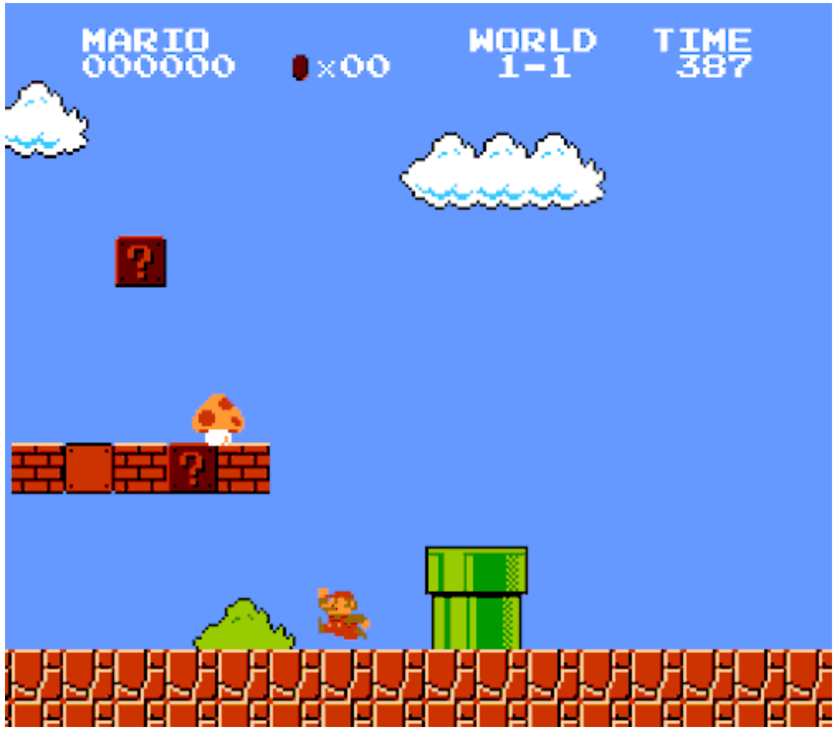
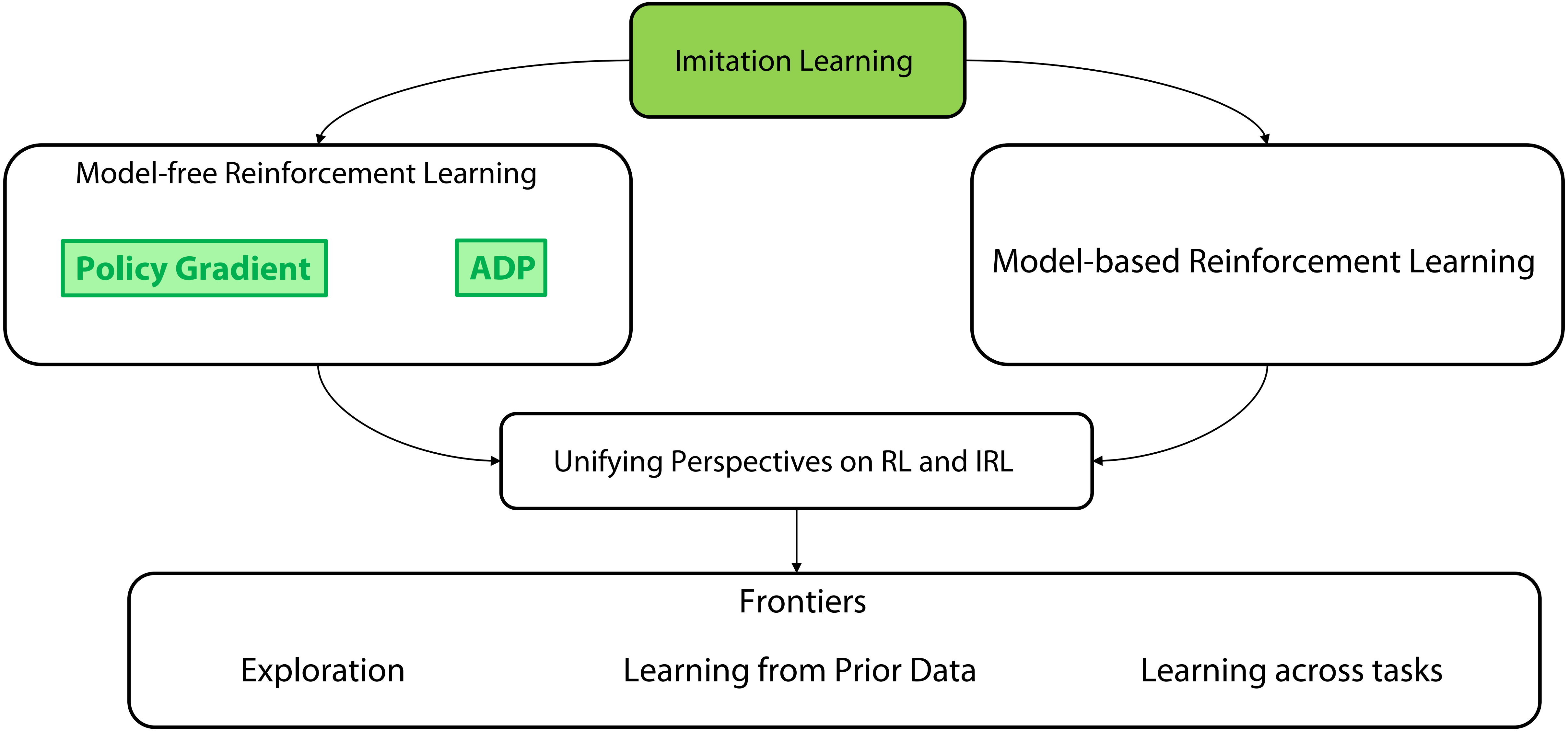
Reinforcement Learning
Spring 2024

Abhishek Gupta
TAs: Patrick Yin, Qiuyu Chen



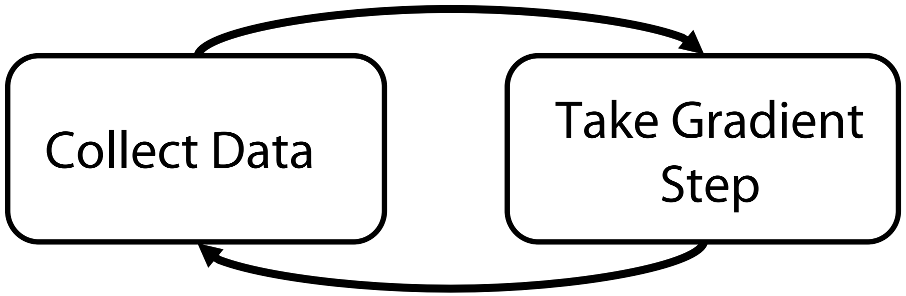
Class Structure



Resulting Algorithm (REINFORCE)

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau$$

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta J(\theta)|_{\theta=\theta_i}$$



REINFORCE algorithm:

On-policy

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Challenges in Policy Gradient

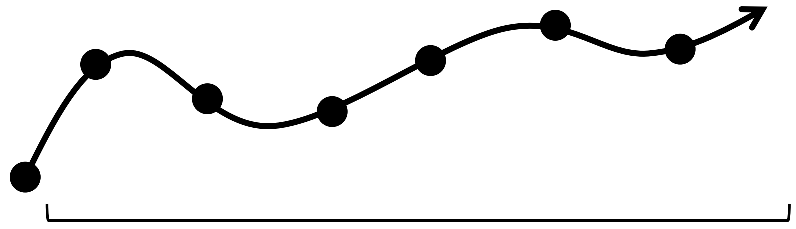
$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau$$

$$\approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

High variance estimator!!

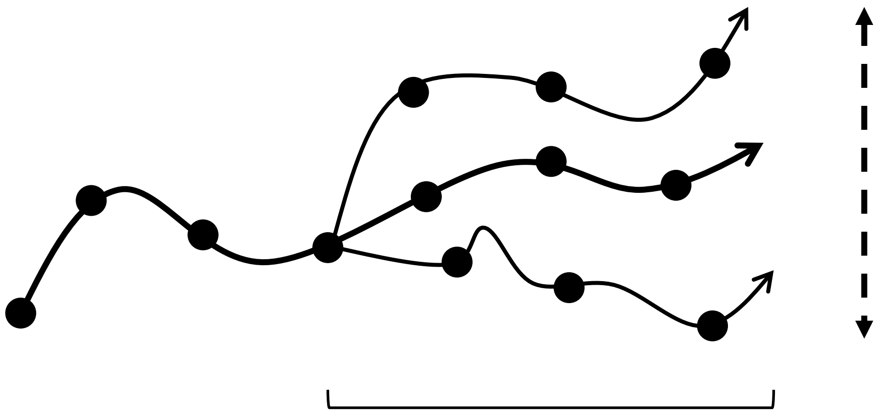
Hard to tell what matters without many samples

<u>What we do</u>



Single sample estimate

<u>What we actually want</u>



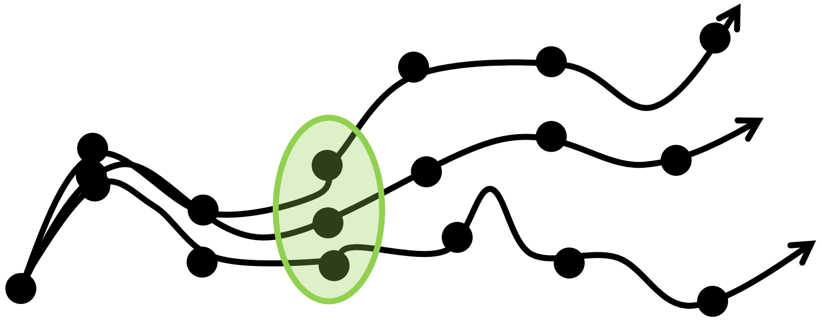
Averaged return estimate

What can we do to lower variance?

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau$$

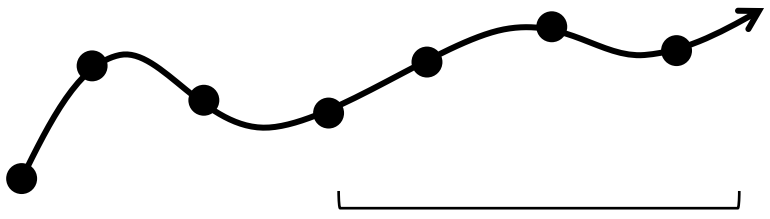
$$\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \underbrace{\sum_{t'=t}^{T} r(s_{t'}^i, a_{t'}^i)}$$

Idea: bundle this across many (s, a) with a function approximator



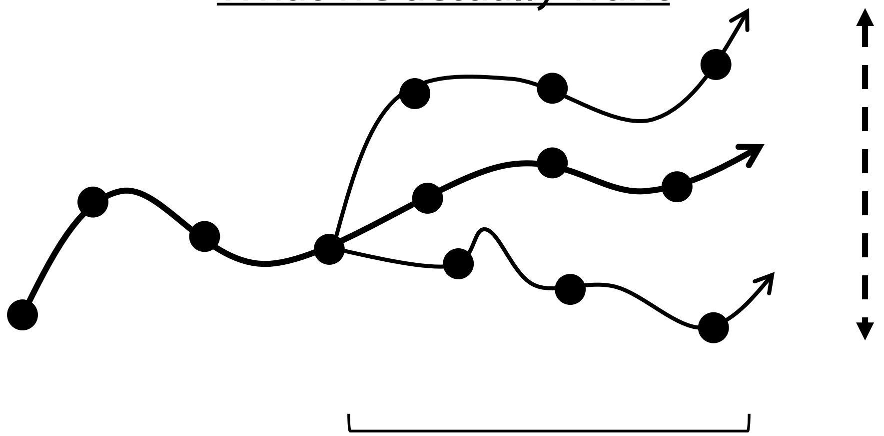
Function approximator bundles return estimates across states

<u>What we do</u>



Single sample estimate

<u>What we actually want</u>

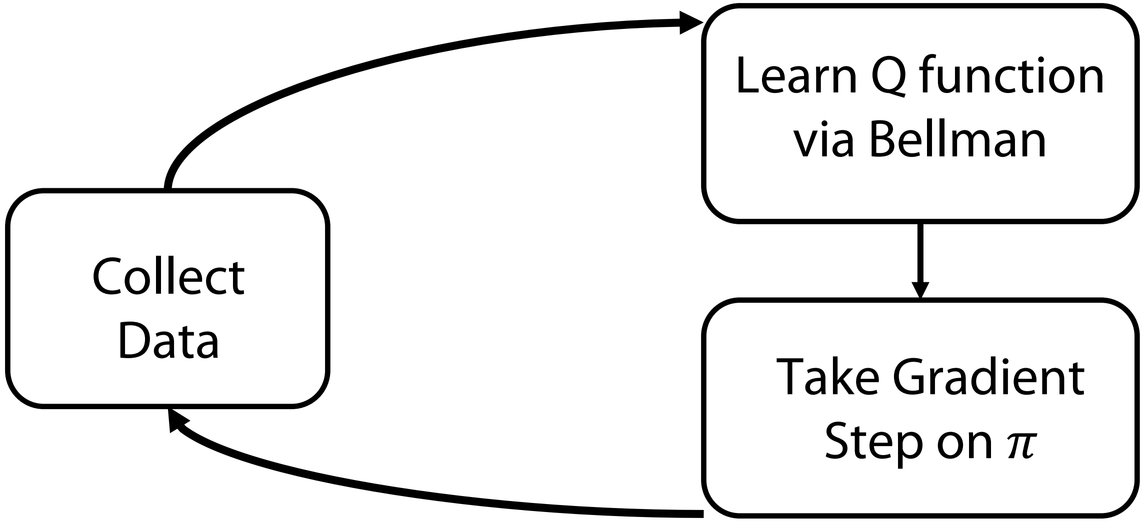


Averaged return estimate

Recap of Off-Policy Reinforcement Learning

Critic: learned via the Bellman update (Policy Evaluation)

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



Lowers variance and is off-policy!

