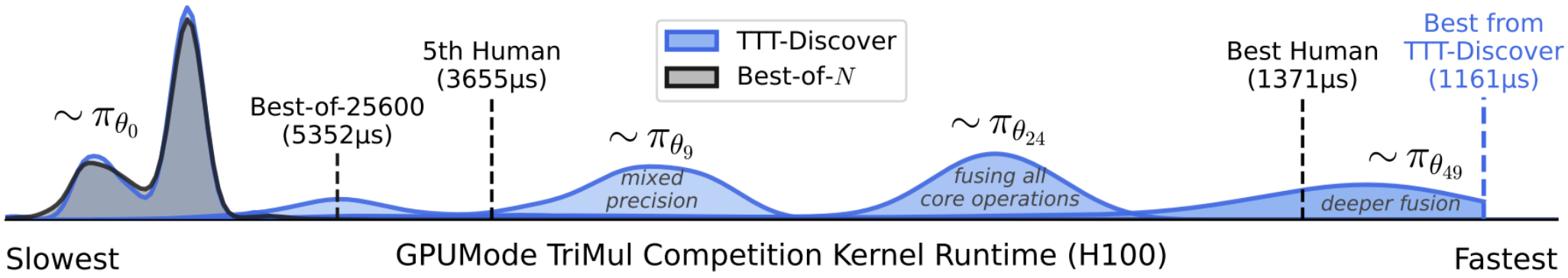
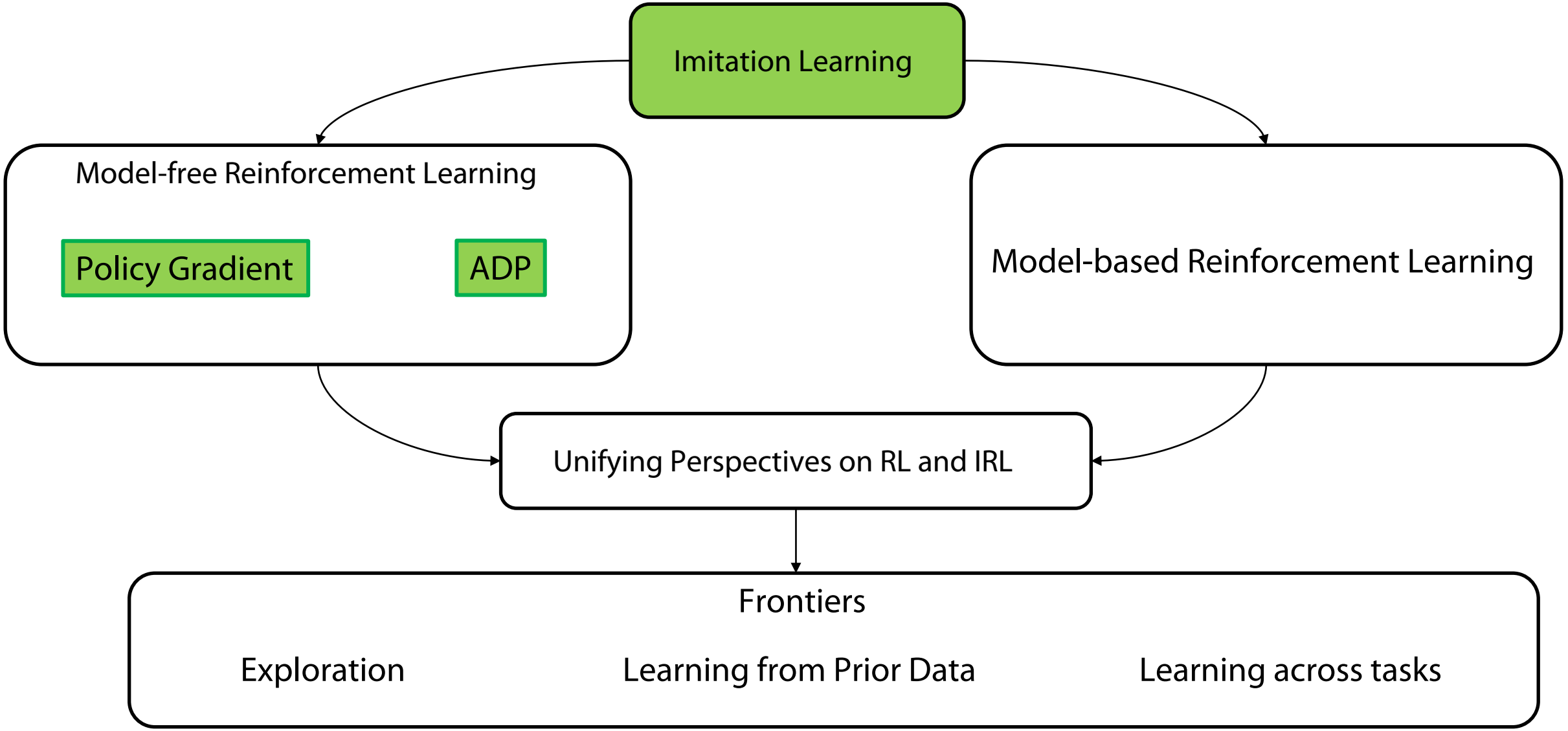
Reinforcement Learning
Spring 2026

Abhishek Gupta

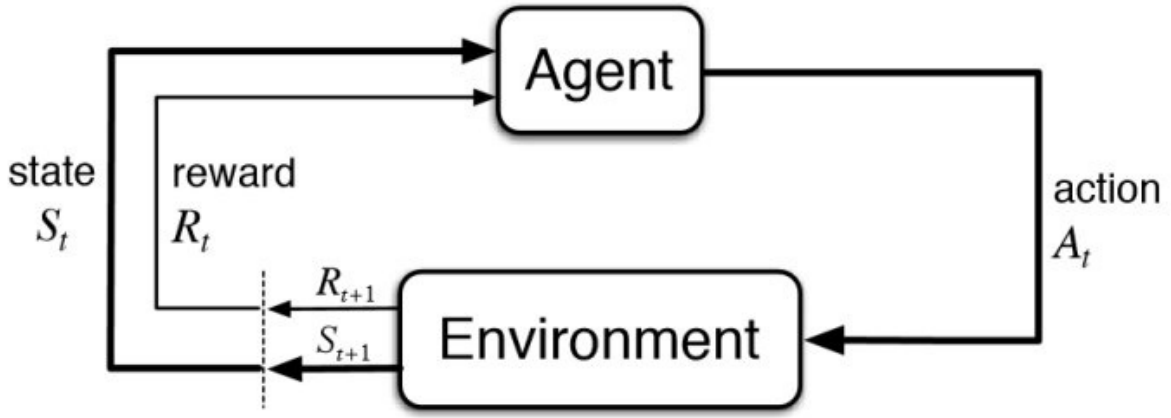
TA: Mateo Guaman Castro



Class Structure



How should we optimize this objective?



