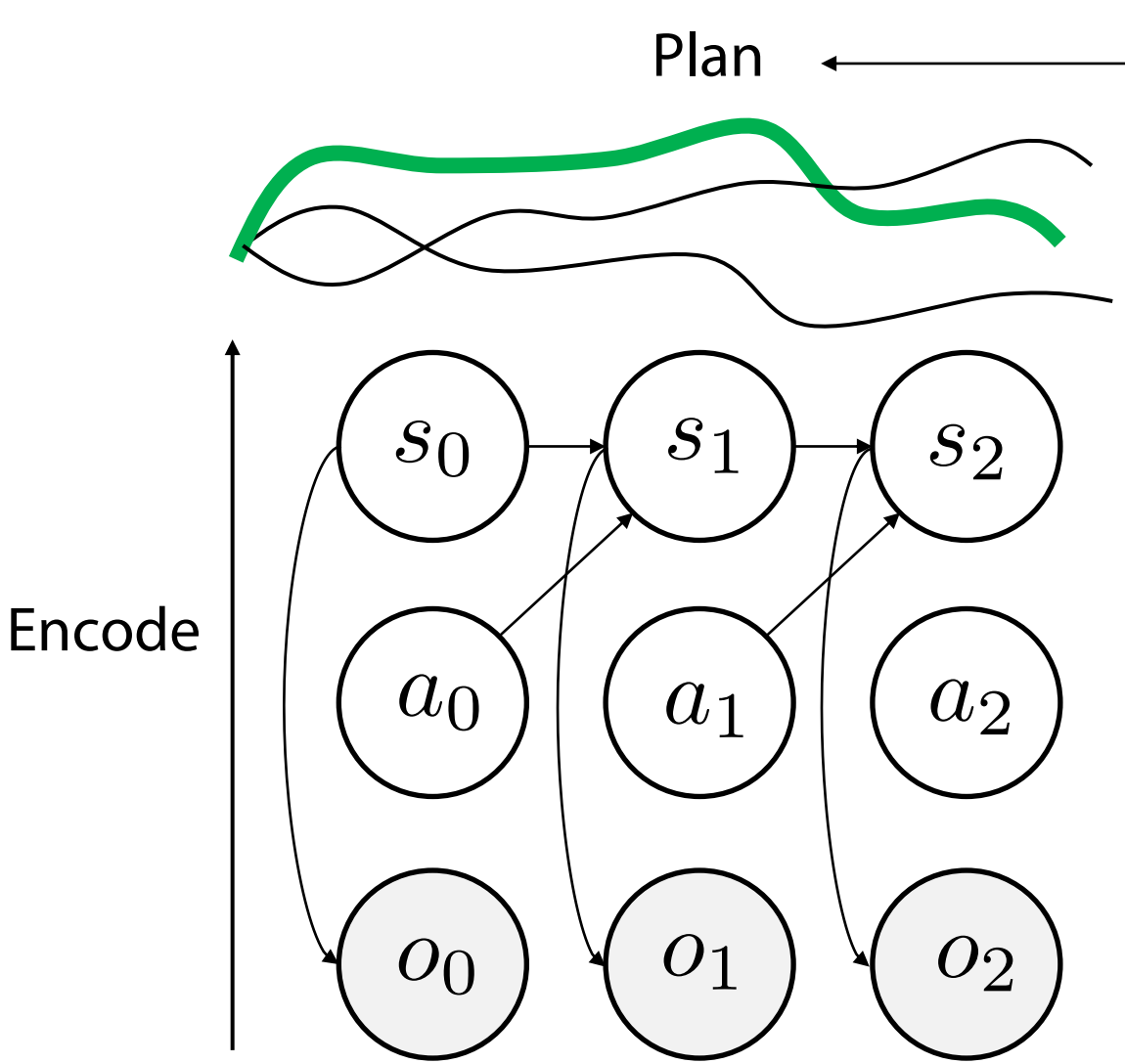
Learn reward predictor from latent state $p_\zeta(r_t|s_t)$

$$\log p_\zeta(r_t|q_\phi(s_t|o_{1:t}))$$



Can derive the whole thing from first principles using variational inference!

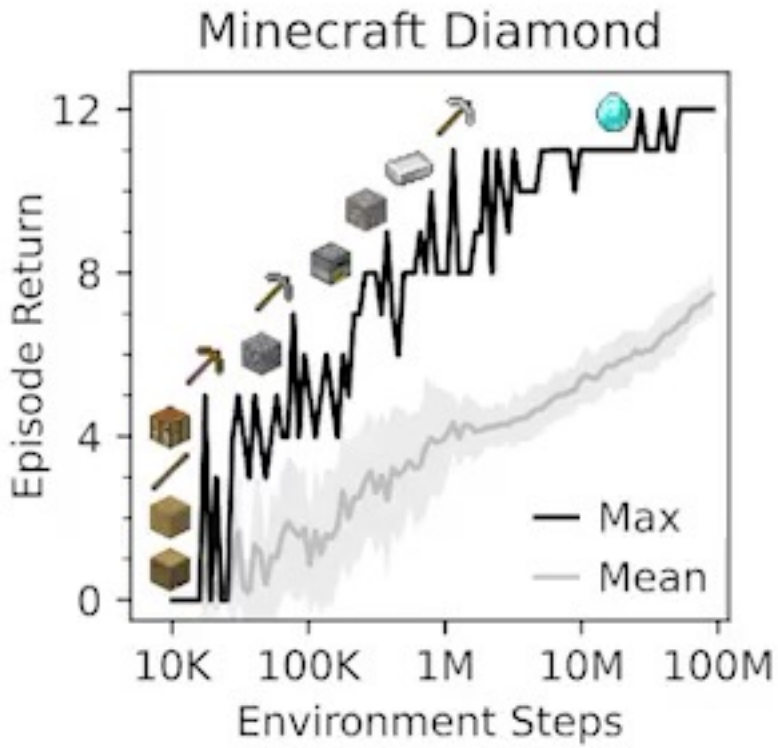
How do we **use** latent space models?