Actor: updated using learned critic (Policy Improvement)

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

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

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

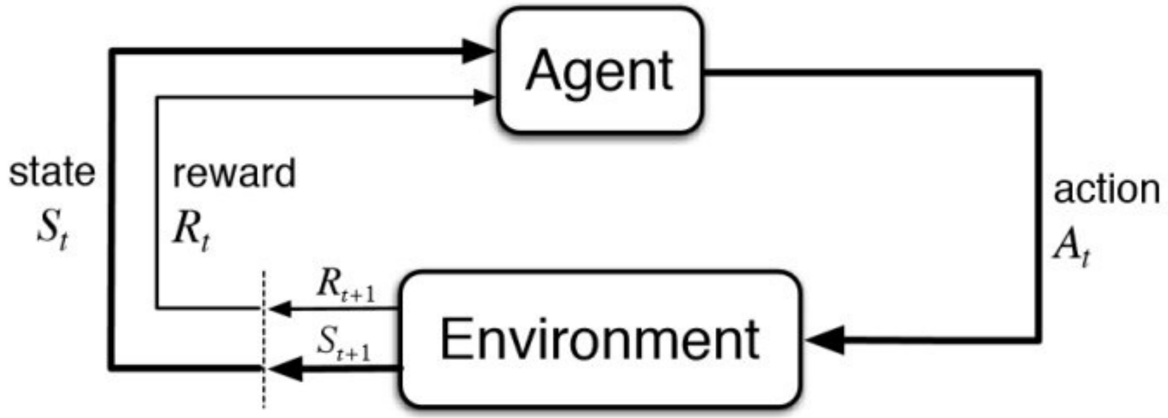
↓

Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

Landscape of Reinforcement Learning Algorithms



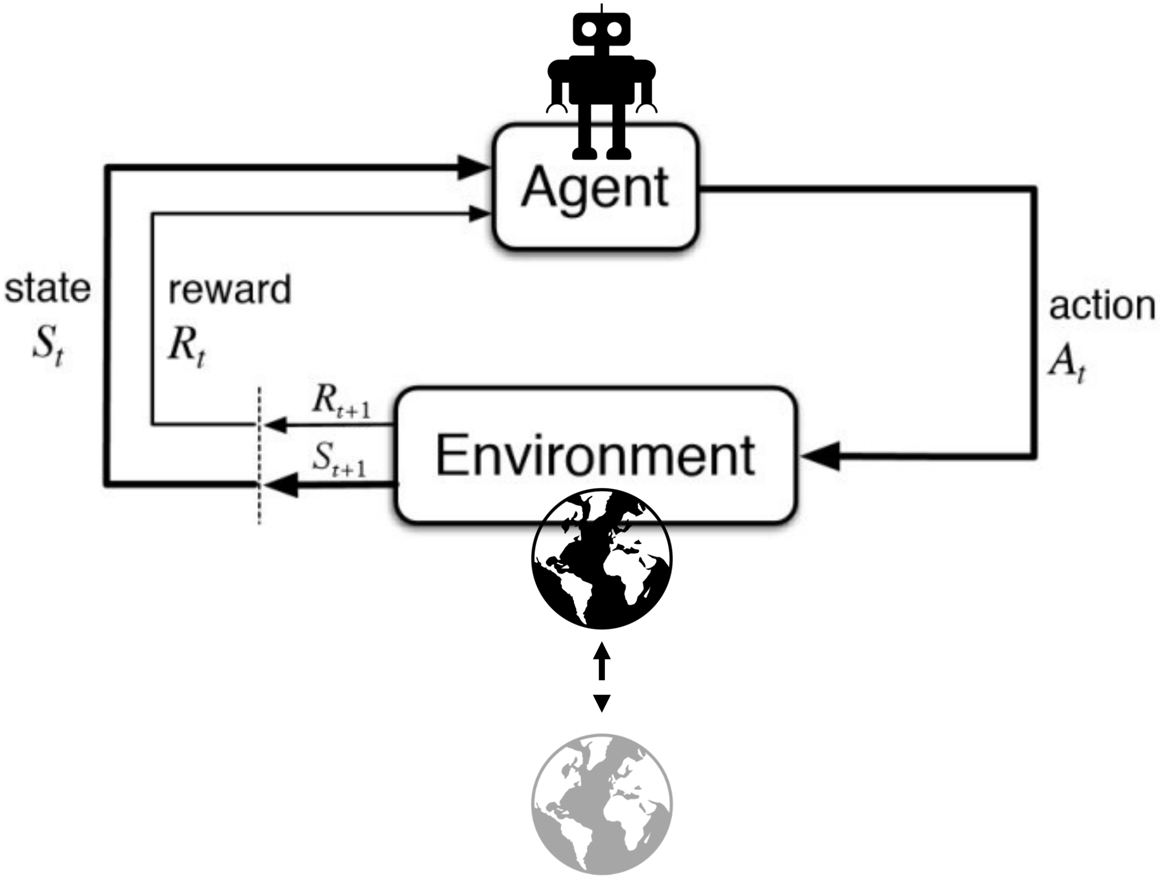
$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

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

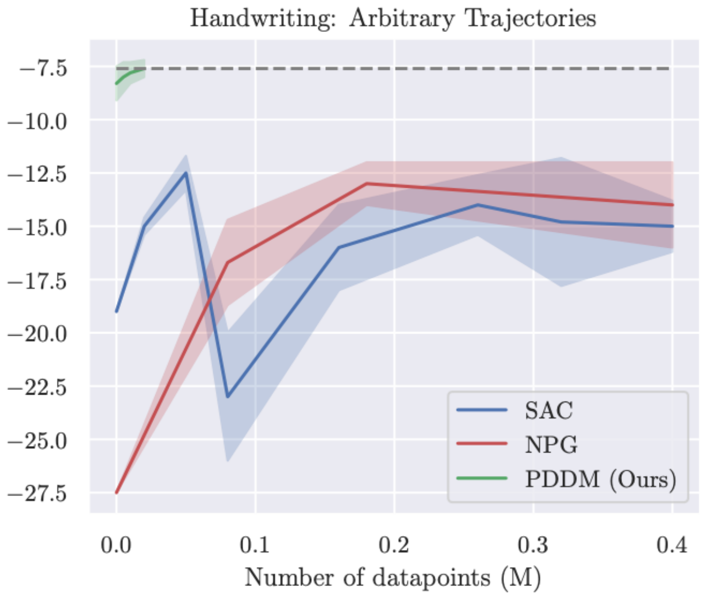
Intuitive: learn how the world works first and then plan in that model

Why do model-based RL?

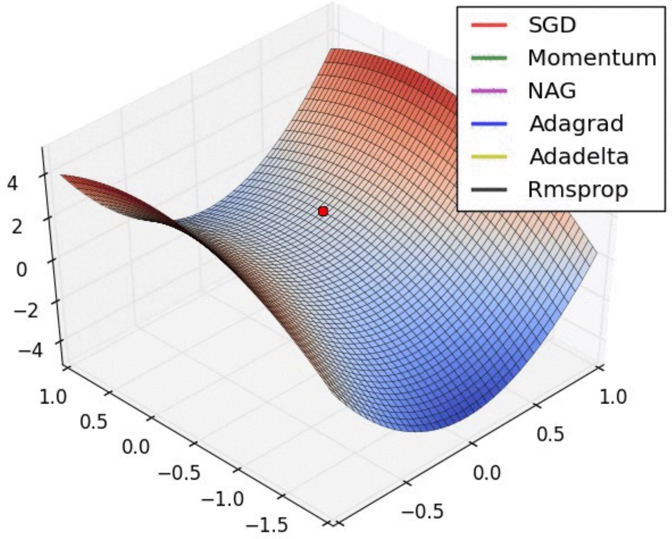
Why would we do this?

Transfer/Adaptive

Efficiency

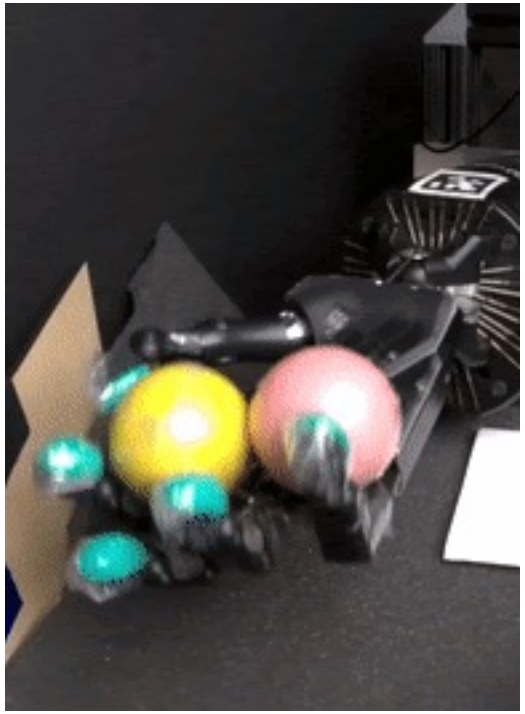


Simplicity



Naturally off-policy!

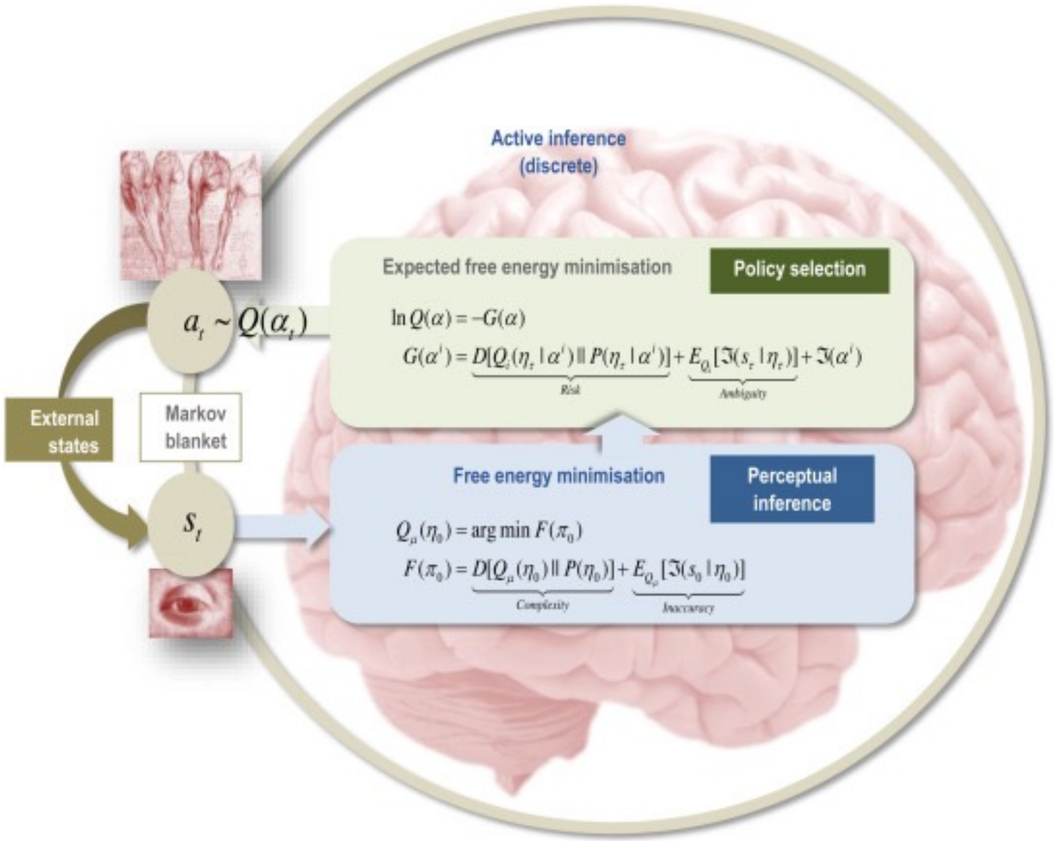
Why do model-based RL?

Just 2 hours of real robot training

Connections to Cognitive Science

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



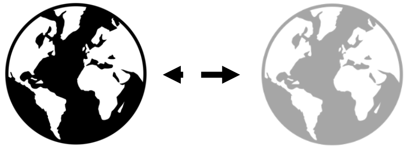
Reinforcement learning in the brain

Yael Niv

Psychology Department & Princeton Neuroscience Institute, Princeton University

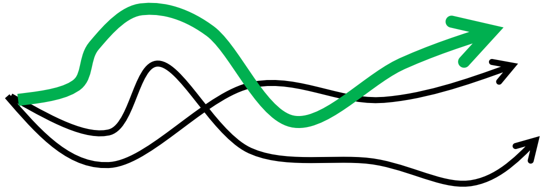
Model Based RL – Problem Statement

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?

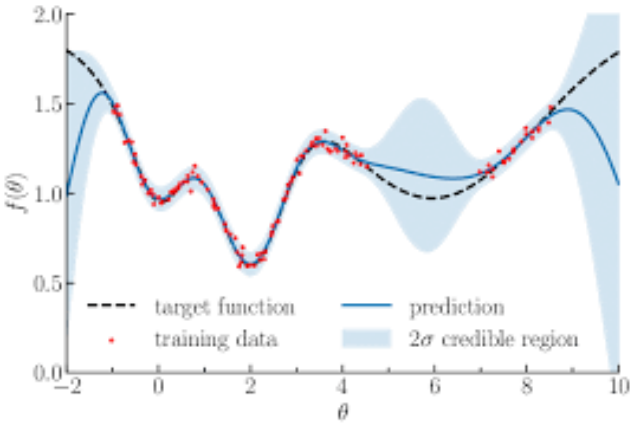
<u>iLQR/iLQG</u>

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

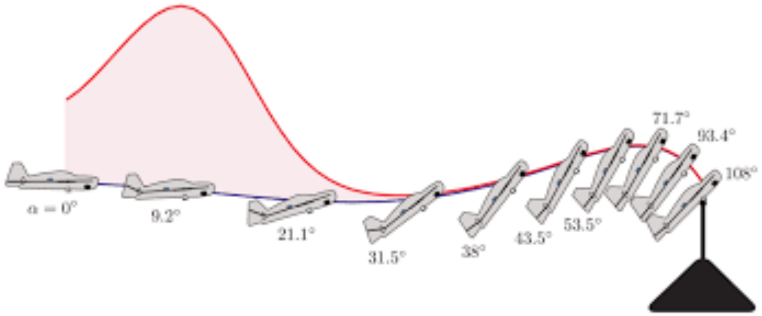
$$f(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{F}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \mathbf{f}_t$$

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

<u>MBRL with GPs/Non-Parametrics</u>



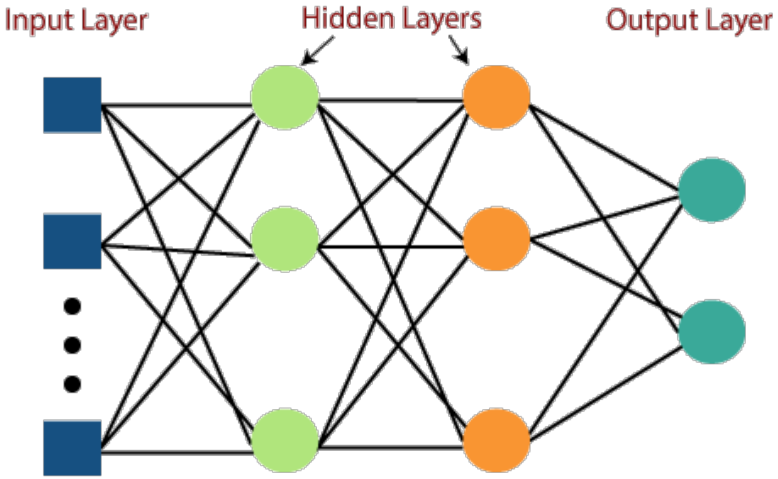
<u>Non-linear TrajOpt</u>



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

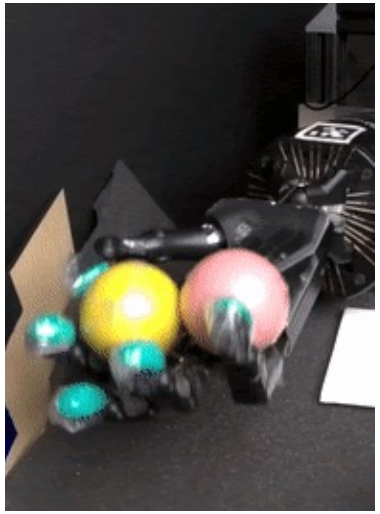
What will we cover today?

Use neural networks as our model!