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

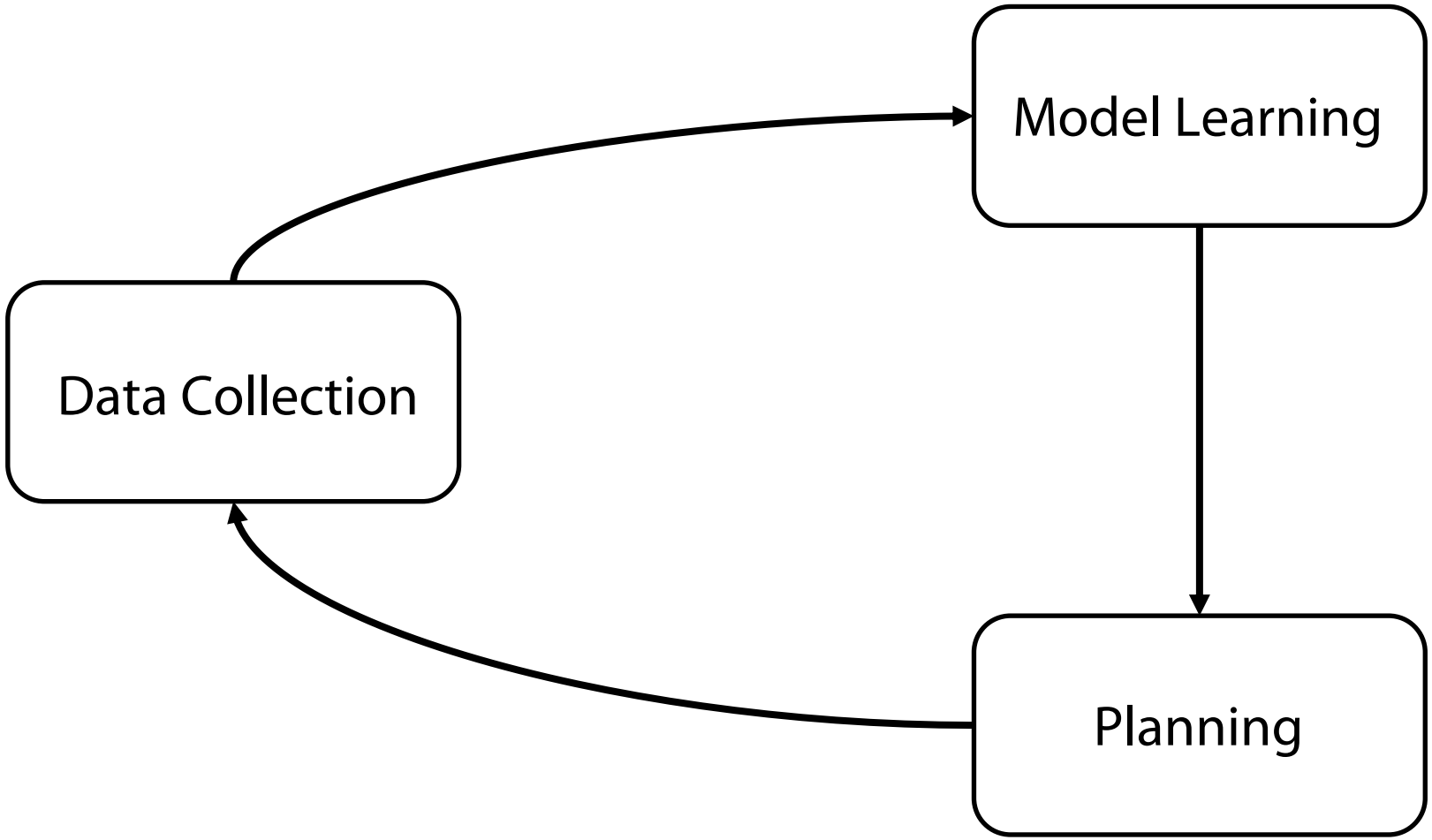
Gradient Ascent

Dynamic Programming

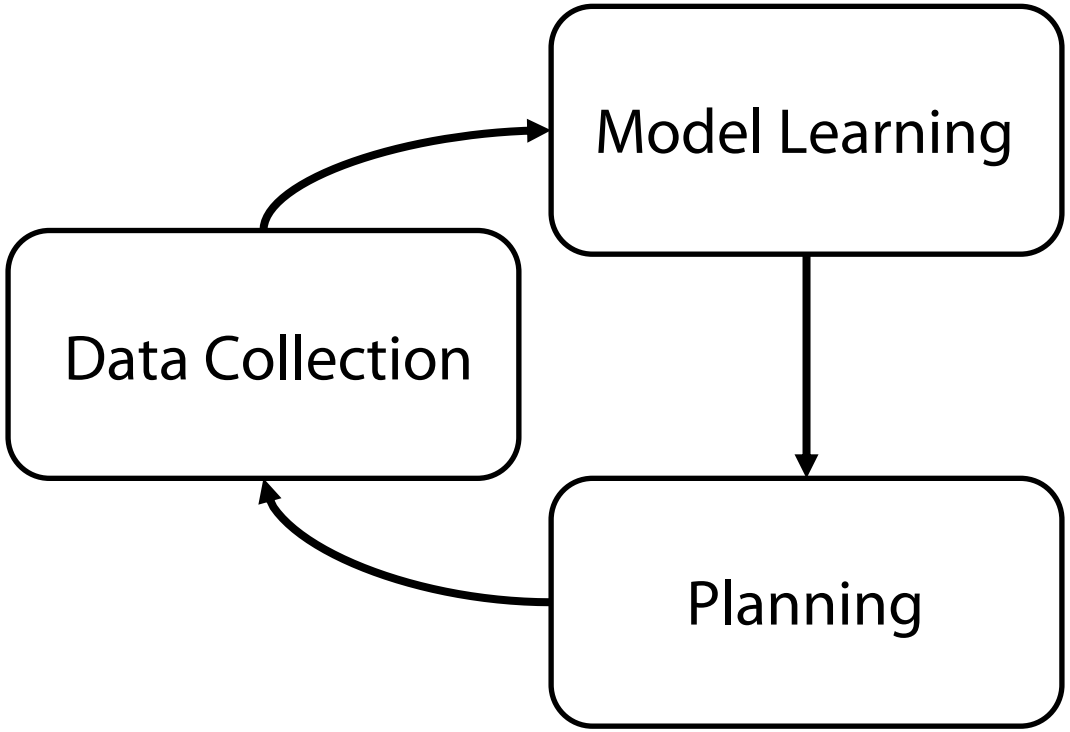
Model-Based Optimization

Each method has it's own +/-

Model Based RL – A template



Model Based RL – Naïve Algorithm (v0)