Apply any of the methods from this lecture, just in latent space!

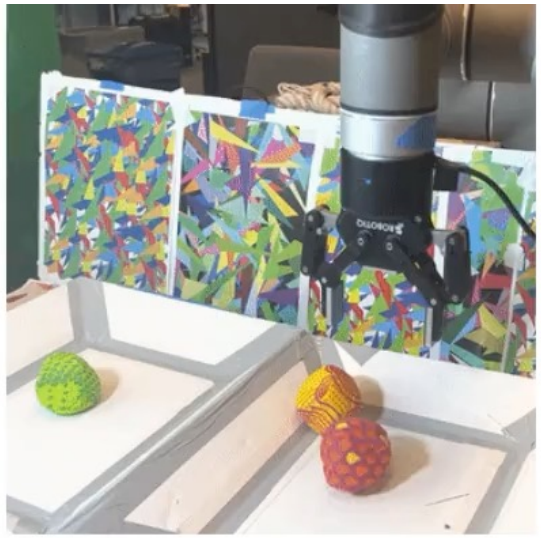
1. Avoids predicting image frames at planning time
2. Scales much better than image prediction
3. Allows for longer horizon predictions

Does this work?

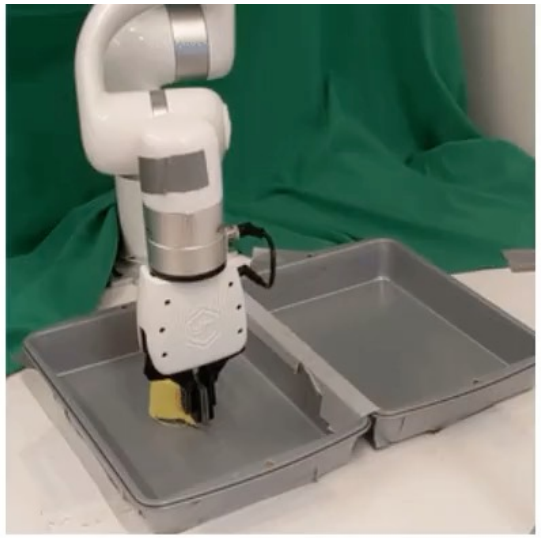


Does this work?

A1 Quadruped Walking



UR5 Multi-Object Visual Pick Place



XArm Visual Pick and Place

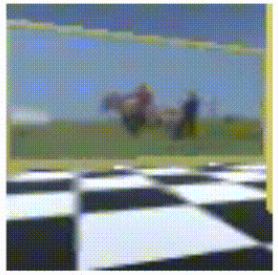
Sphero Ollie Visual Navigation

Training from images in < 1 hour!

When is reconstruction not ideal?

What if reconstruction captures the wrong information?

Reconstruction may not capture decision-relevant information



Agent in a maze with a noisy TV

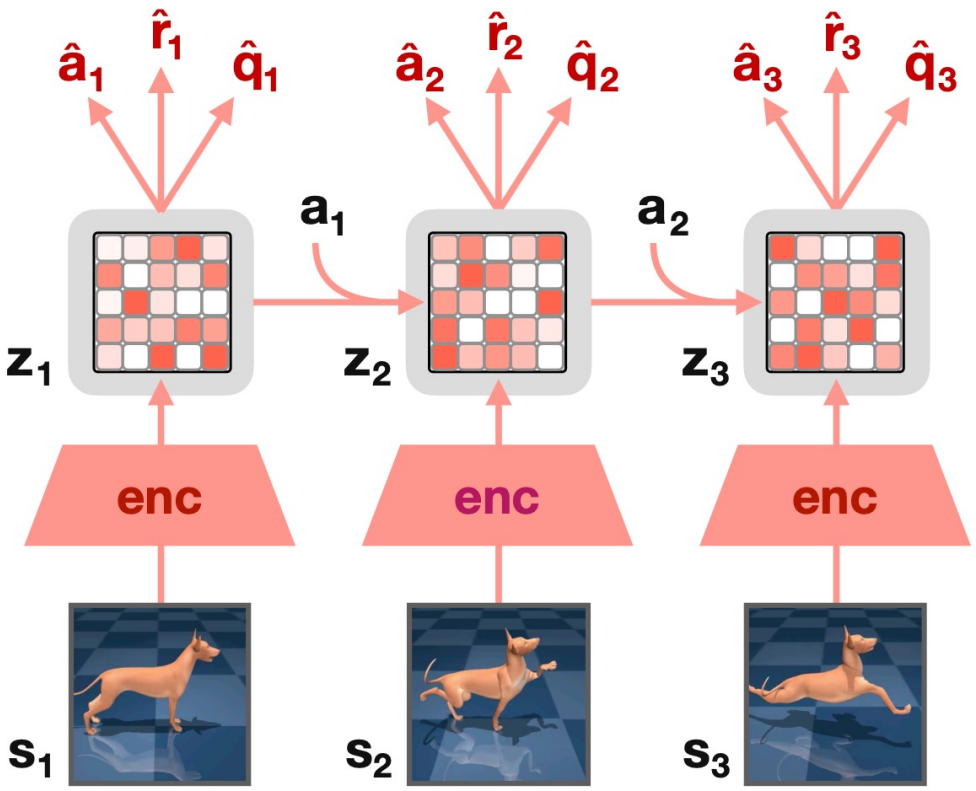
Reconstruction overindexes on pixel accuracy