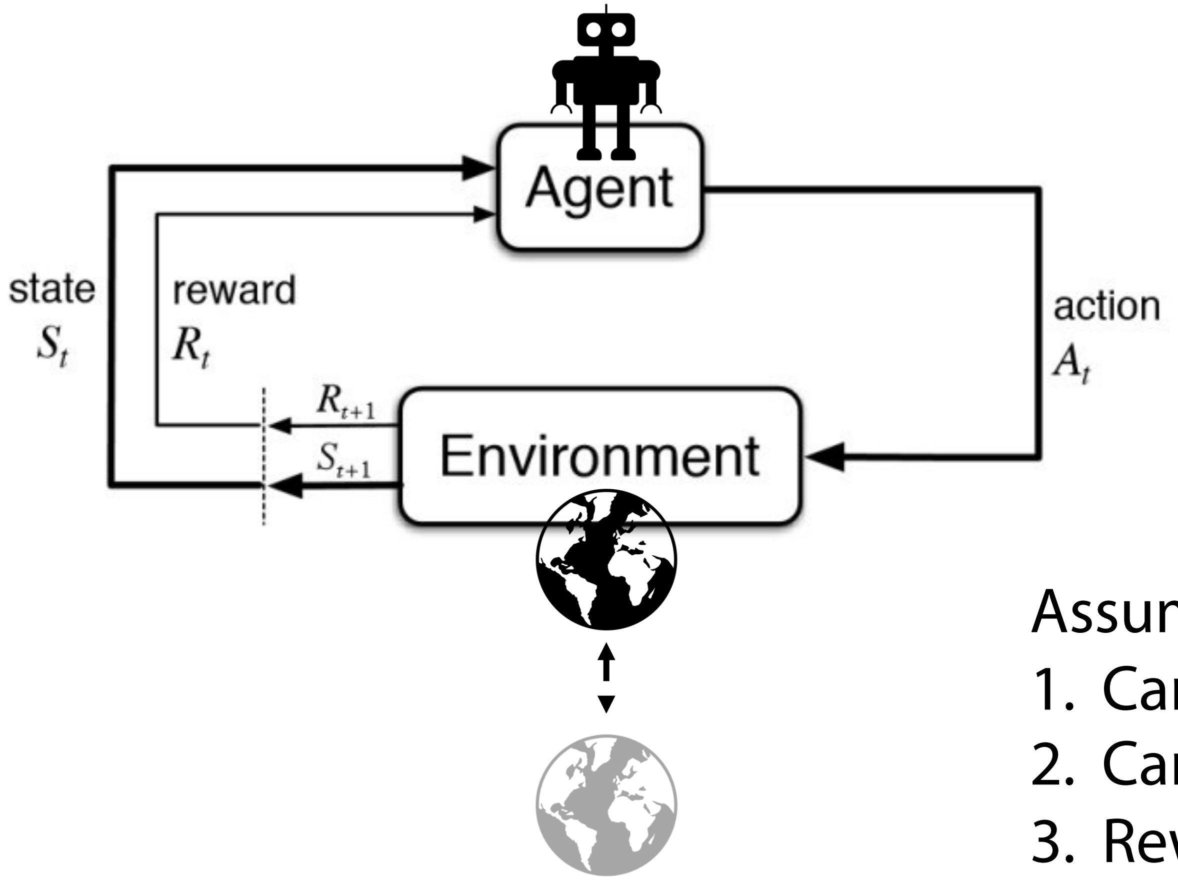


$$\hat{p}_\theta \leftarrow \arg\min_{\hat{p}_\theta} \mathcal{L}(\mathcal{D}, \hat{p}_\theta)$$

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



Model Based RL – Assumptions



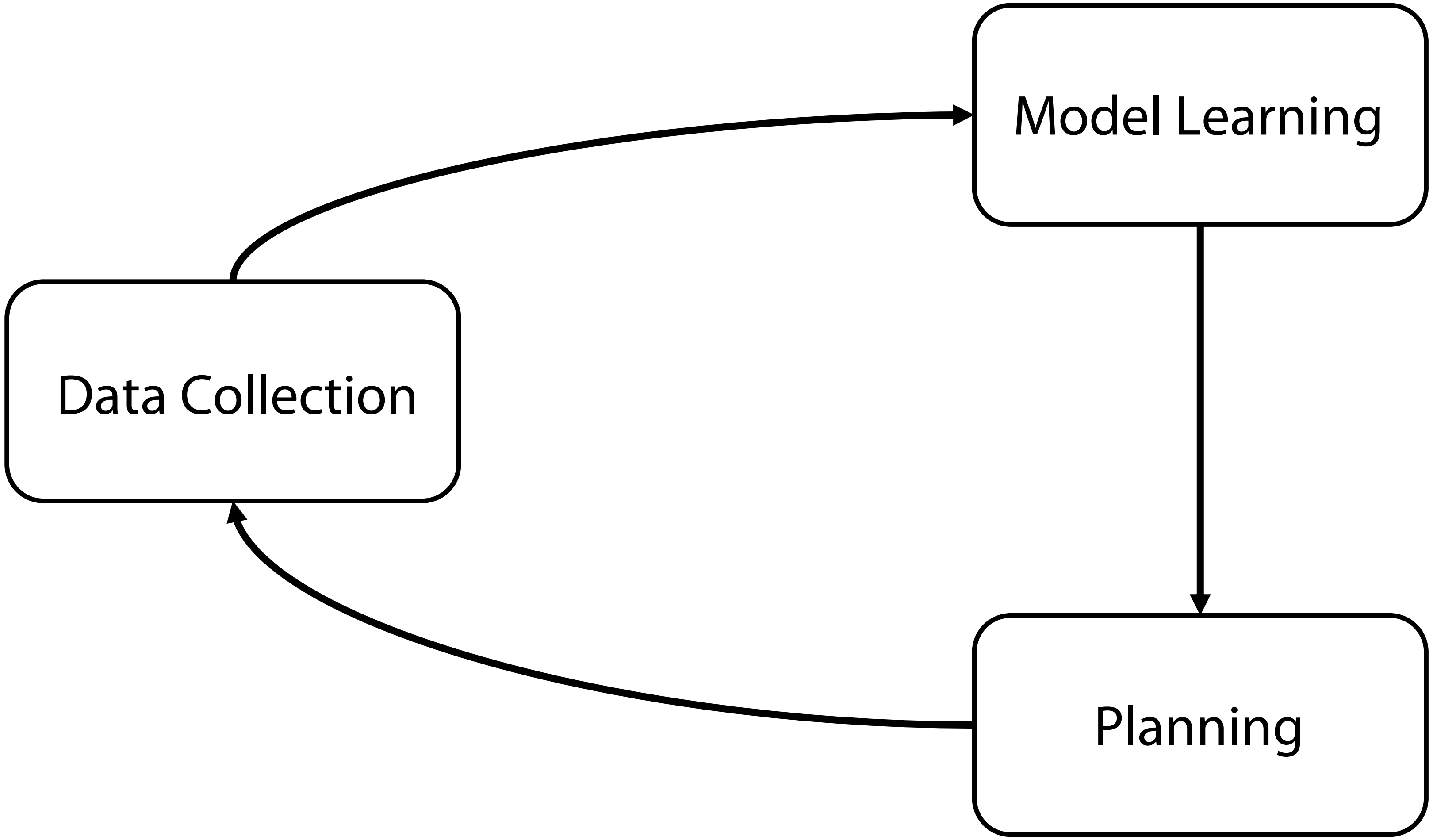
$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Assumptions:

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

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

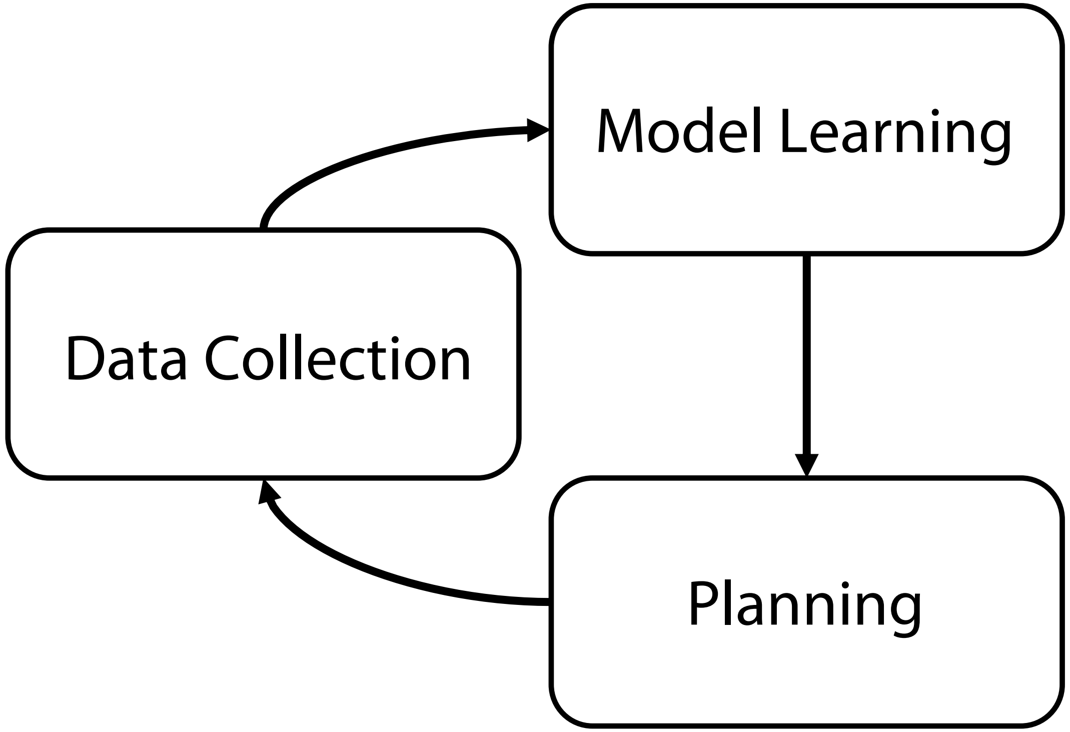
↓

Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

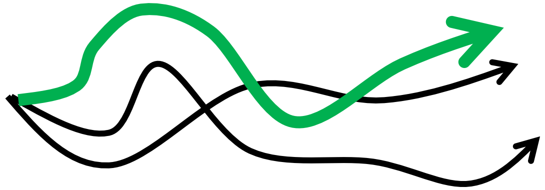
Model Based RL – Naïve Algorithm (v0)



Maximum likelihood supervised Learning

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

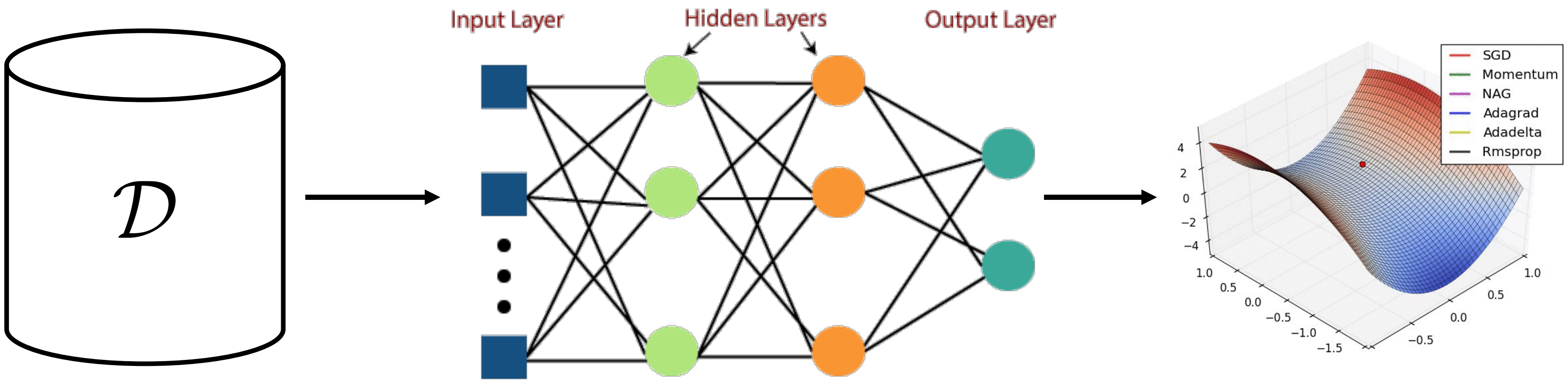
Random Search



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



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

More expressive may be better, at the risk of overfitting

Trick: Model Residual's (s' –s)

Model Based RL – Naïve Algorithm (Planning)

Planning

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

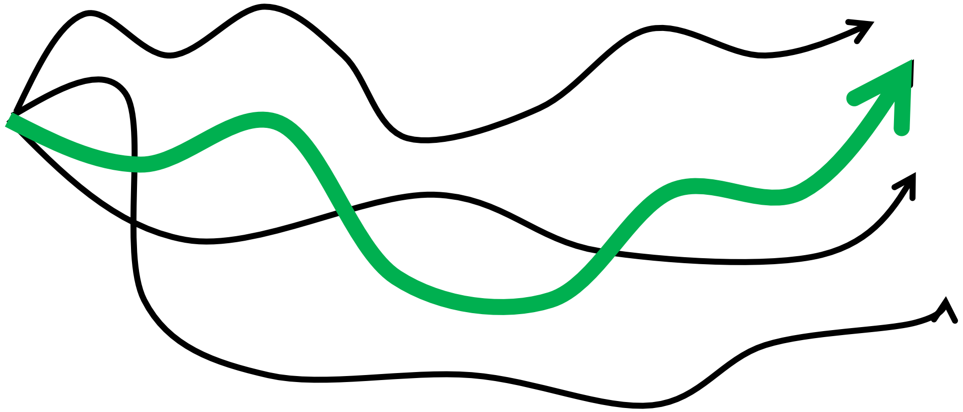
$$\hat{s}_{t+1} \sim \hat{p}_\theta(s_{t+1} | \hat{s}_t, a_t)$$

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

$\Downarrow$ Just do random search!

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

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

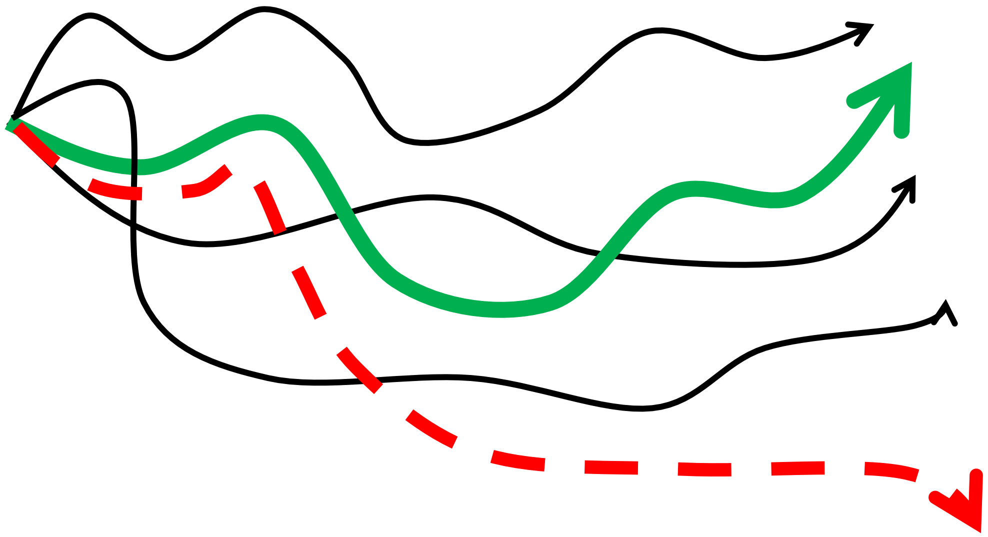


Just execute actions open loop!