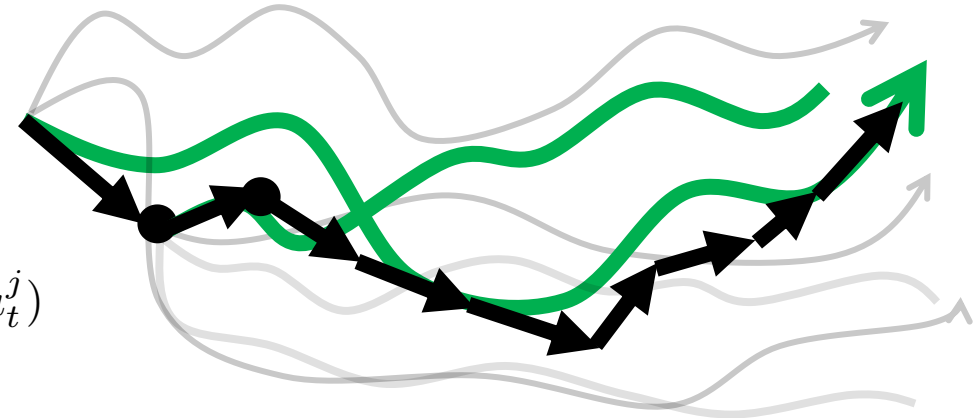


Data collected from planner

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

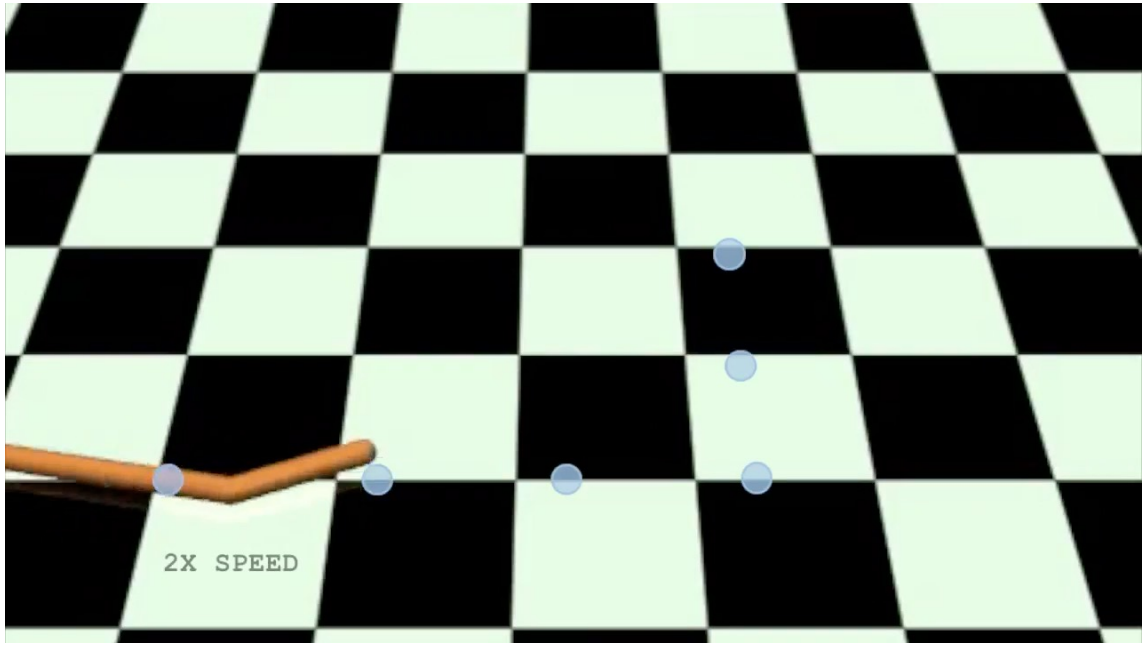


$$\arg \max_{a_0^j, a_1^j, \ldots, a_T^j} \sum_{t=0}^{T} r(\hat{s}_t^j, a_t^j)$$

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

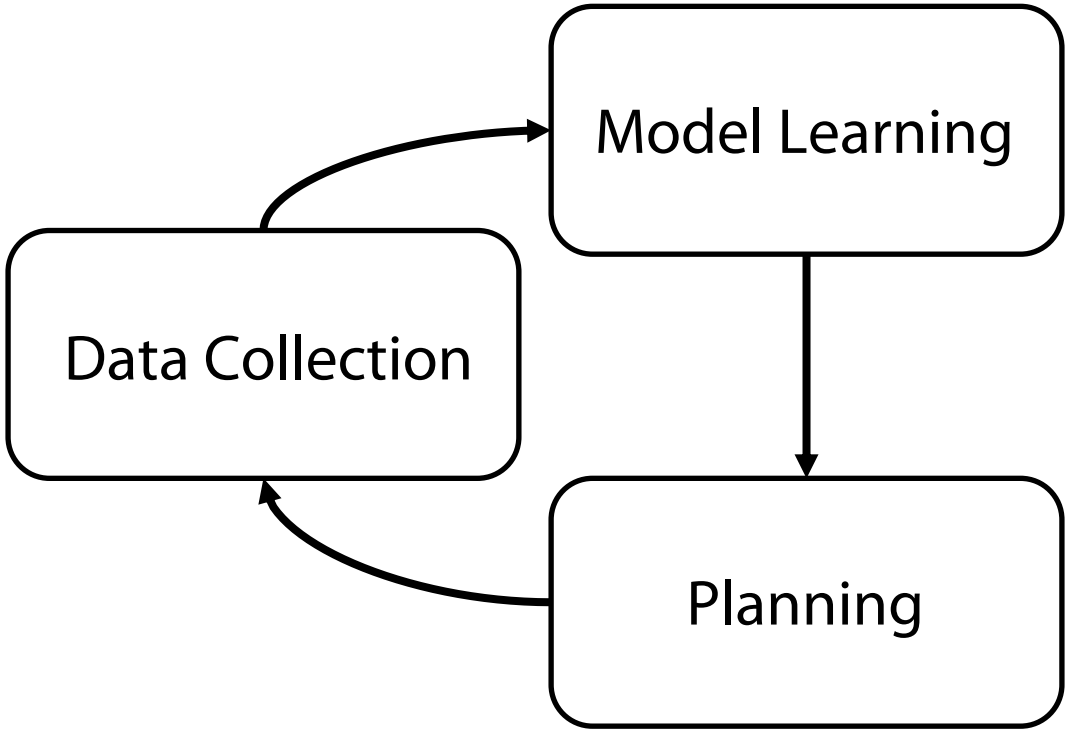
Better than open loop planning because of feedback

Does it work?



Just 20 minutes of training time with random data!

What might be the issue?