Large-objects dominate reconstruction objective

Some solutions to avoid reconstruction

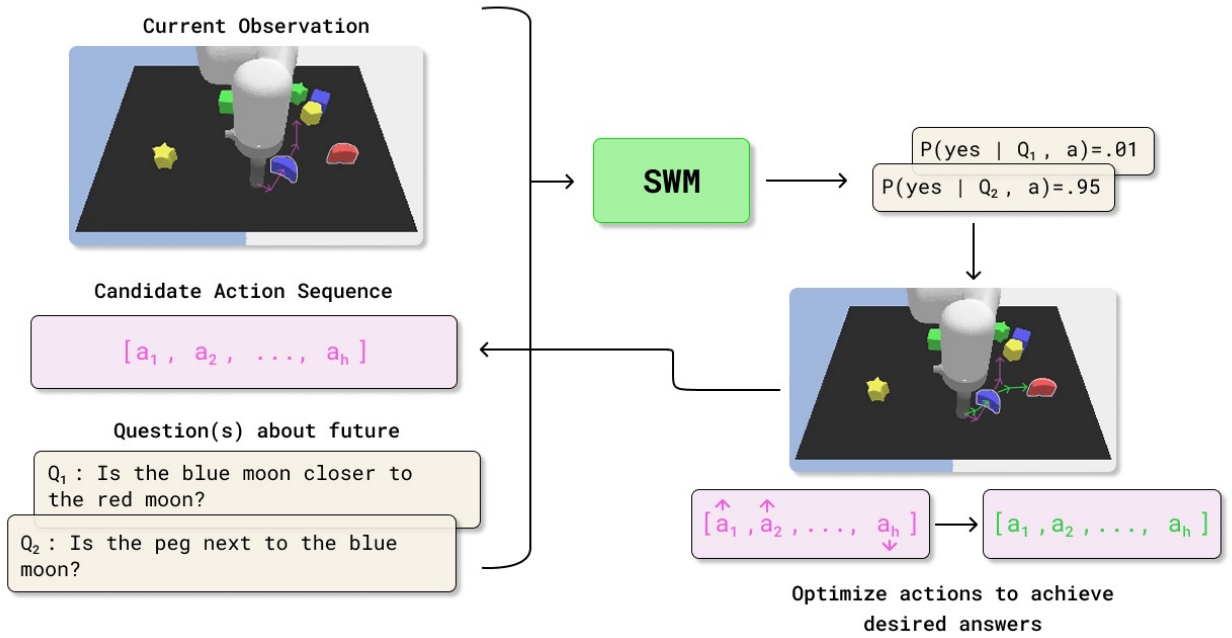
Predict “task-relevant” objectives

Predict values/rewards



TD-MPC2, Hansen et al ‘23

Predict semantics instead of pixels



SWM, Berg et al ‘25

Lecture outline

The Anatomy of Model-Based Reinforcement Learning

↓

Model based RL v0 → random shooting + MPC

↓

Model based RL v1 → MPPI + MPC

↓

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

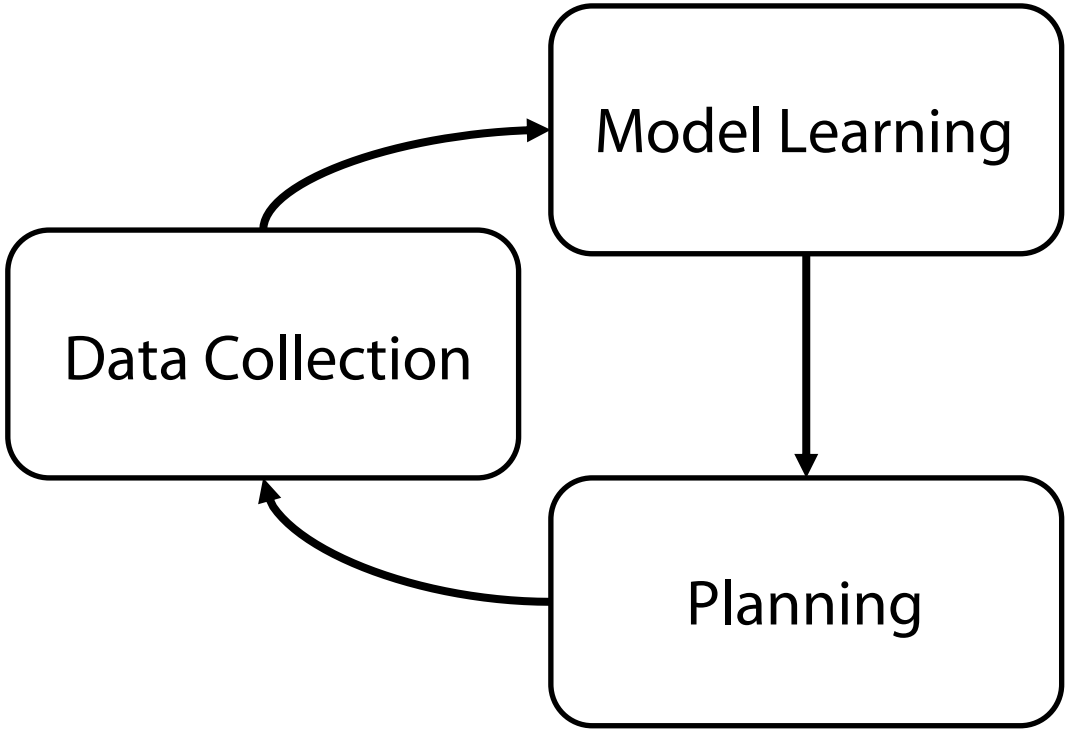
↓

Model based RL v4 → latent space models with images

↓

Model based RL v5 → From MPPI to MCTS

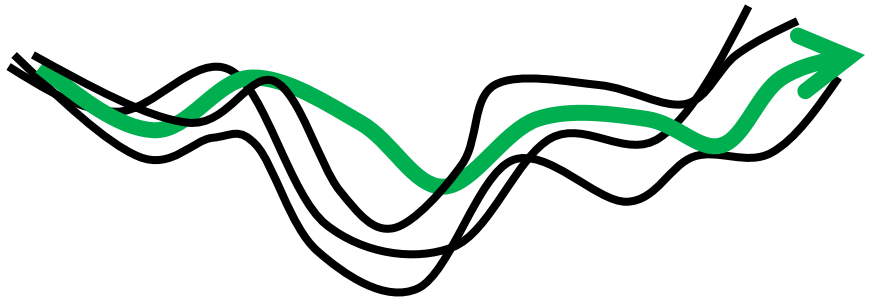
Can we do better than MPPI?