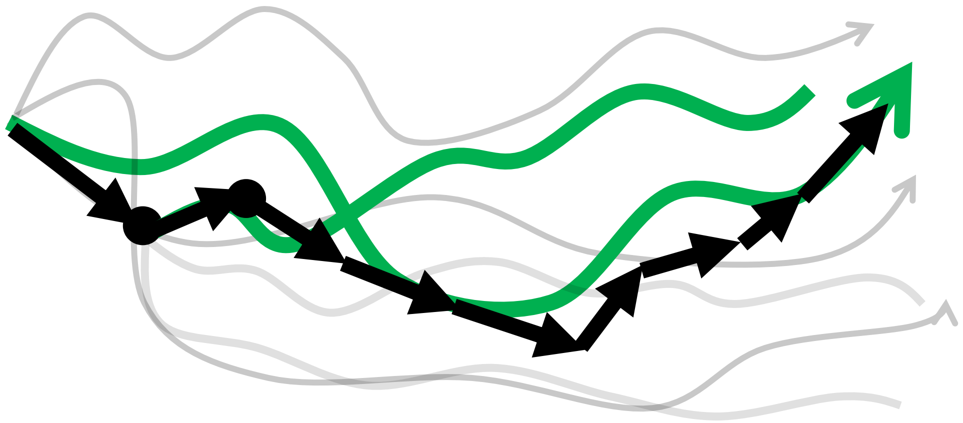
Can soften by taking softmax rather than argmax

Model Based RL – Naïve Algorithm (MPC)

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



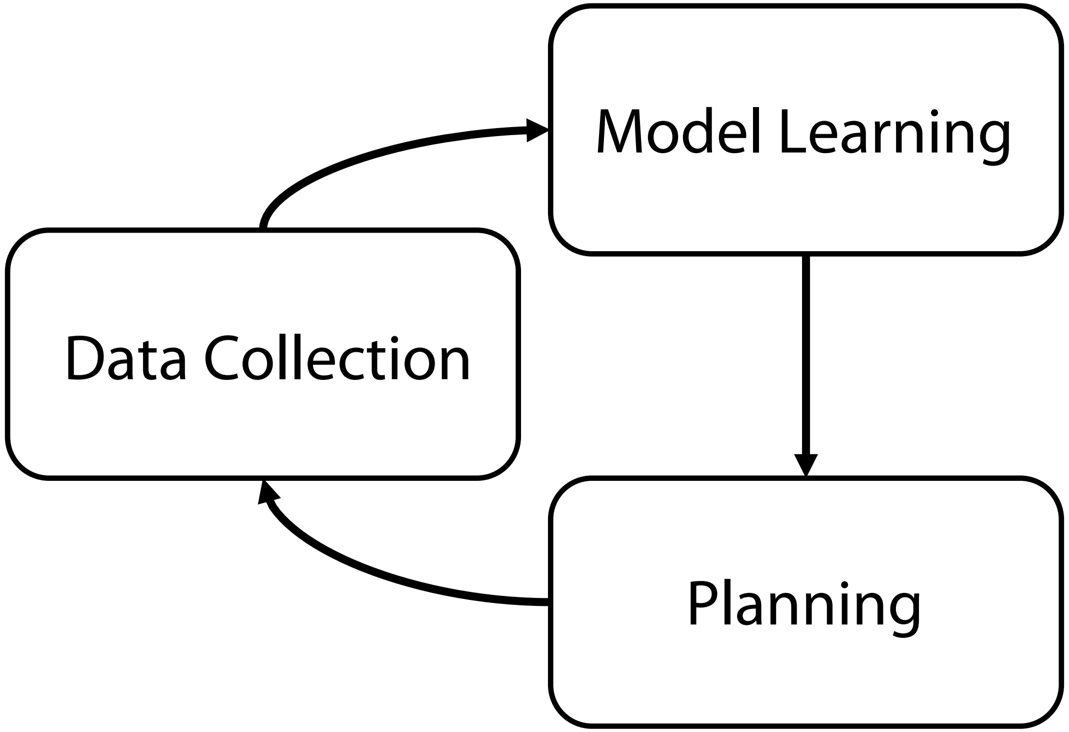
Replanning can help with divergence



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)

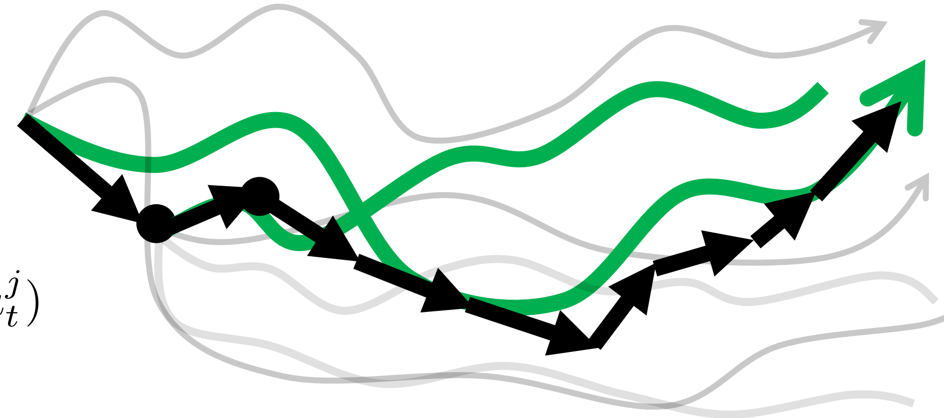


Data collected from planner

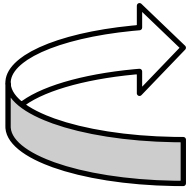
Maximum likelihood supervised Learning

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

Planning with Shooting + MPC



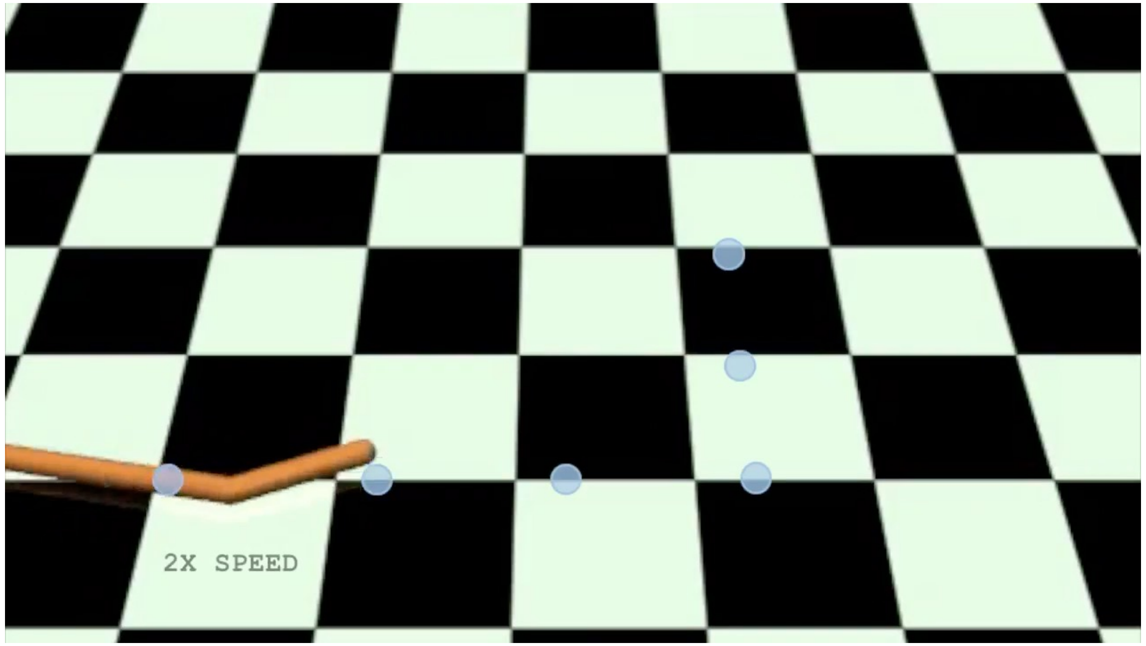
Better than open loop planning because of feedback



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

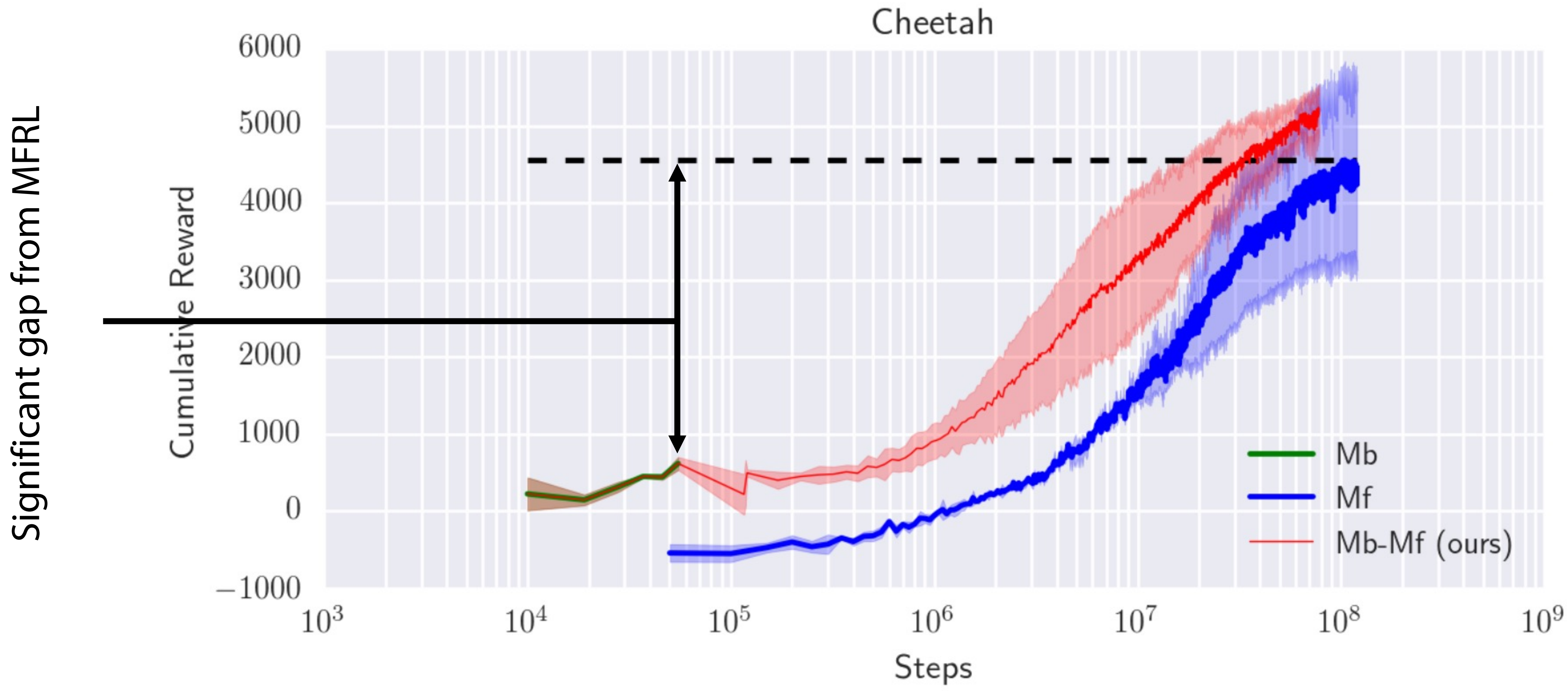
$$\hat{s}_{t+1}^j \sim \hat{p}_{\theta}(.|\hat{s}_t^j, a_t^j)$$

Does it work?



Just 20 minutes of training time with random data!

Does it work?



Cheetah
Significant gap from MFRL
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

↓

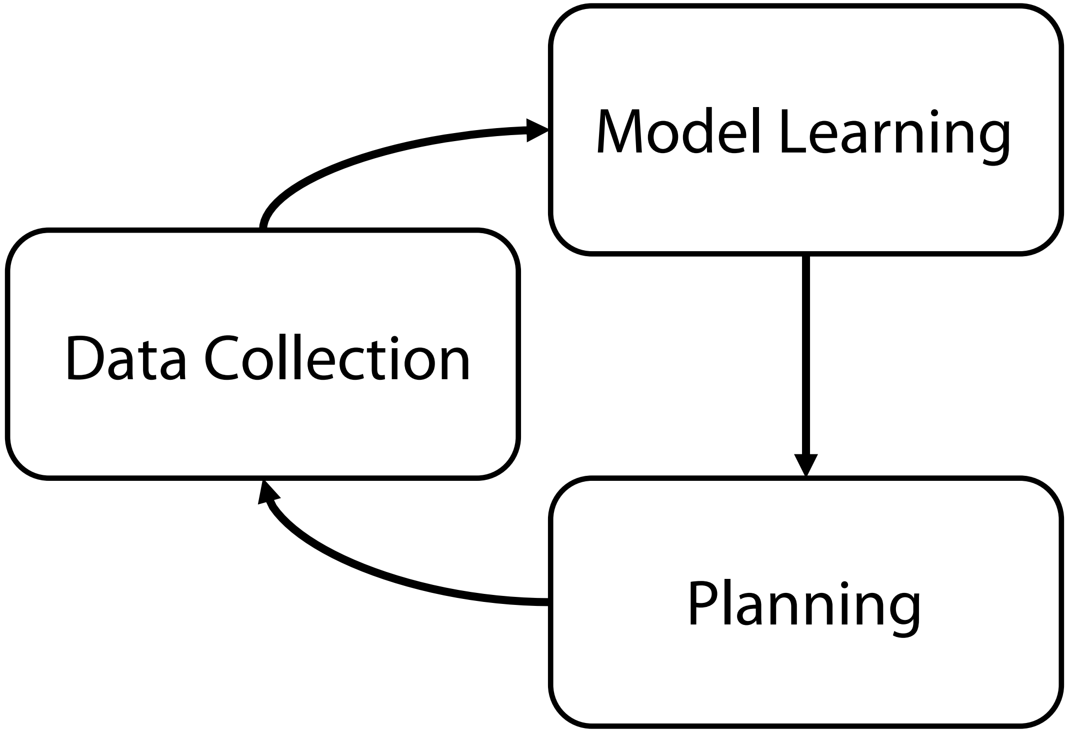
Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

What might be the issue?

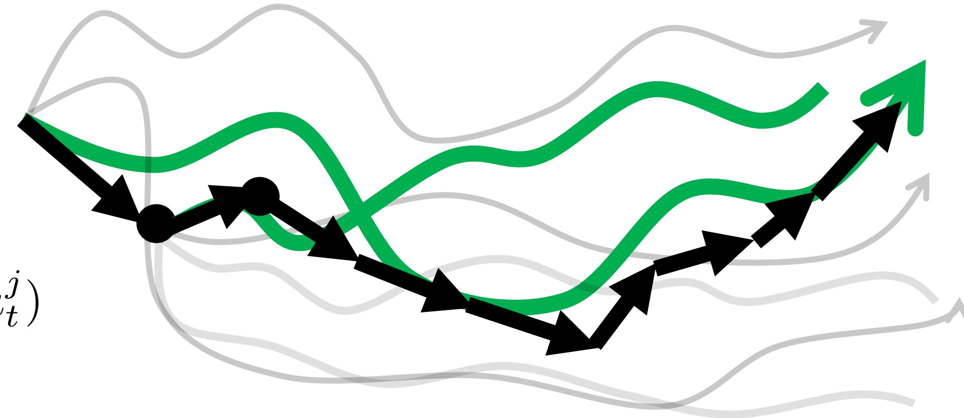
Data collected from planner



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



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

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

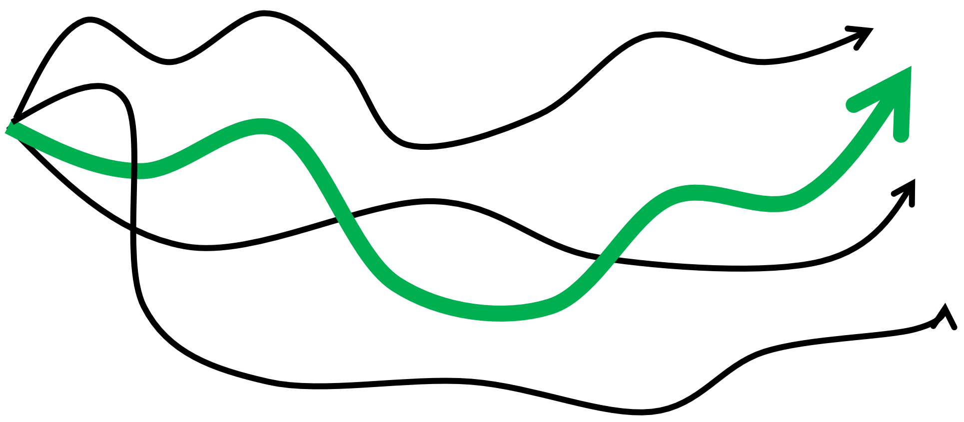
$$\hat{s}_{t+1}^j \sim \hat{p}_\theta(.|\hat{s}_t^j, a_t^j)$$

Better Sampling Techniques for Shooting

Sampled from stationary uniform/gaussian distribution

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

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

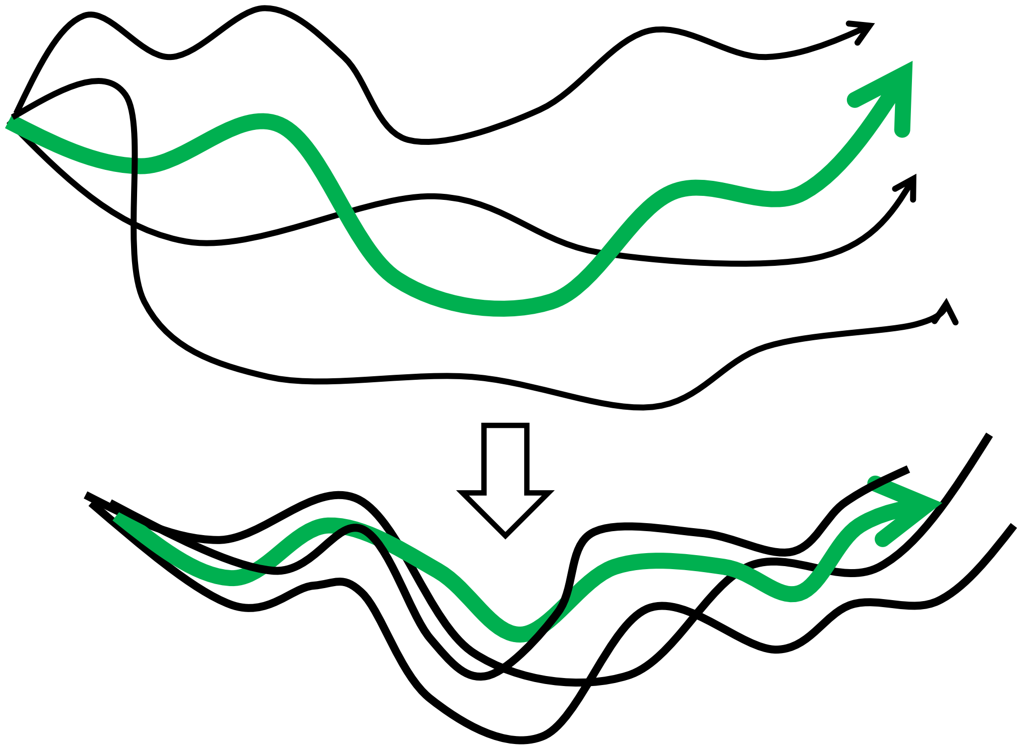


Can we inform the sampling function with the reward function?