Data collected from planner

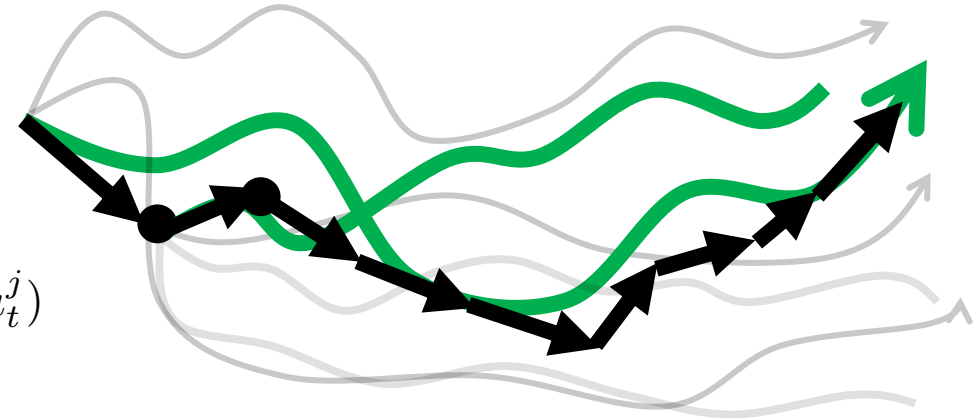
Maximum likelihood supervised Learning

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

Planning with Shooting + MPC

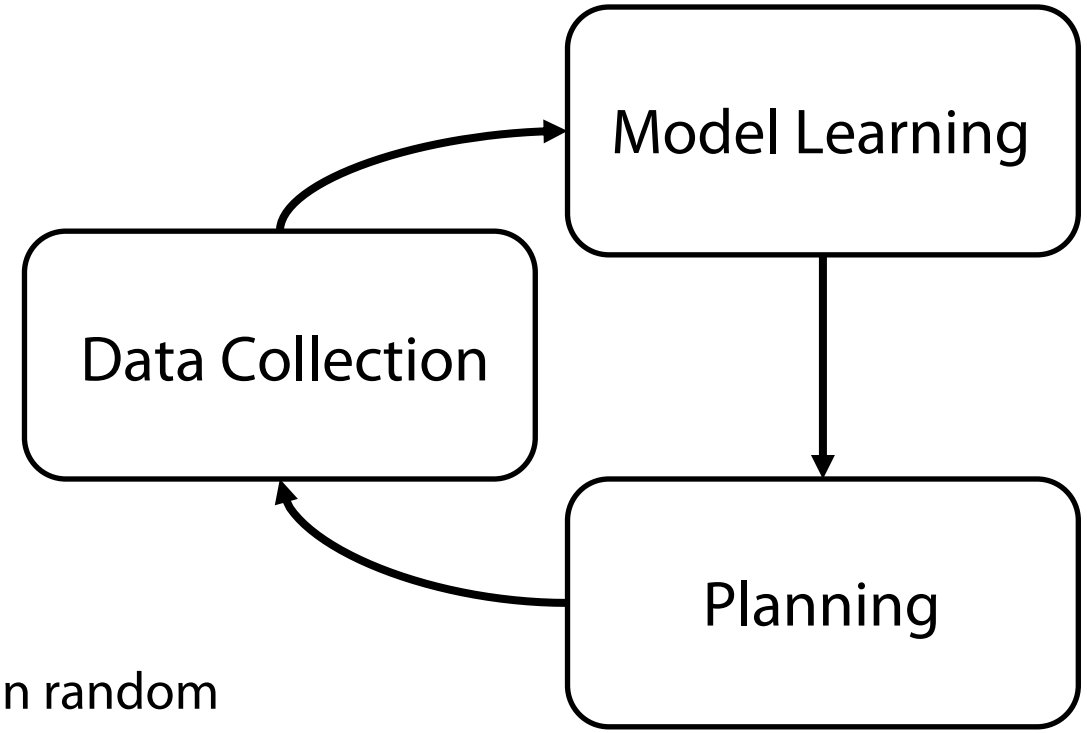
$$\arg \max_{a_0^j, a_1^j, \ldots, a_T^j} \sum_{t=0}^{T} r(\hat{s}_t^j, a_t^j)$$

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



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

Model Based RL – Better Sampling Methods (v1)

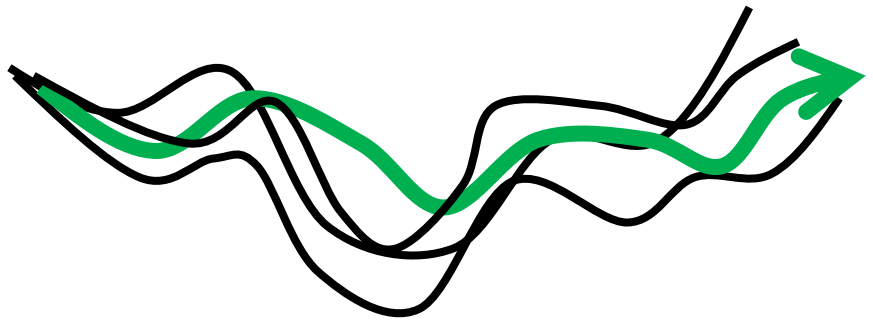


Data collected from planner

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

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

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

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

Deriving MPPI from first principles

Let's consider a nominal action trajectory

$$U = (u_0, \ldots, u_{H-1})$$

And some perturbations to this trajectory

$$u_t^i = u_t + \epsilon_t^i,$$

Consider the cost of such a perturbation

$$S(\epsilon^i) = \sum_{t=0}^{H-1} \ell(x_t^i, u_t + \epsilon_t^i) + \ell_f(x_H^i)$$

What do we want

→ best cost while staying close to current sampling distribution

$$q^\star = \arg\min_q \; \mathbb{E}_{\epsilon \sim q}\left[S(\epsilon)\right] + \lambda KL(q(\epsilon) \,\|\, p(\epsilon))$$

Deriving MPPI from first principles

$$q^\star = \arg\min_q \left[\int q(\epsilon) S(U+\epsilon)\, d\epsilon + \lambda \int q(\epsilon) \log \frac{q(\epsilon)}{p(\epsilon)}\, d\epsilon \right]$$

Optimization

$$\int q(\epsilon) d\epsilon = 1.$$

Lagrangian

$$\mathcal{L}(q,\eta) = \int q(\epsilon) S(U+\epsilon)\, d\epsilon + \lambda \int q(\epsilon) \log \frac{q(\epsilon)}{p(\epsilon)}\, d\epsilon + \eta \left(\int q(\epsilon) d\epsilon - 1 \right).$$

Take gradient and set to 0

Deriving MPPI from first principles

$$\mathcal{L}(q,\eta) = \int q(\epsilon) S(U+\epsilon)\, d\epsilon + \lambda \int q(\epsilon) \log \frac{q(\epsilon)}{p(\epsilon)}\, d\epsilon + \eta \left(\int q(\epsilon) d\epsilon - 1 \right).$$

$$S(U+\epsilon) + \lambda \left(\log \frac{q^\star(\epsilon)}{p(\epsilon)} + 1 \right) + \eta = 0.$$

$$q^\star(\epsilon) = \frac{1}{Z} p(\epsilon) \exp\left(-\frac{1}{\lambda} S(U+\epsilon) \right).$$

(MPPI weights)