$$\arg \max_{a_0^j, a_1^j, \ldots, a_T^j} \sum_{t=0}^{T} r(\hat{s}_t^j, a_t^j)$$

$$\hat{s}_{t+1}^j \sim \hat{p}_\theta(.|\hat{s}_t^j, a_t^j)$$

$$p(a) \leftarrow p(a) \frac{\exp(\sum_t r(s_t, a_t))}{Z}$$

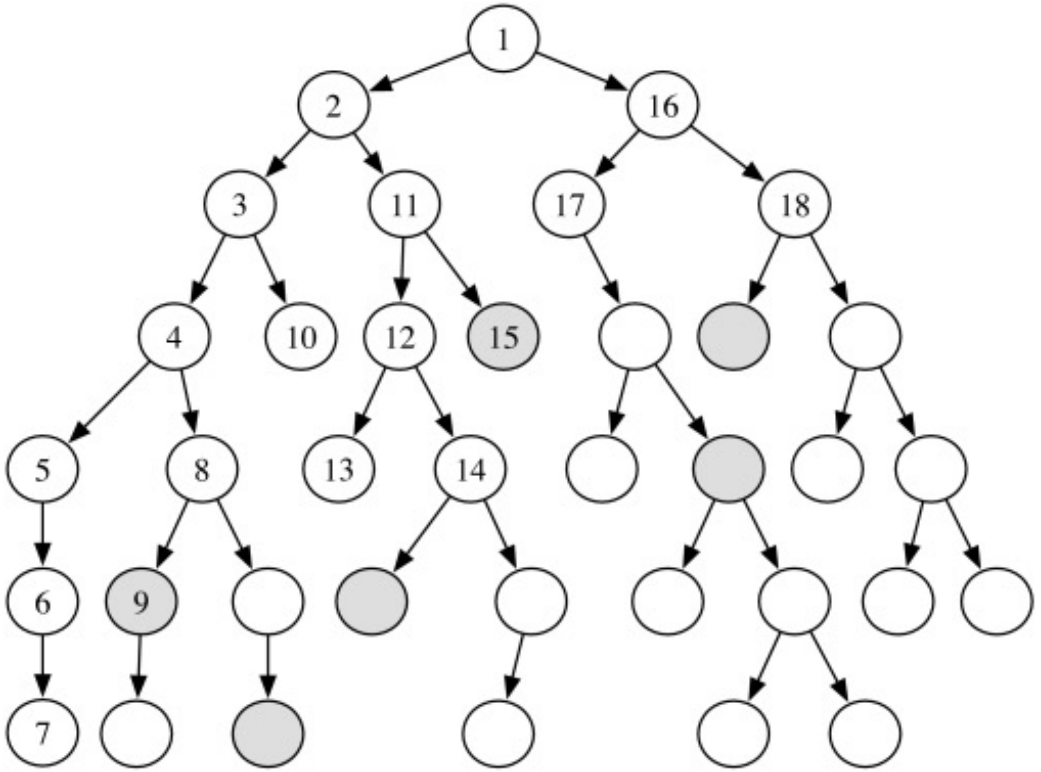


Can we improve on this planning strategy for discrete action spaces?

Discrete action spaces allow for better data structures

Total number of possible sequences for search - $|A|^H$

Are we wasting computation?



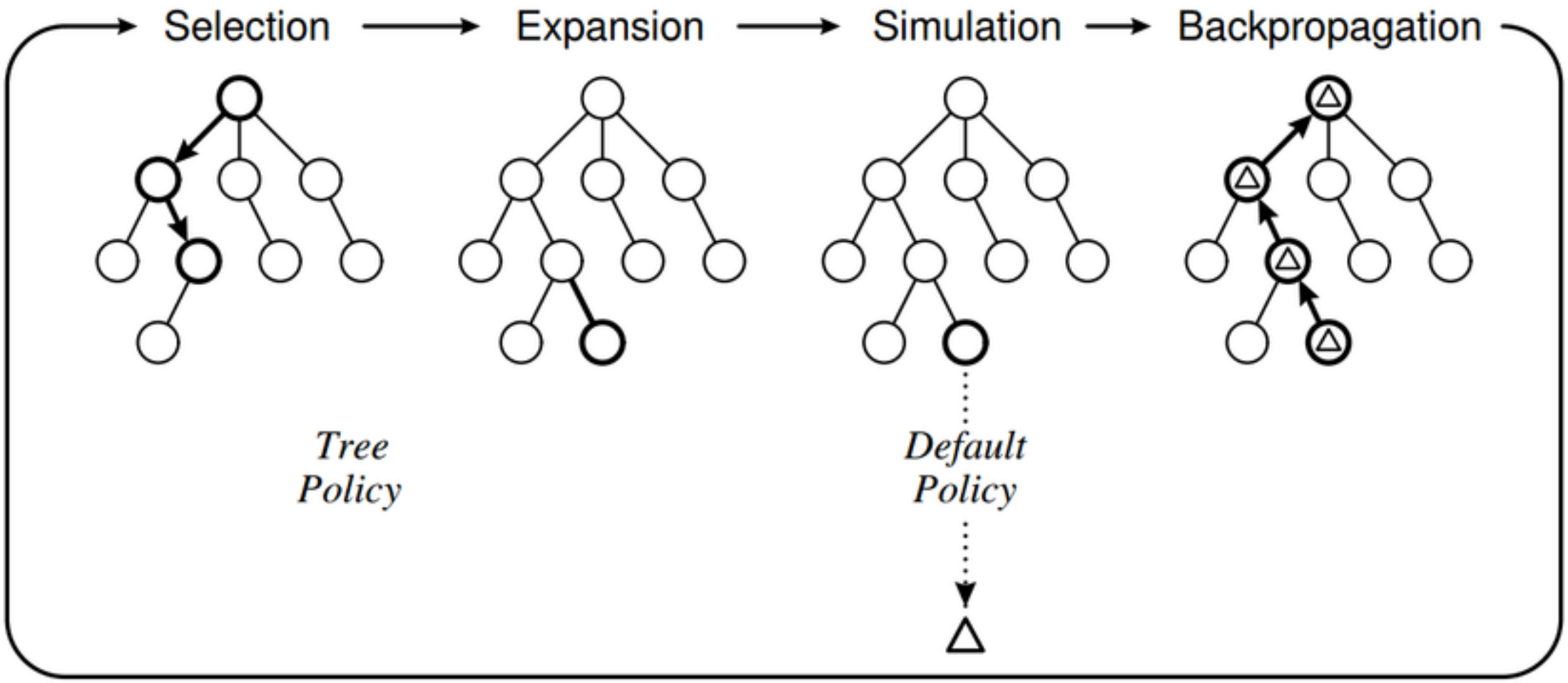
Tree structures can allow you to focus your computational effort

⇩

Can't build whole tree, how to approximate?