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

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

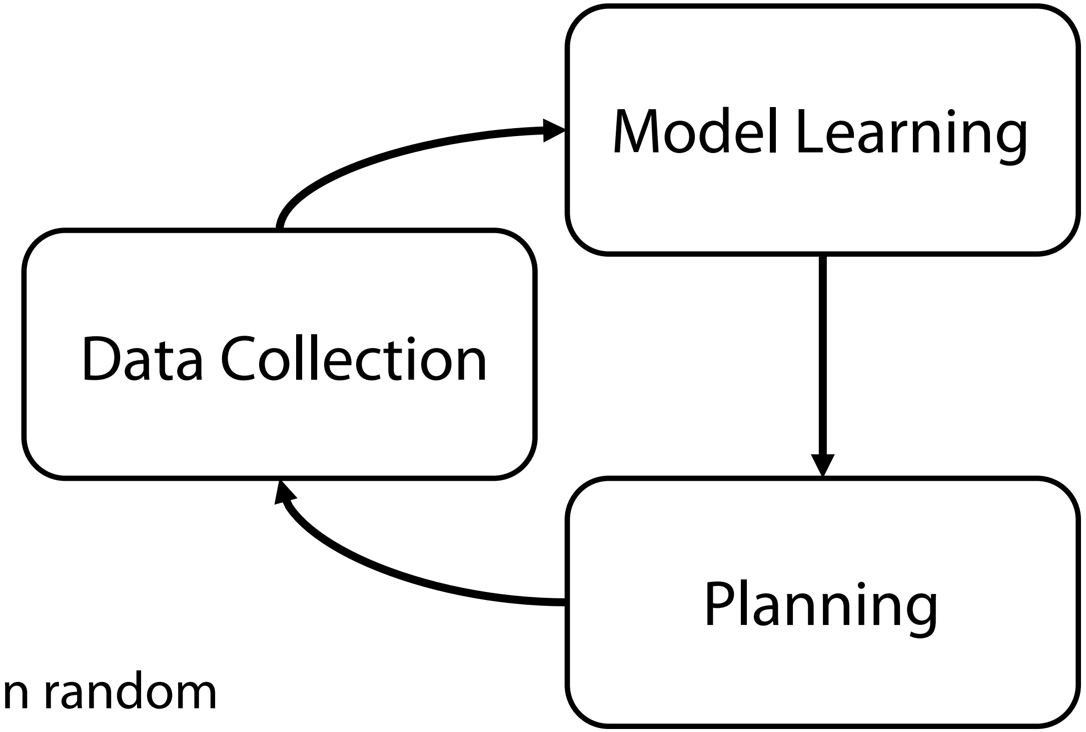
Sample trajectories using these action sequences with the model $\hat{p}_\theta$

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

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)

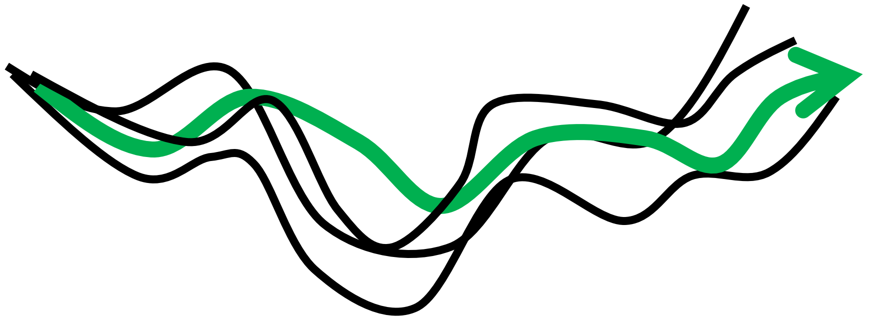


Data collected from planner

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!

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

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

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

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

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

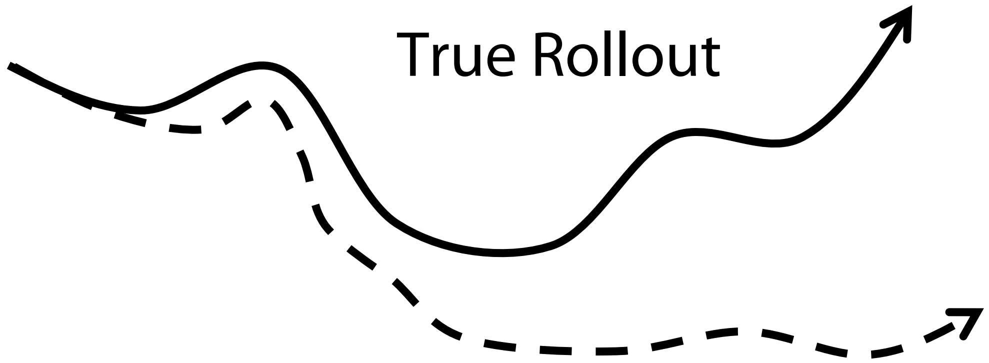
↓

Model based RL v4 → latent space models with images

What might be the issue?

Rollouts under learned model != Rollouts under true model

→ Model bias/compounding error



Predicted Rollout Under Model

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

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

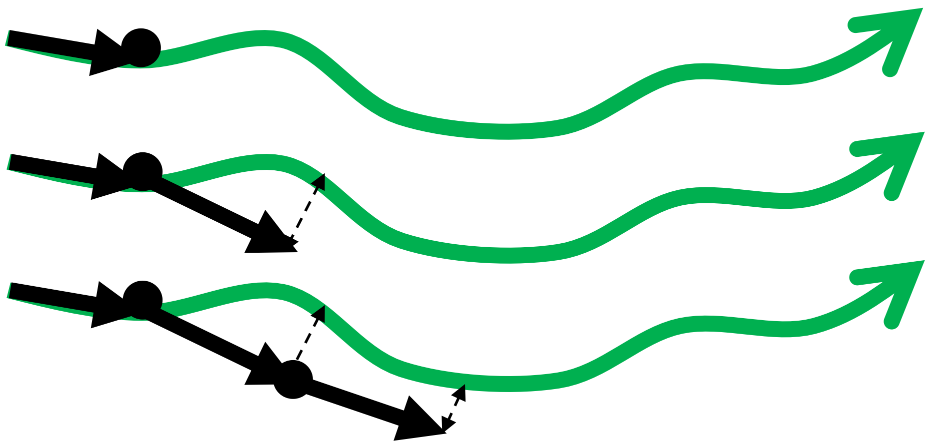
Simulation error – K step prediction error

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

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

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

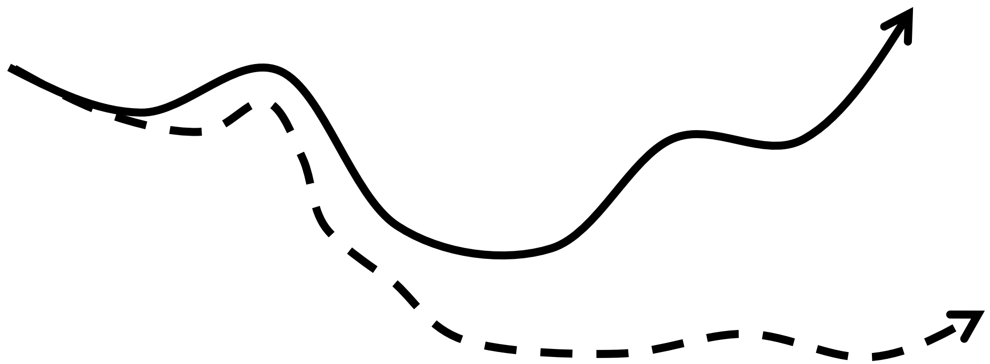
Can be a challenging non-convex optimization!

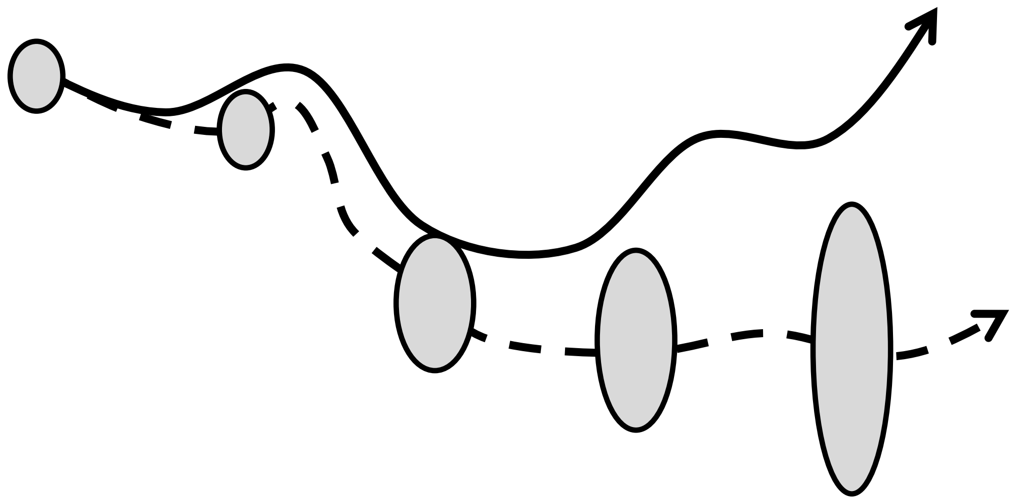


How might we deal with compounding error?

Idea 2: Estimate when OOD and account for it

→ Measure uncertainty!





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

What is uncertainty?

<u>Alleatoric Uncertainty</u>

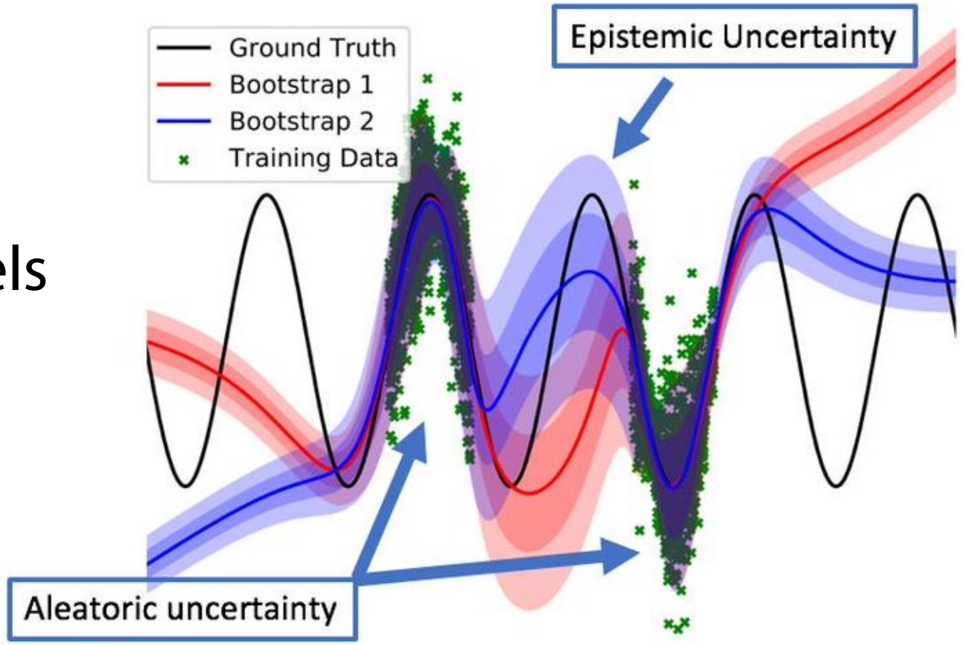
(environment stochasticity)

Easier, can use stochastic models

<u>Epistemic Uncertainty</u>

(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!

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

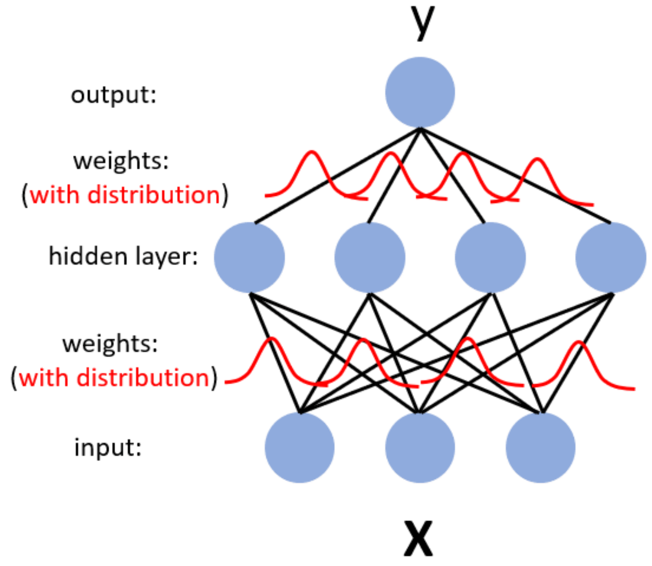
1. Bayesian neural networks
2. Ensemble methods
3. …

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



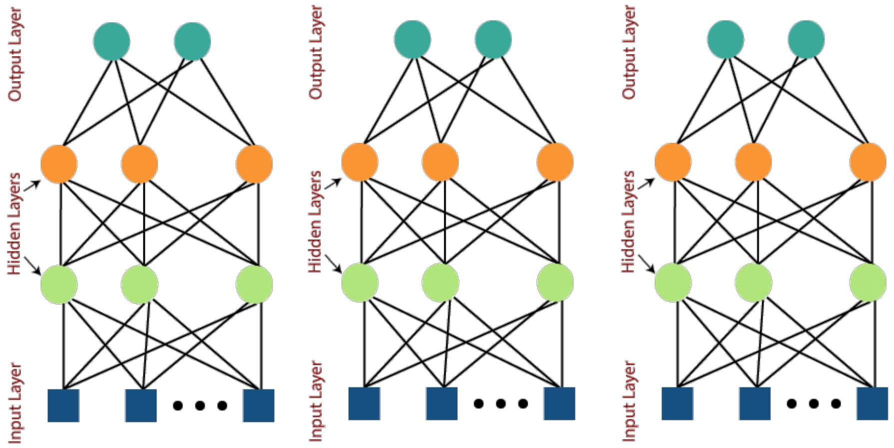
How might we measure uncertainty?

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

Difficult to estimate directly!

Learn an ensemble of models

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



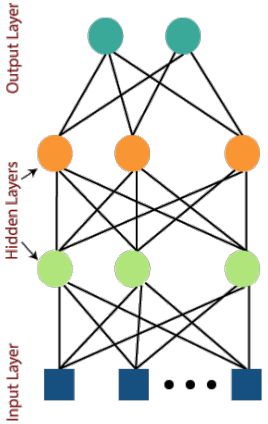
Low data regime → high ensemble variance

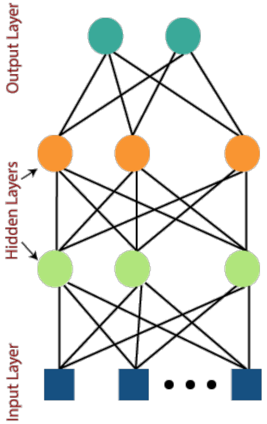
Approximate posterior

Easier and more expressive than BNNs!