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

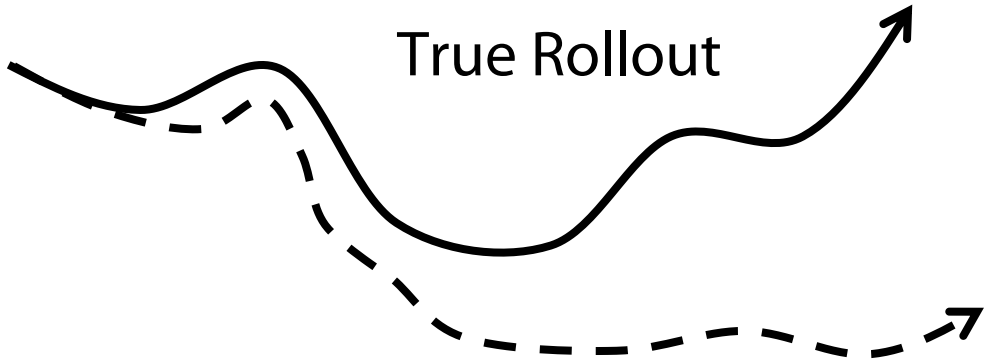
↓

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

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

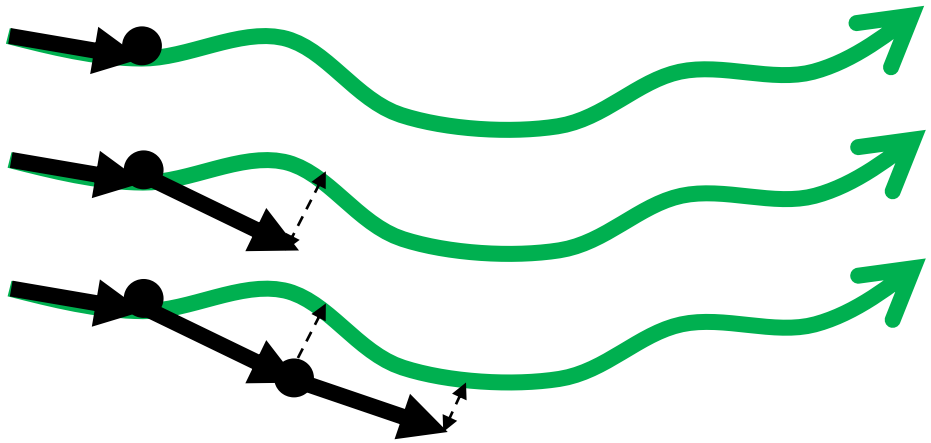
Simulation error – K step prediction error

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

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

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

Can be a challenging non-convex optimization!

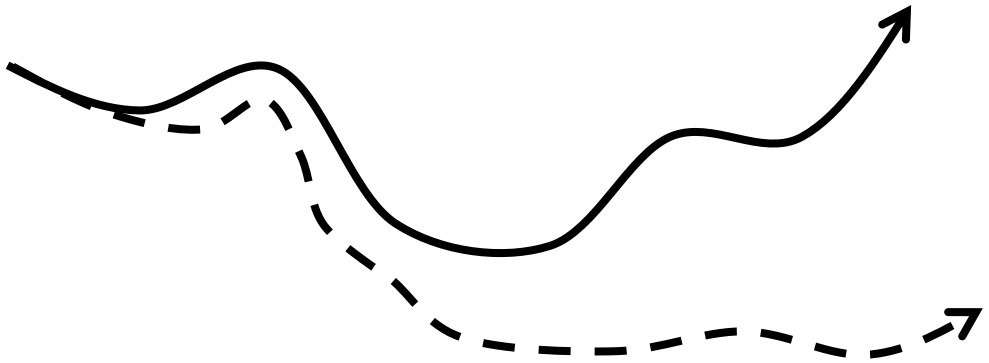


How might we deal with compounding error?

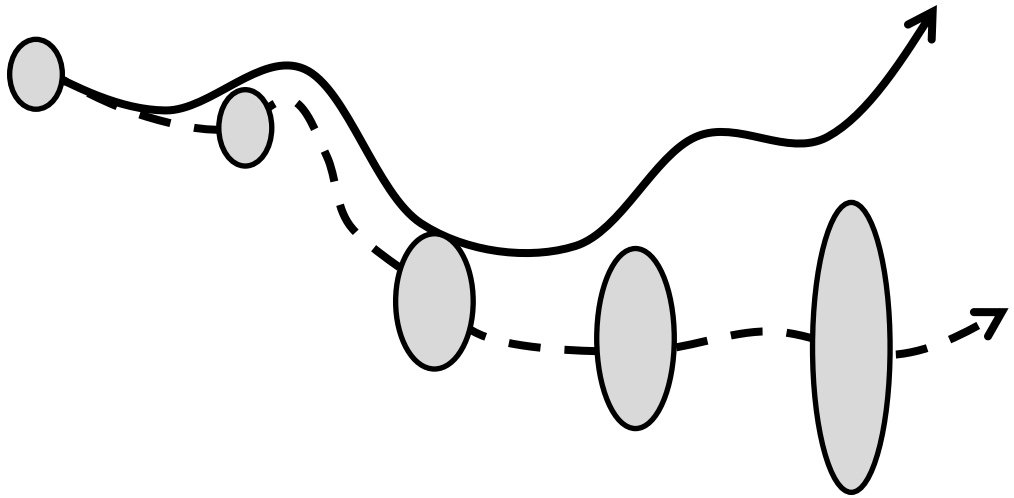
Idea 2: Estimate when OOD and account for it

→ Measure uncertainty!

Maximum likelihood models



Uncertainty-aware models



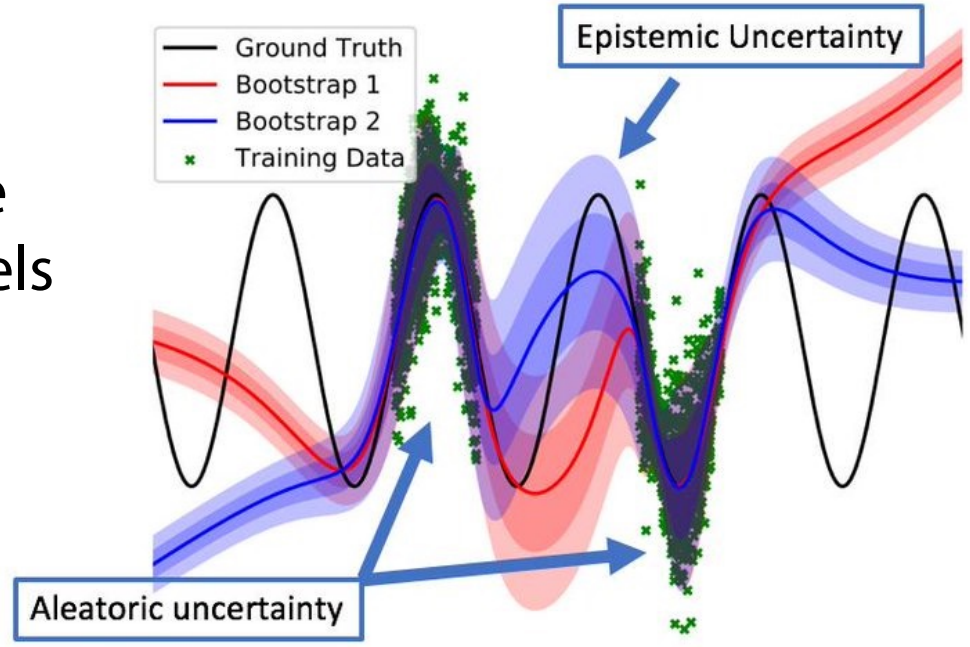
Being aware of uncertainty allows us to account for the effects of model bias!