Monte-Carlo Tree Search

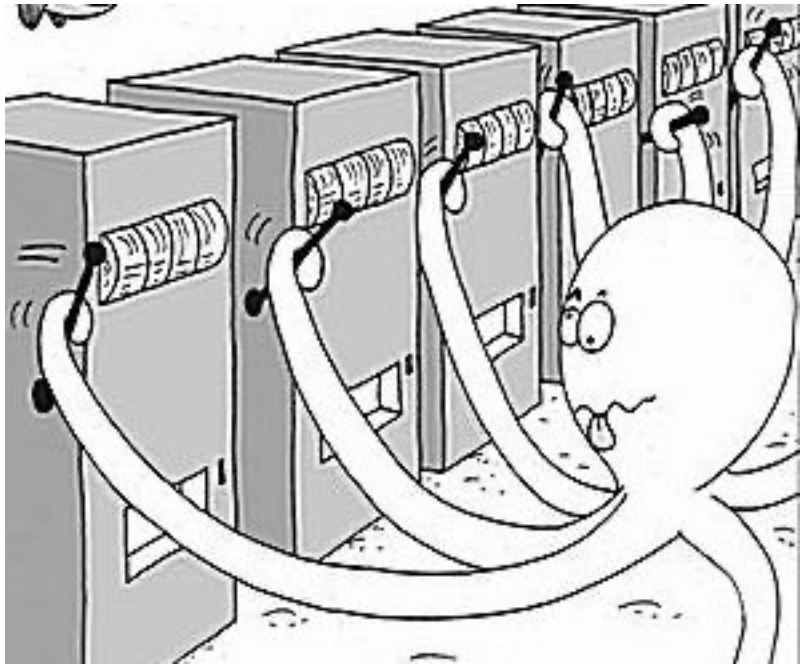
Easy way to approximate tree search for large tree structures, with known/learned models

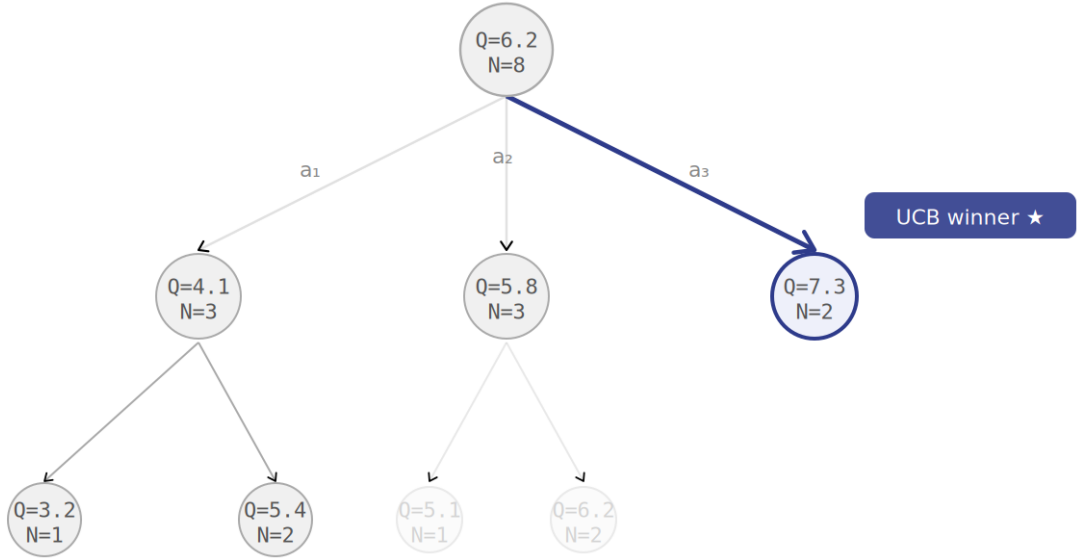


(https://www.cs.swarthmore.edu/~mitchell/classes/cs63/f20/reading/mcts.html)

Monte-Carlo Tree Search: Selection

Selection aims to find the best path down the tree to the frontier + some exploration