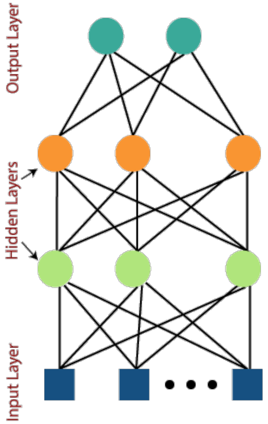
Model Based RL – Learning Ensembles of Dynamics Models

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





…

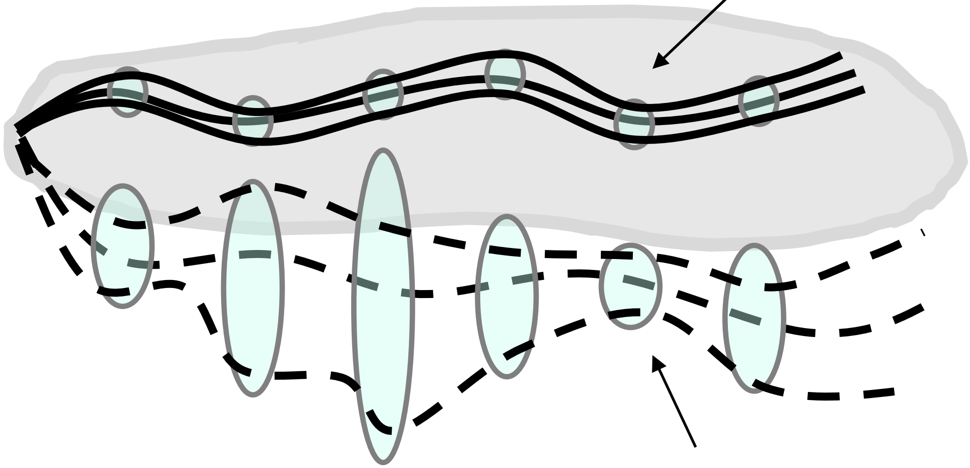


$$\max_\theta \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\log \hat{p}_\theta(s'|s,a)\right] \quad \max_\theta \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\log \hat{p}_\theta(s'|s,a)\right] \quad \max_\theta \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\log \hat{p}_\theta(s'|s,a)\right]$$

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

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

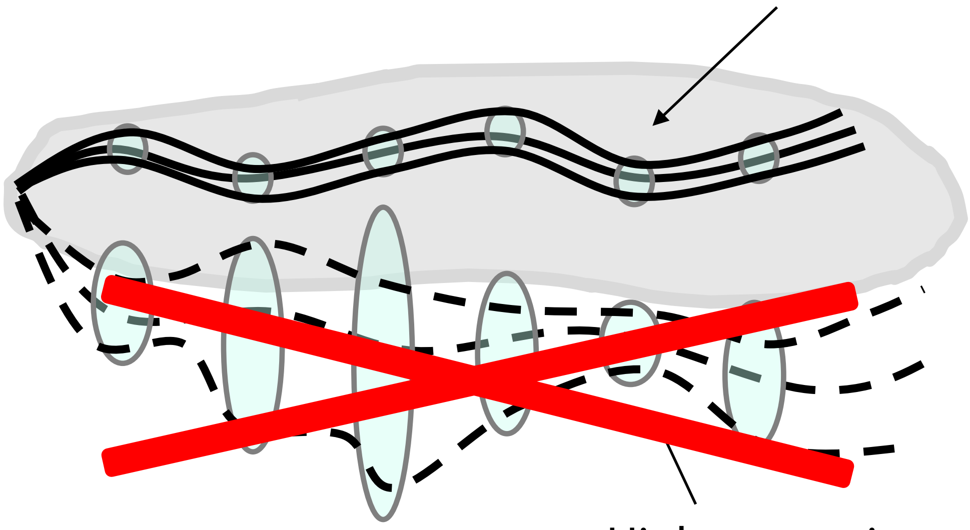
$$(\hat{s}_{t+1}^j)^i \sim \hat{p}_{\theta_i}(.|(\hat{s}_t^j)^i, a_t^j)$$

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



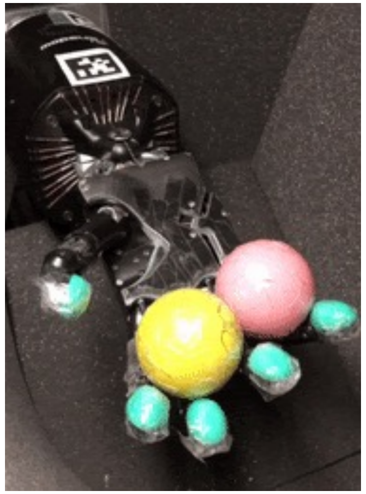
Penalize ensemble variance

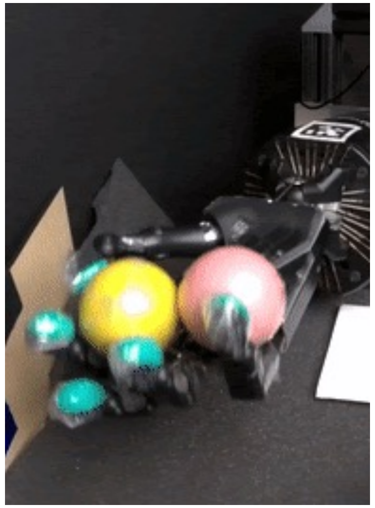
$$\arg \max_{(a_0^j, a_1^j, \ldots, a_T^j)_{j=1}^N} \sum_{i=1}^{K} \sum_{t=0}^{T} r((\hat{s}_t^j)^i, a_t^j) - \lambda \mathrm{Var}((\hat{s}_t^j)^i)$$

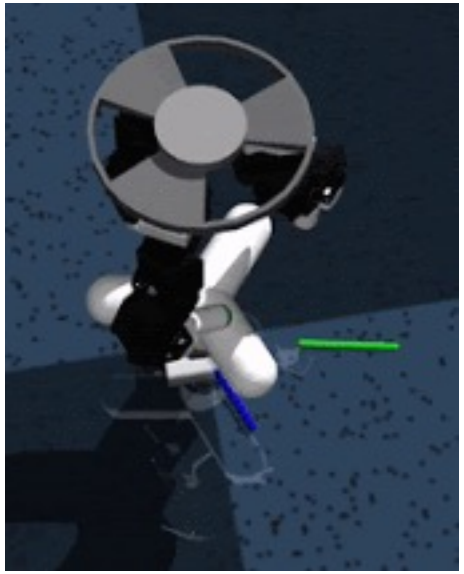
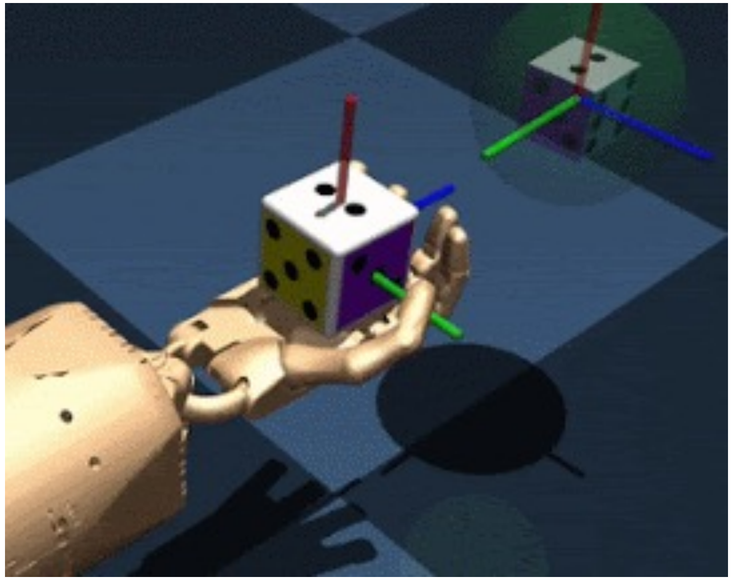
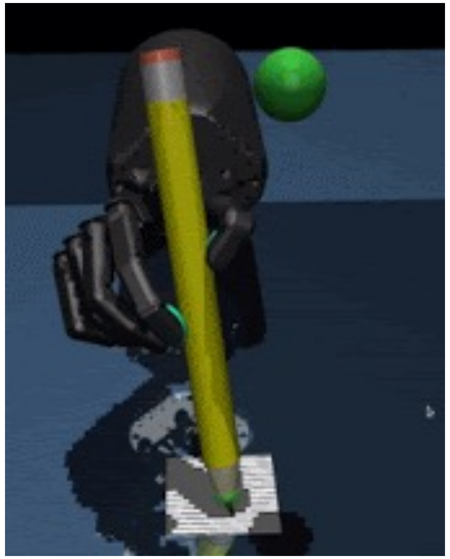
$$(\hat{s}_{t+1}^j)^i \sim \hat{p}_{\theta_i}(.|(\hat{s}_t^j)^i, a_t^j)$$

Avoids overly OOD settings since these states are explicitly penalized

Does this work?



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

↓

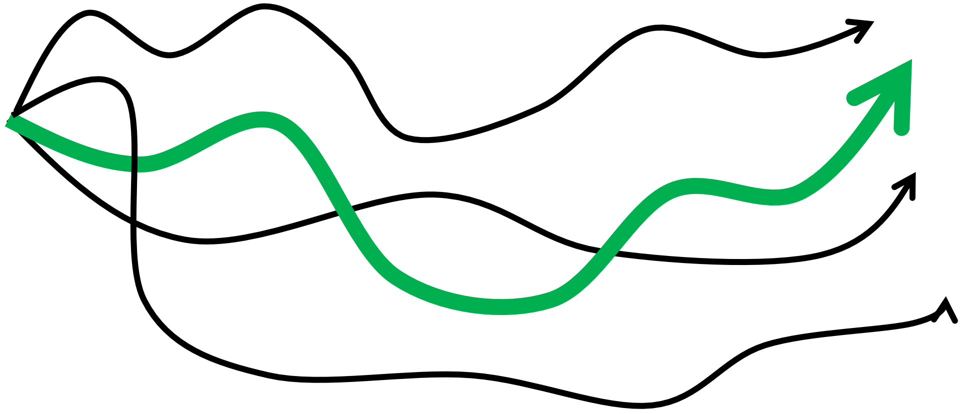
Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

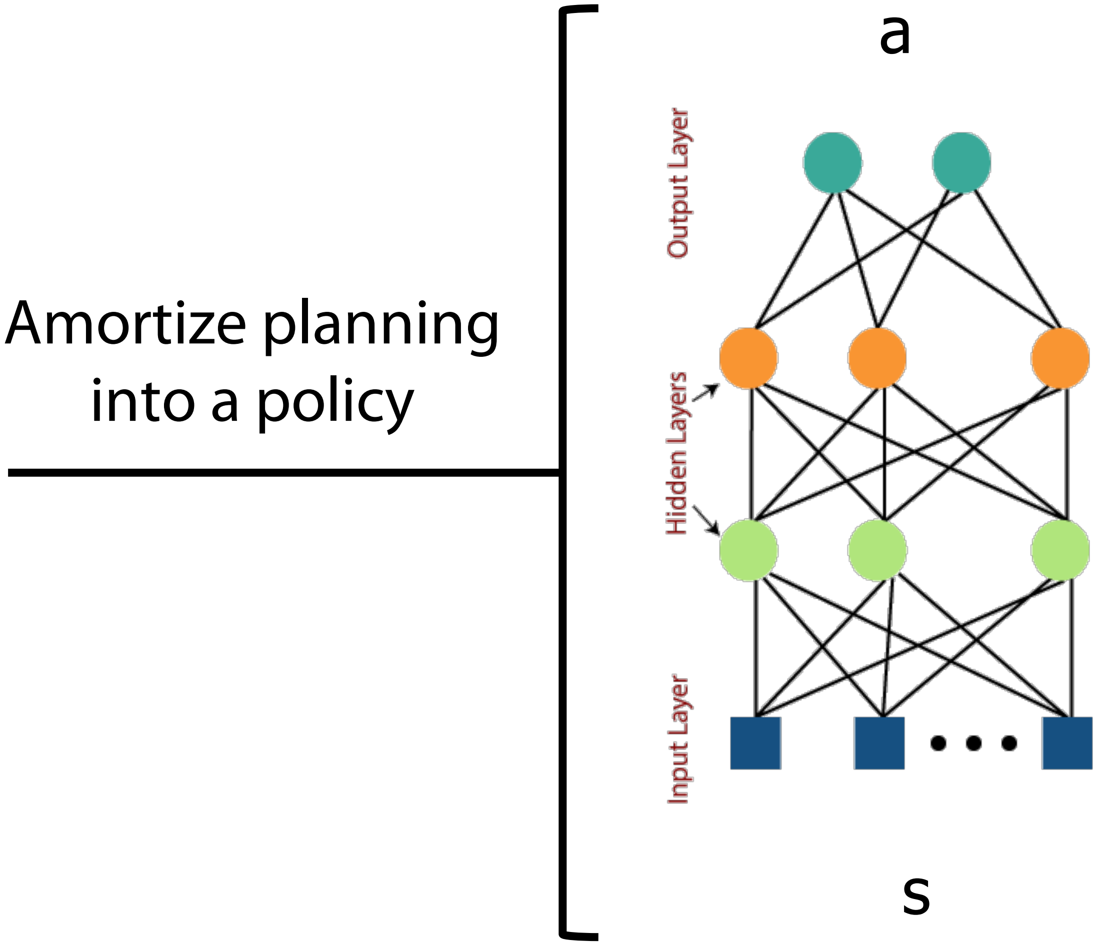
What might be the issue?