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!

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

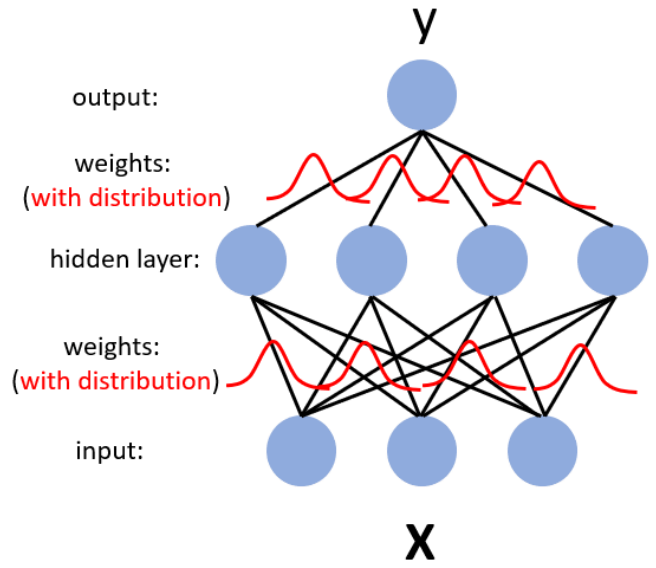
1. Bayesian neural networks
2. Ensemble methods
3. …

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



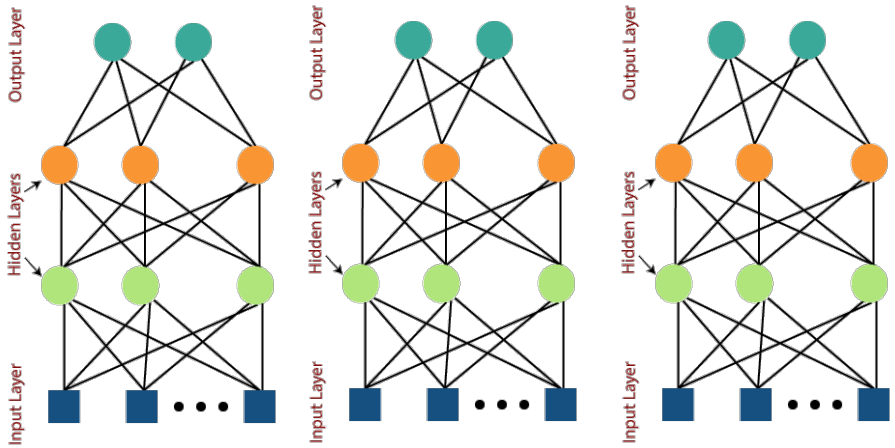
How might we measure uncertainty?

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

Difficult to estimate directly!

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

Learn an ensemble of models



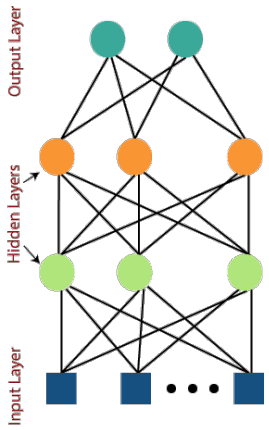
Low data regime → high ensemble variance

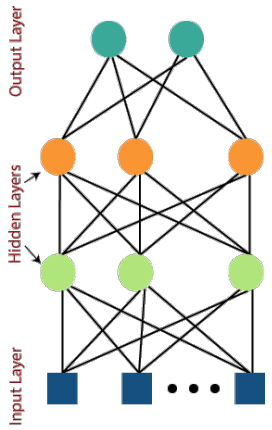
Approximate posterior

Easier and more expressive than BNNs!

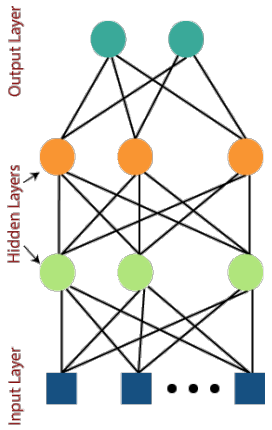
Model Based RL – Learning Ensembles of Dynamics Models

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





...

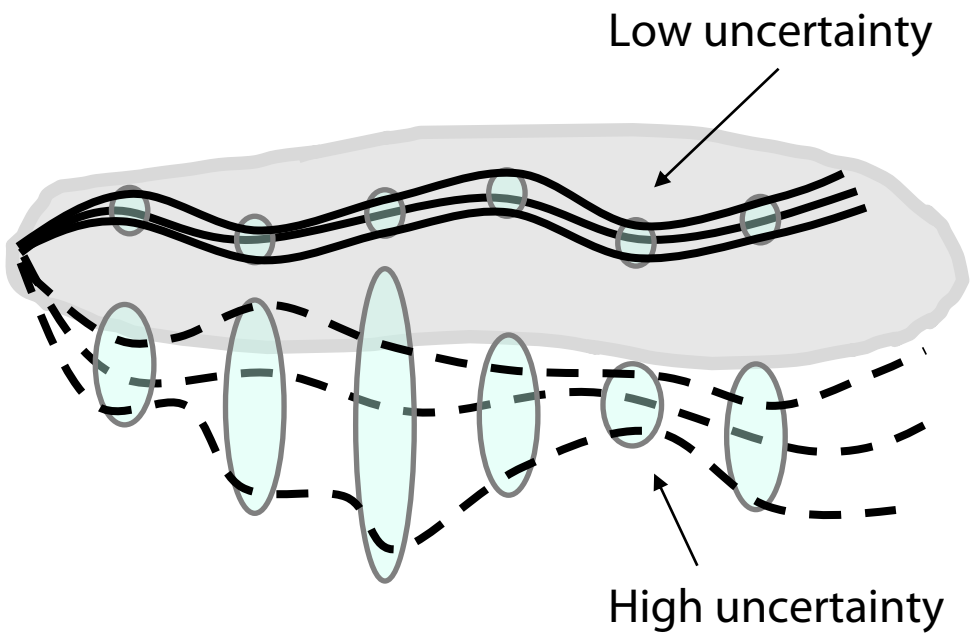


$$\max_\theta \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right] \quad \max_\theta \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right] \quad \max_\theta \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right]$$