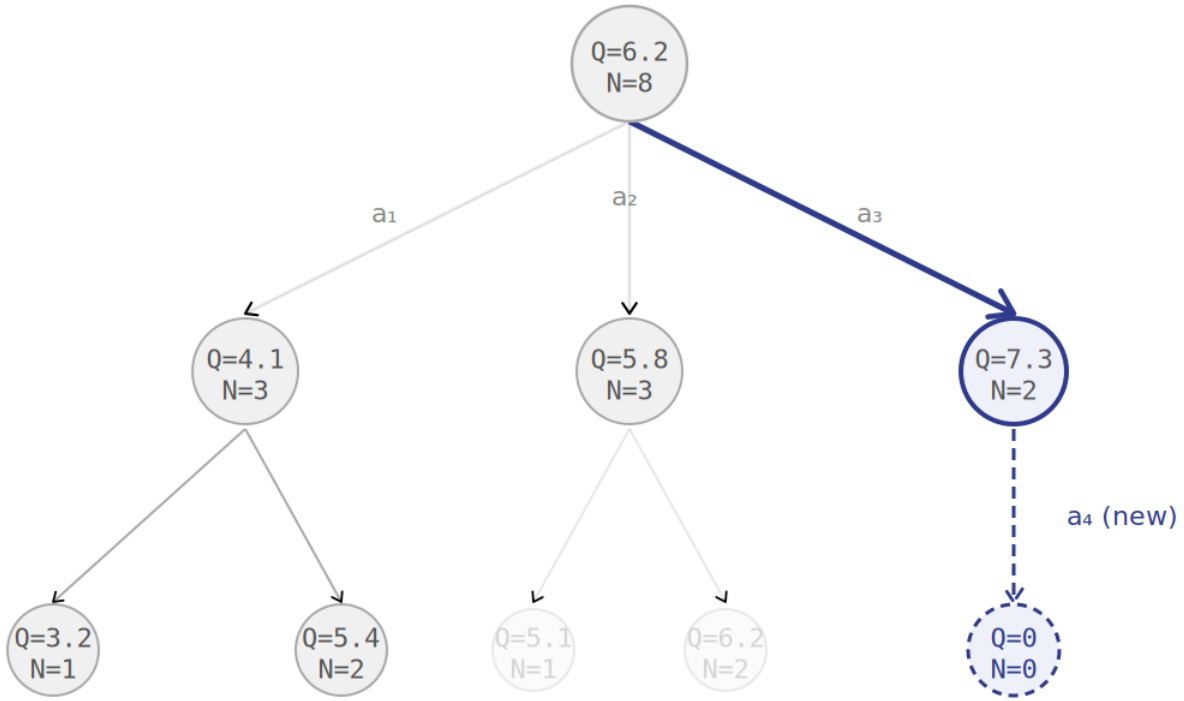




$$a^* = \arg\max_a \left[Q(s,a) + c\sqrt{\frac{\ln N(s)}{N(s,a)}} \right]$$

Monte-Carlo Tree Search: Expansion

Grow the tree by one level

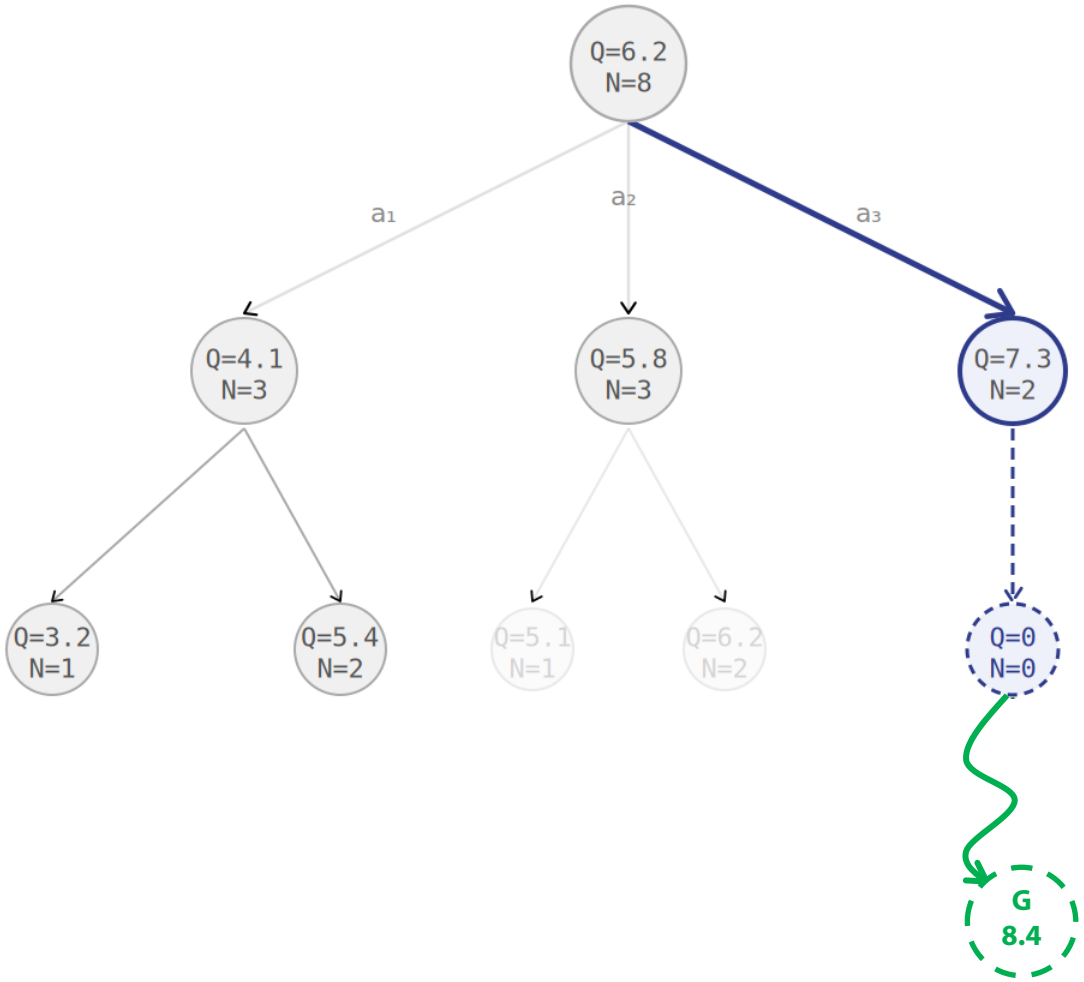


Set Q, N = 0

Updated over time through rollouts

Monte-Carlo Tree Search: Simulation

Tree is too large to construct and keep in memory → Monte-Carlo Approximation!