Huge number of samples
needed to reduce variance



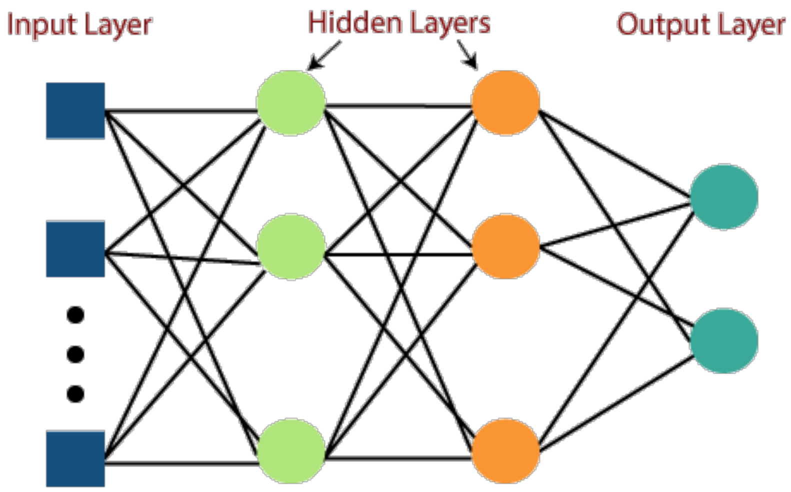
Extremely slow, hard to run in real time

Amortize planning
into a policy

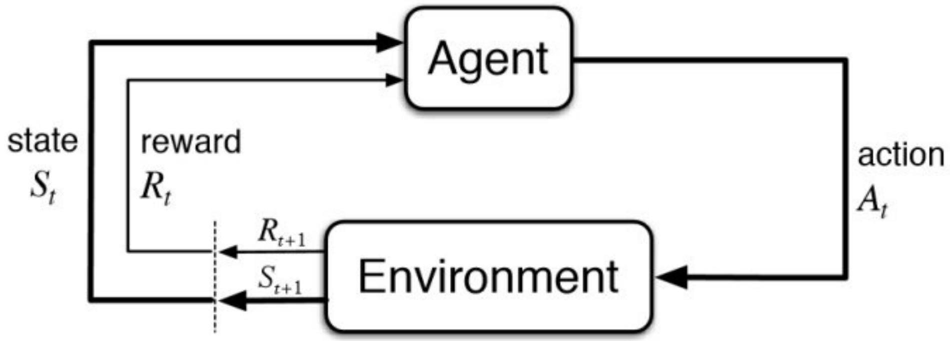


Speeding Up Model-Based Planning

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

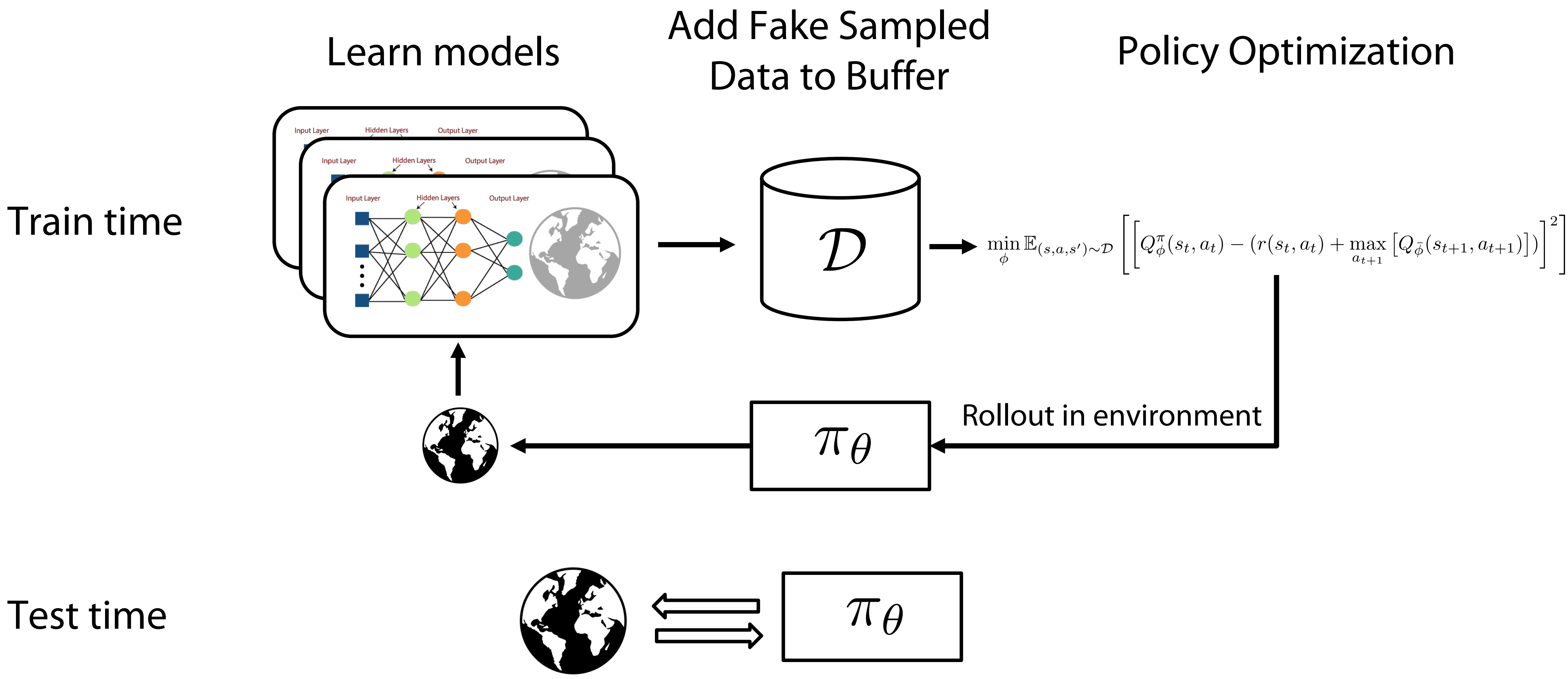


Use model(s) to generate data for policy optimization

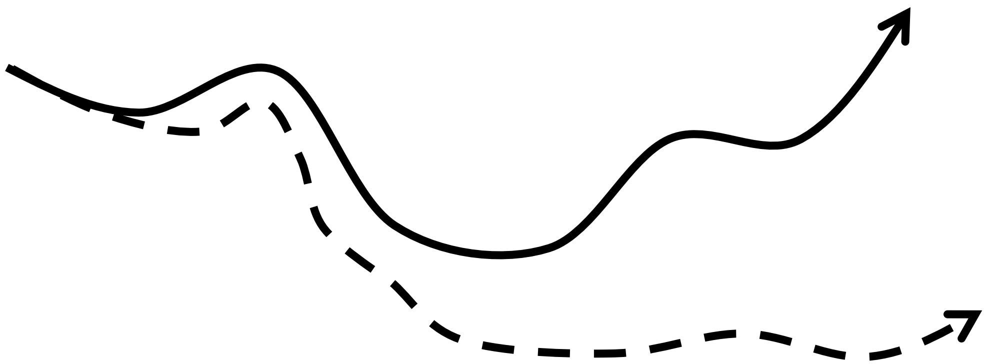


Can use PG or off-policy!

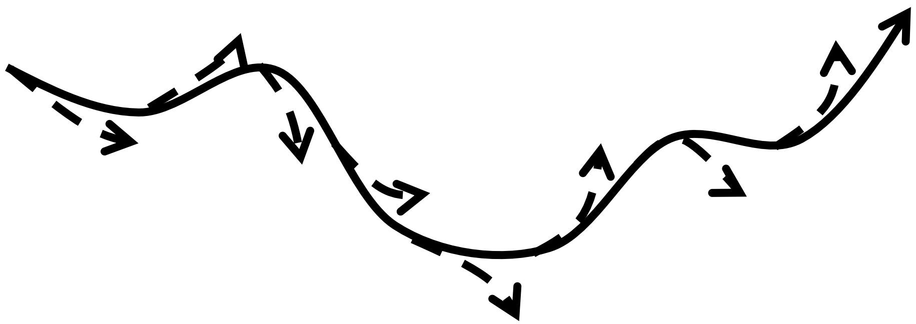
Generating Data for Policy Optimization



What matters in generating data from models?



Long horizon rollouts can deviate

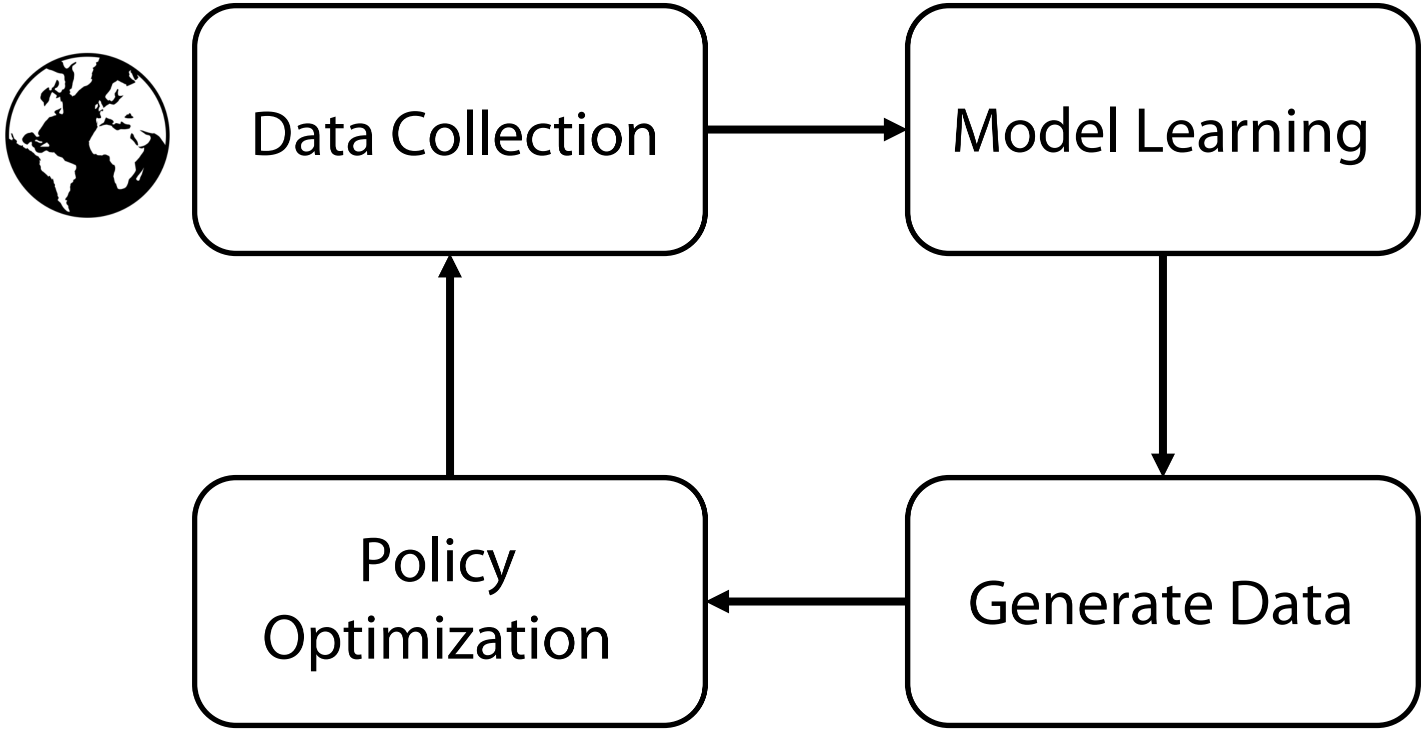


Short horizon rollouts deviate far less

Balance between off-policy coverage and compounding error

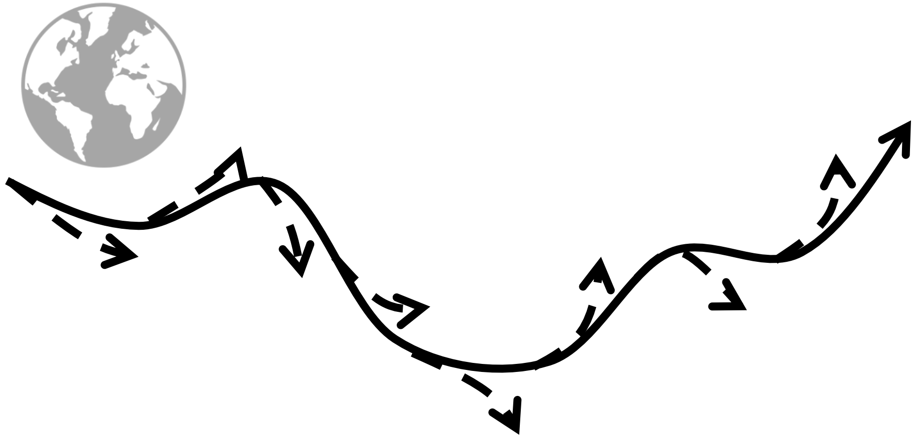
More in the readings!

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



Maximum likelihood supervised Learning

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



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

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

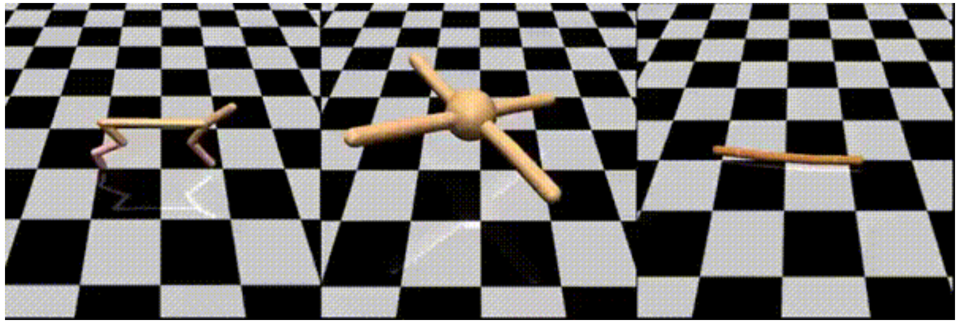
↓

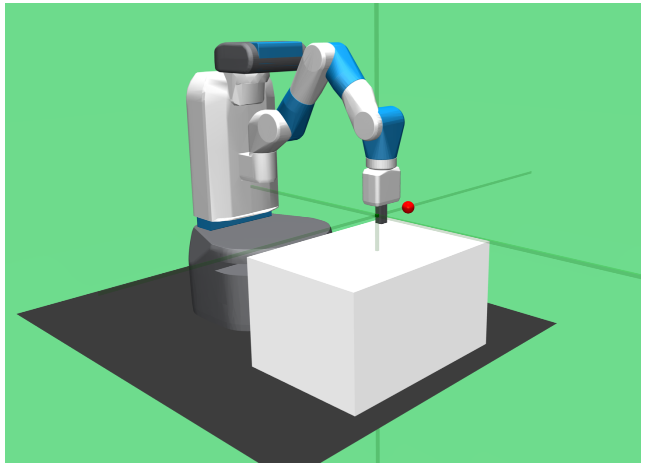
Model based RL v3 → policy optimization with models

↓

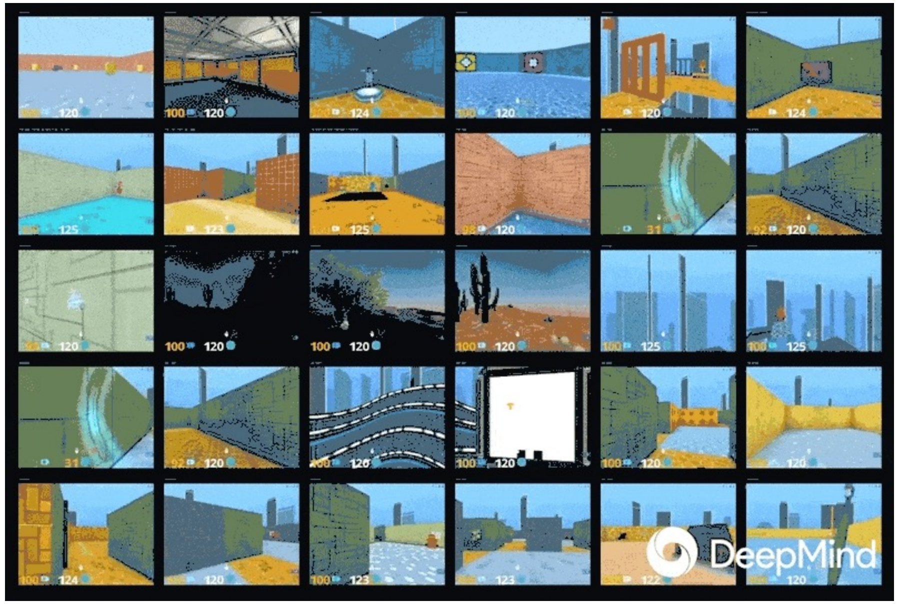
Model based RL v4 → latent space models with images

What about images?





State based domains



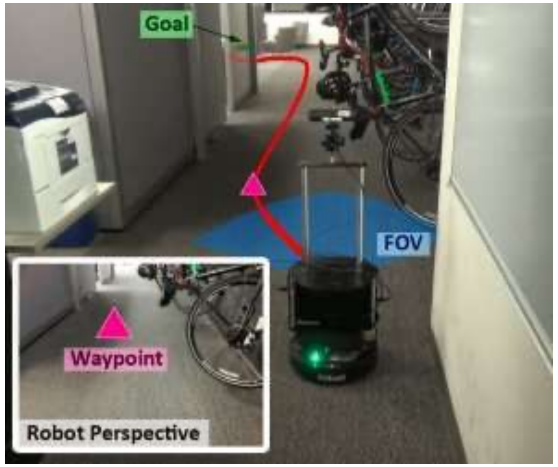
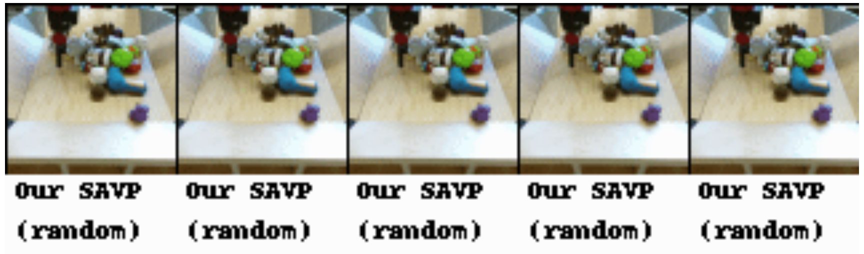


Image based domains

Why is learning from images hard?