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

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

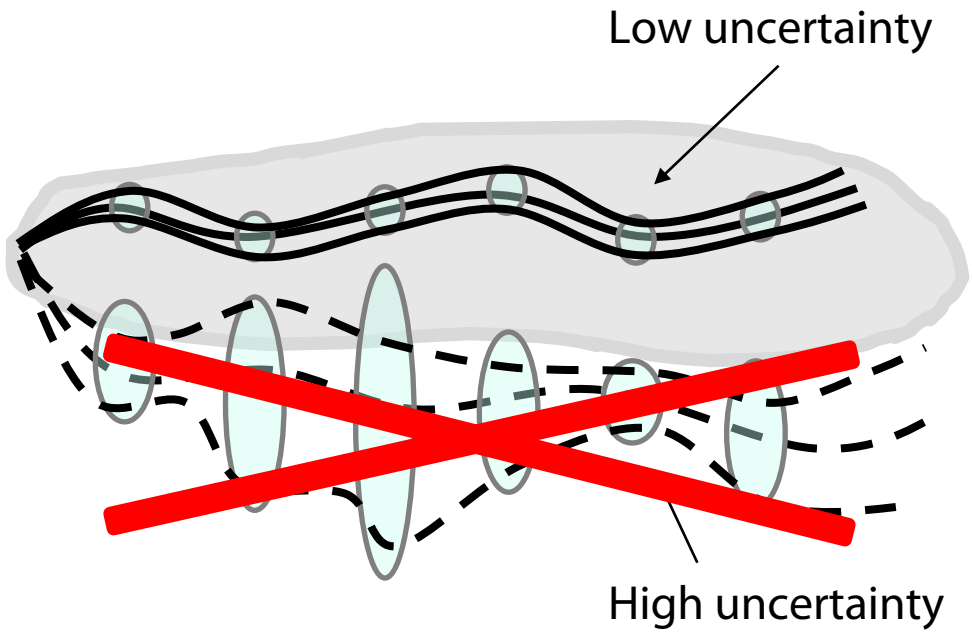
$$(\hat{s}_{t+1}^j)^i \sim \hat{p}_{\theta_i}(.|(\hat{s}_t^j)^i, a_t^j)$$

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



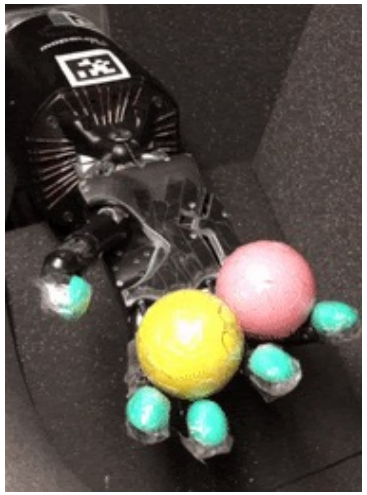
Penalize ensemble variance

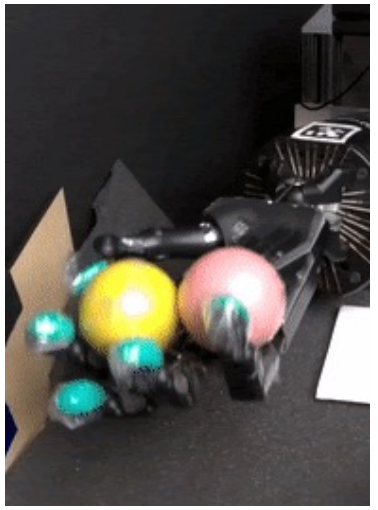
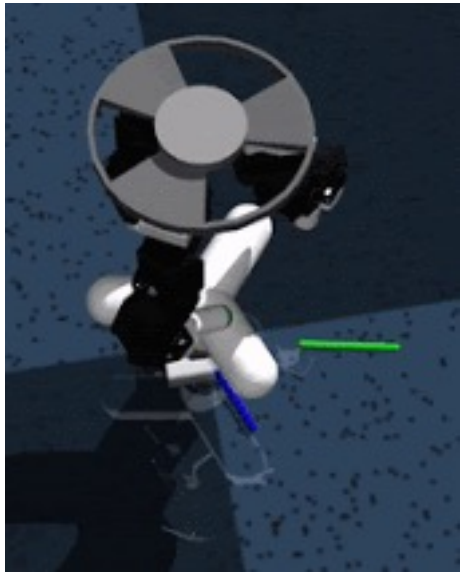
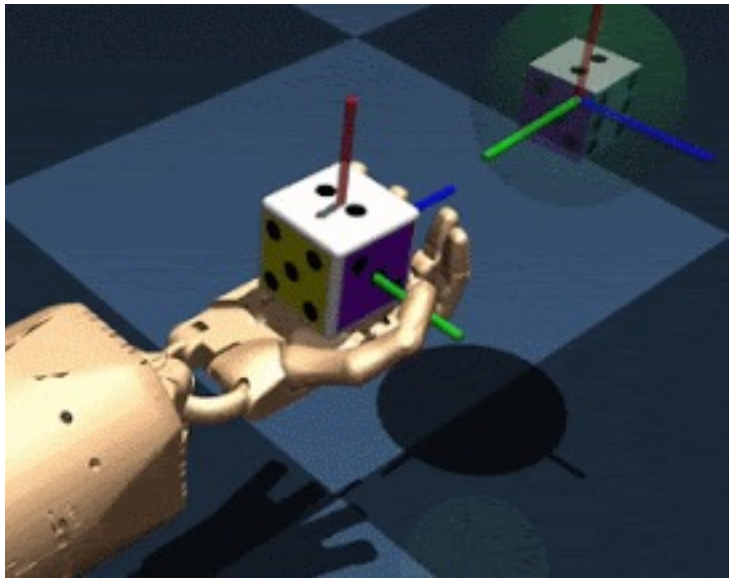
$$\arg \max_{(a_0^j, a_1^j, \ldots, a_T^j)_{j=1}^N} \sum_{i=1}^{K} \sum_{t=0}^{T} r((\hat{s}_t^j)^i, a_t^j) - \lambda \mathrm{Var}((\hat{s}_t^j)^i)$$

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

Avoids overly OOD settings since these states are explicitly penalized

Does this work?



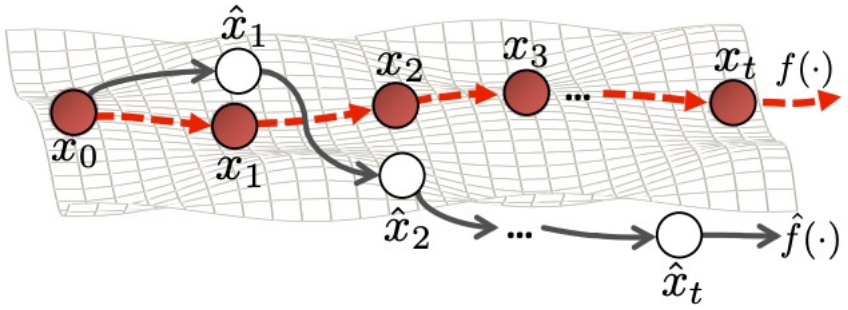






How might we deal with compounding error?