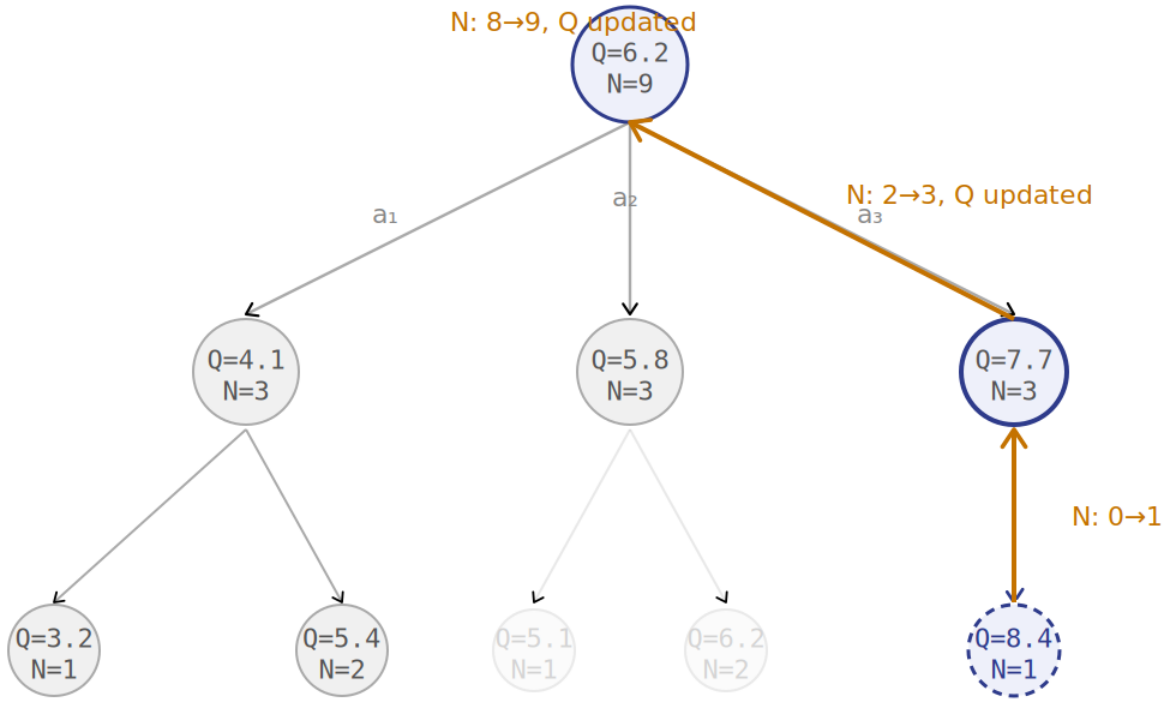


Run samples with policy and approximate value of the remaining tree

Start with random policy
→ Replace with rollouts from π

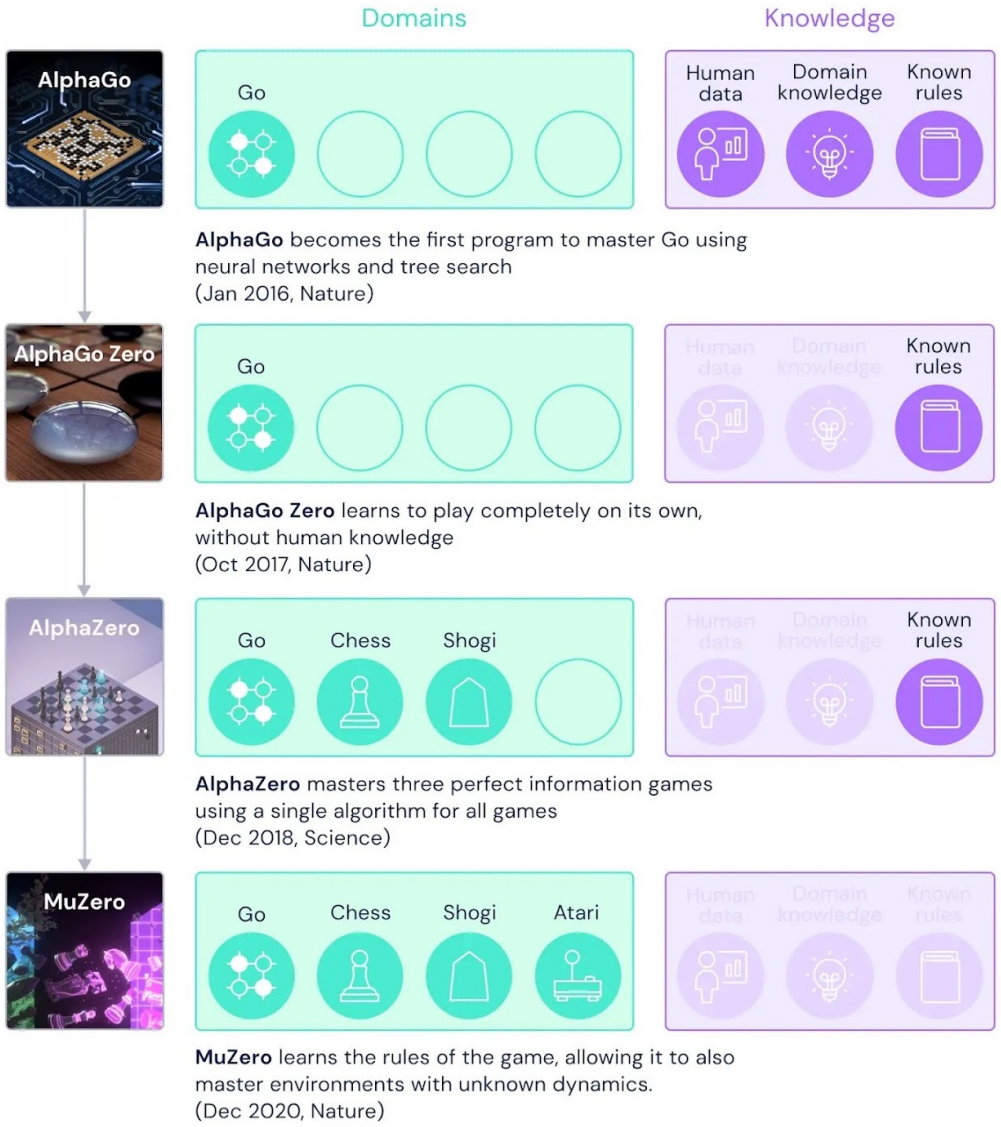
Monte-Carlo Tree Search: Backpropagation