Generative modeling is videos, challenging to model multimodal correlated predictions

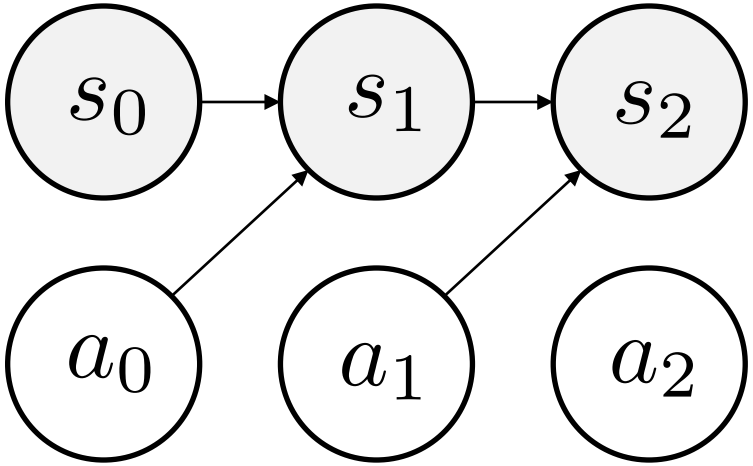


Partially observable!

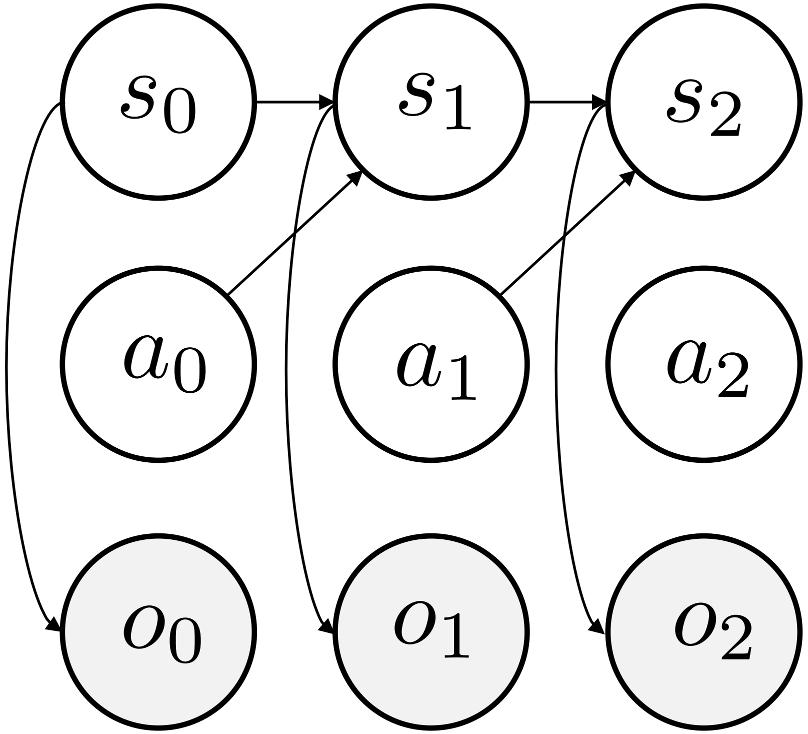
Long horizon predictions in video space can be challenging!

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

Fully observed – Markovian case



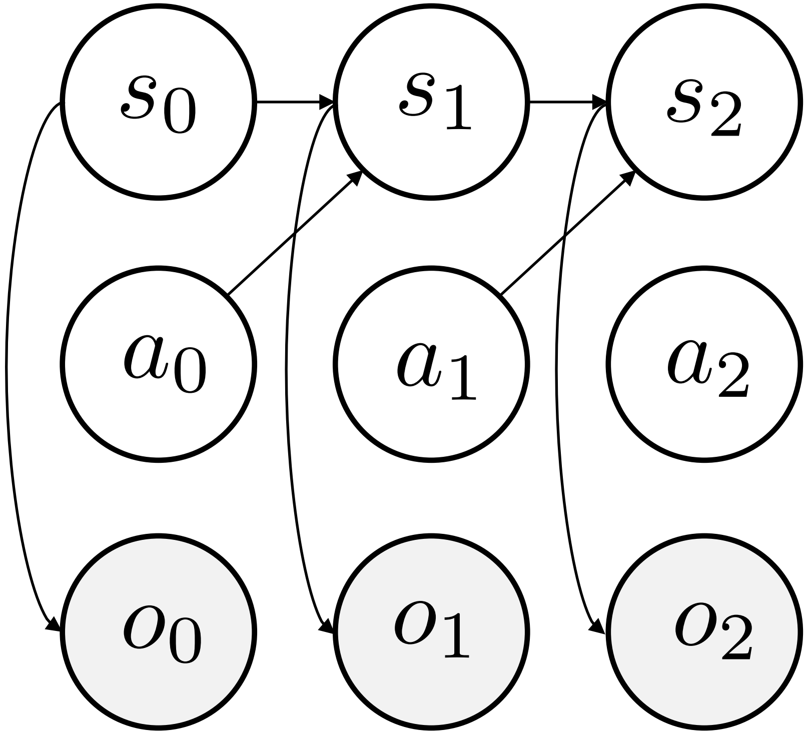
Partially observed – Non-Markovian case



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 $p_\psi(o_t|s_t)$

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

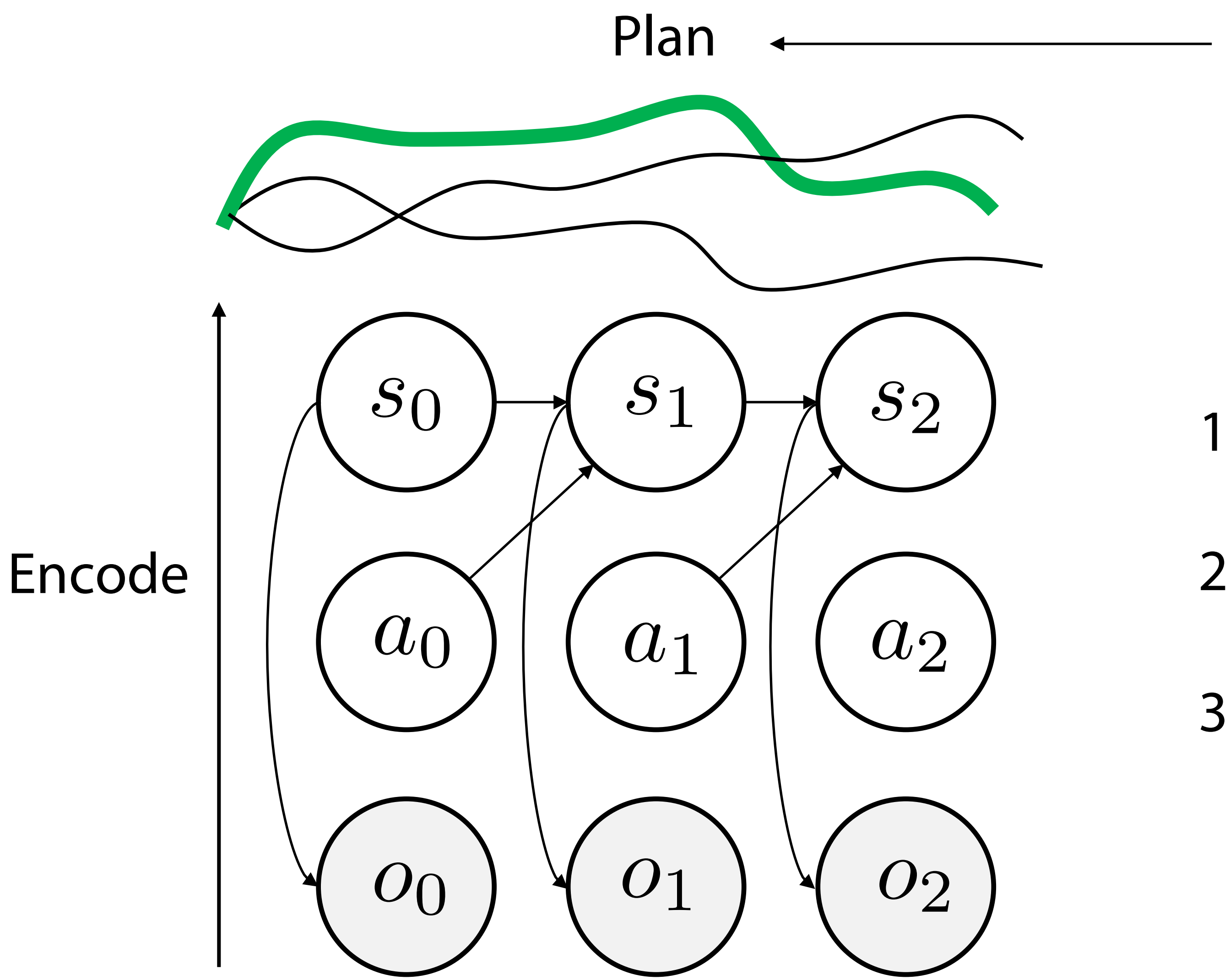
Learn reward predictor from latent state $p_\zeta(r_t|s_t)$

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

$$\boxed{\;+\;} \rightarrow \mathcal{L}$$

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

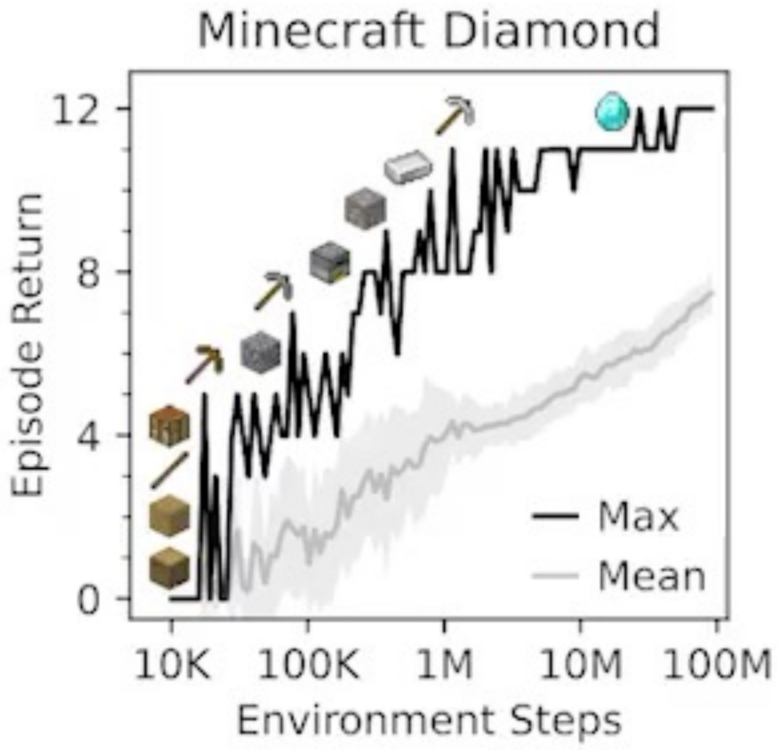
How do we **use** latent space models?



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

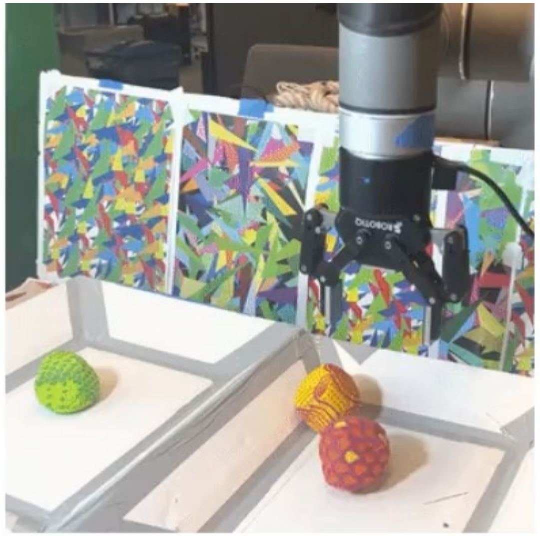
1. Avoids predicting image frames at planning time
2. Scales much better than image prediction
3. Allows for longer horizon predictions

Does this work?

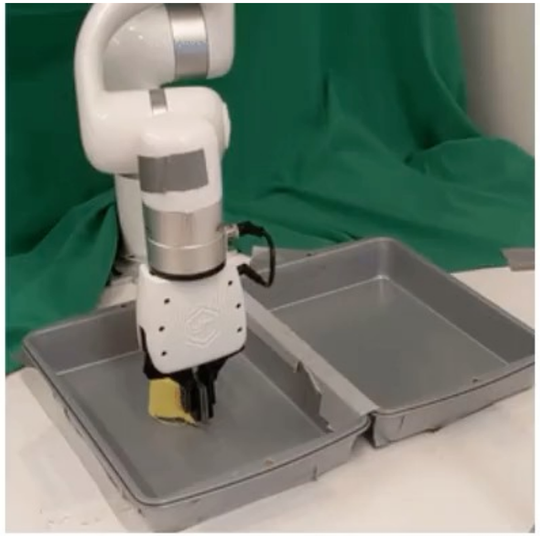


Does this work?

A1 Quadruped Walking



UR5 Multi-Object Visual Pick Place



XArm Visual Pick and Place

Sphero Ollie Visual Navigation

Training from images in < 1 hour!

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

↓

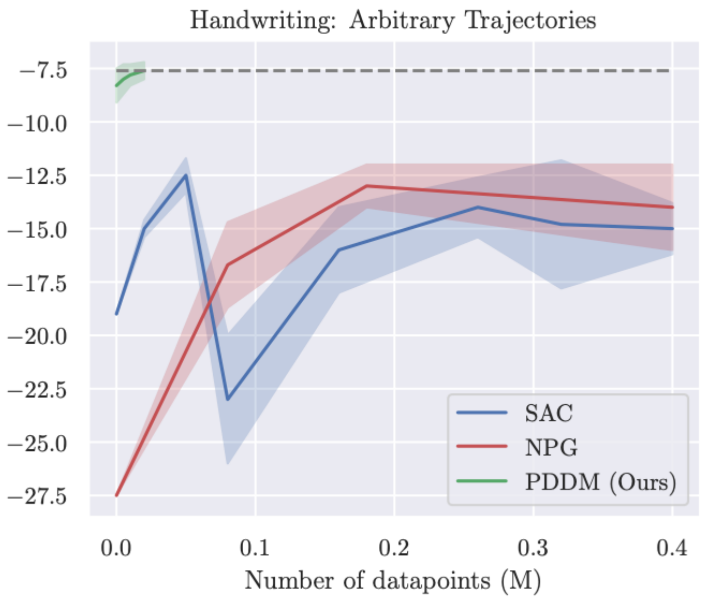
Model based RL v4 → latent space models with images

Why should you care?

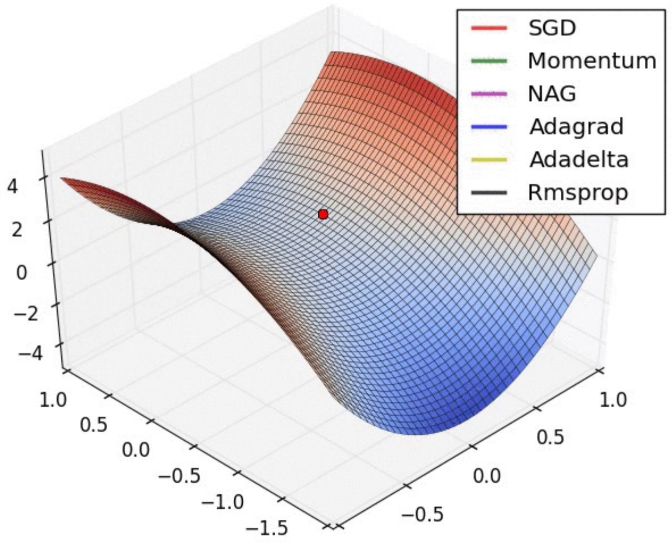
Model based RL **may be** a much more practical path to real world robotics

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