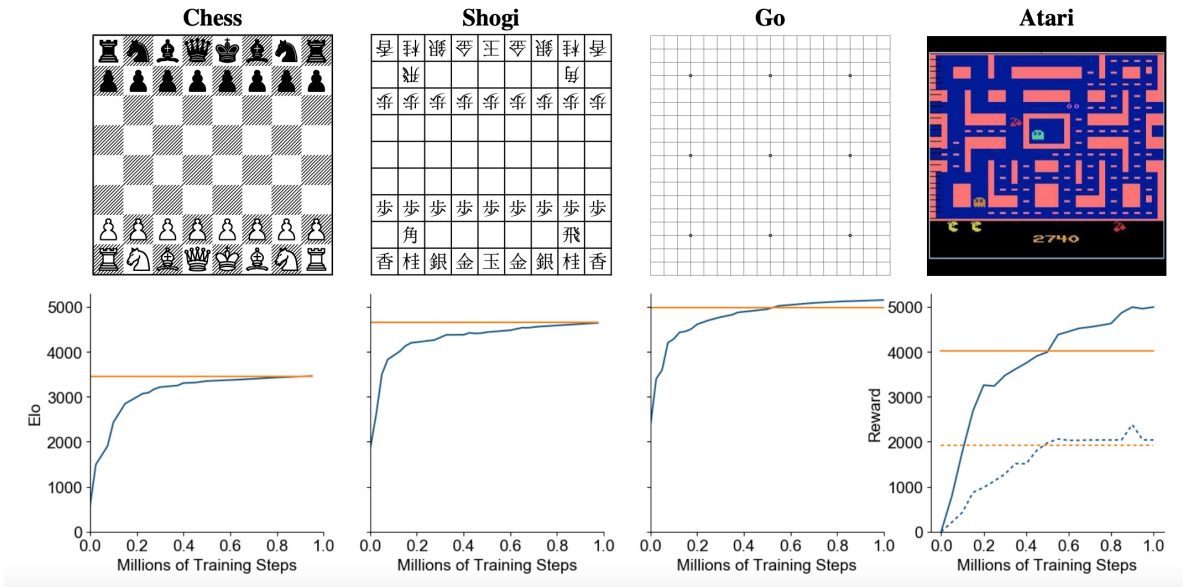
Go backwards and update every node in the tree



Affects future selection decisions

Does this work? → MuZero (2019)





Lecture outline

The Anatomy of Model-Based Reinforcement Learning

↓

Model based RL v0 → random shooting + MPC

↓

Model based RL v1 → MPPI + MPC

↓

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

↓

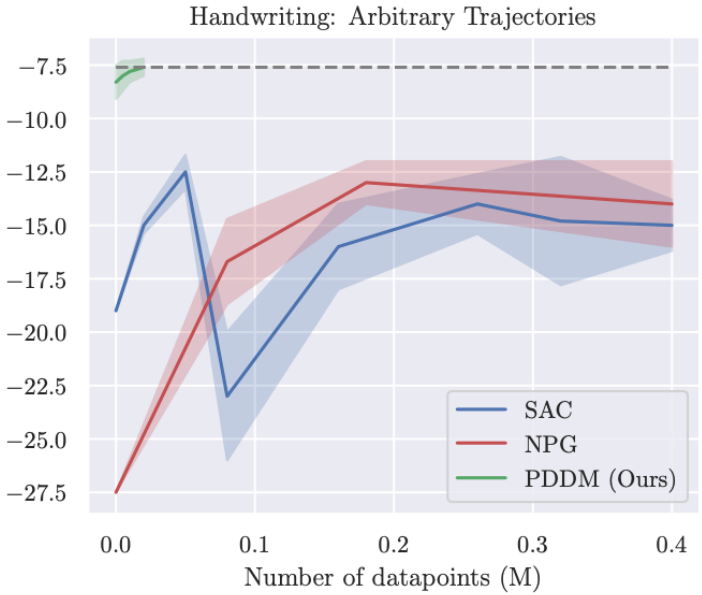
Model based RL v5 → From MPPI to MCTS

Why should you care?

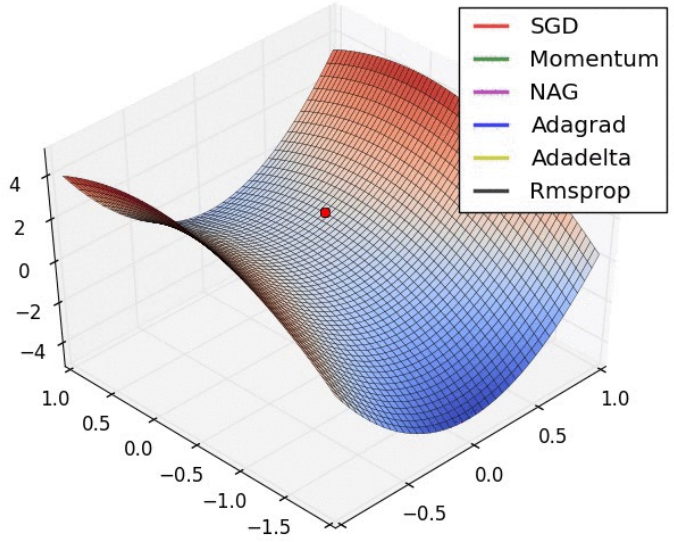
Model based RL **may be** a much more practical path to real world robotics

Transfer/Adaptive

Efficiency



Simplicity



← Likely to be the most future proof one!

Are models really that different than Q-functions?

	Models	Q-functions
Similar	1. Off-policy 2. Models the future Very different than PG methods → on-policy, models current given future	
Different	1. 1-step modeling 2. Models states 3. Can evaluate arbitrary policies 4. Parametric storage of training data	1. Cumulative modeling 2. Models returns 3. Can evaluate only policy π 4. Non-parametric storage of data

Class Structure