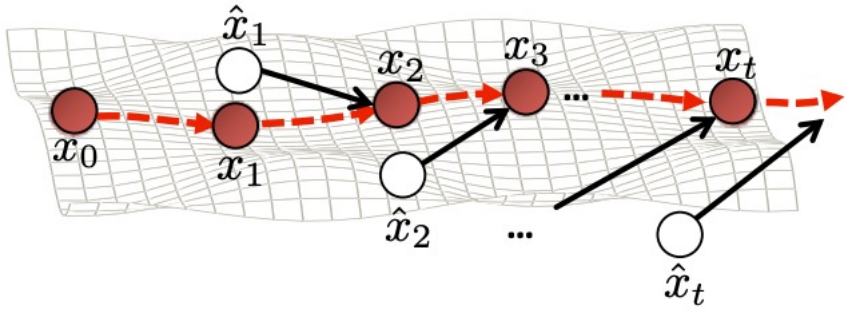
Idea 3: Cast this as an imitation learning problem

→ 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

↓

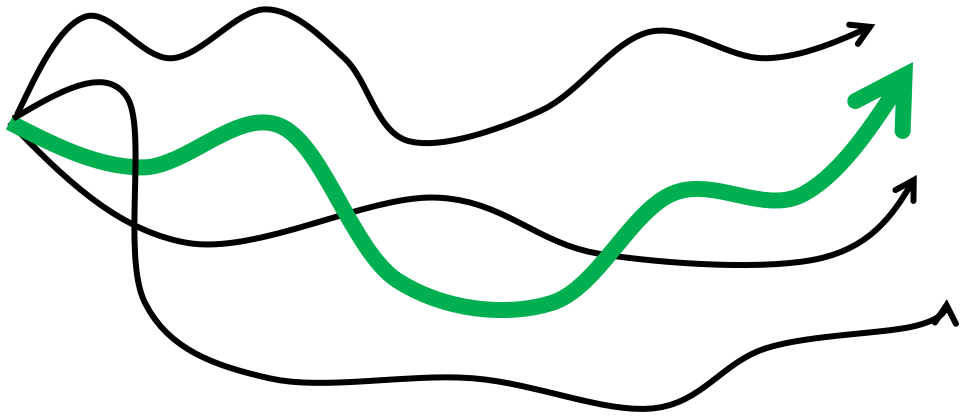
Model based RL v4 → latent space models with images

↓

Model based RL v5 → From MPPI to MCTS

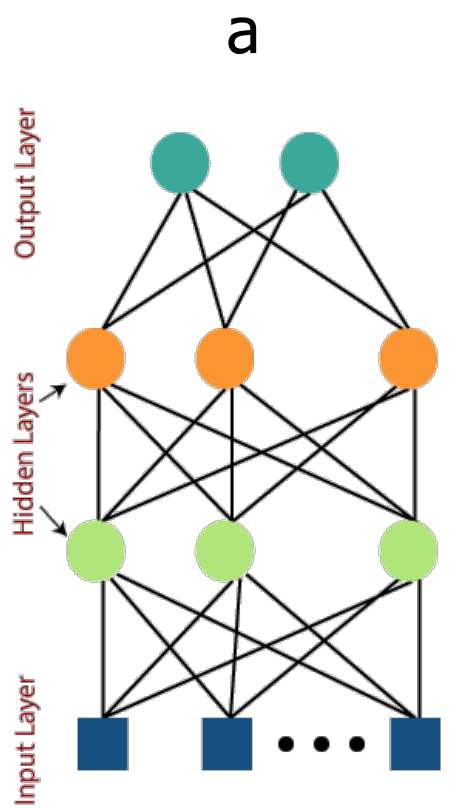
What might be the issue?

Huge number of samples
needed to reduce variance



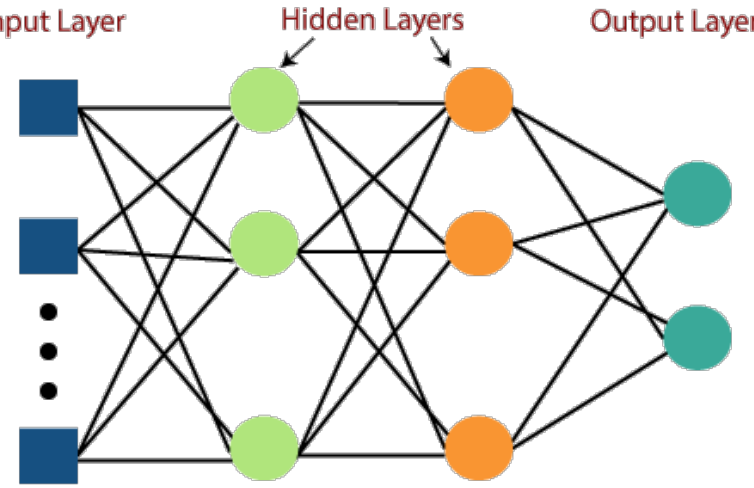
Extremely slow, hard to run in real time

Amortize planning
into a policy

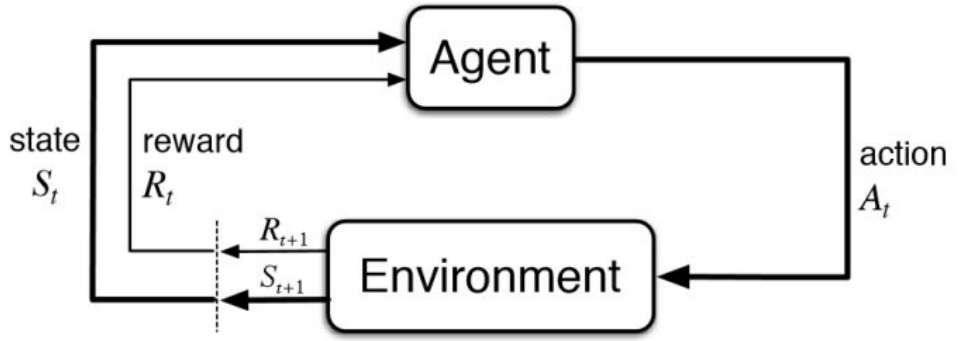


Speeding Up Model-Based Planning

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



Use model(s) to generate data for policy optimization



Can use PG or off-policy!

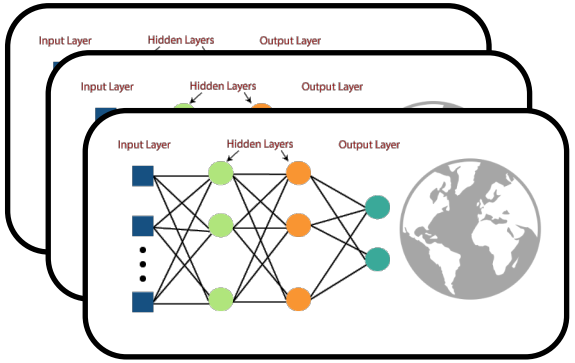
Generating Data for Policy Optimization

Learn models

Add Fake Sampled Data to Buffer

Policy Optimization

Train time



$\mathcal{D}$

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

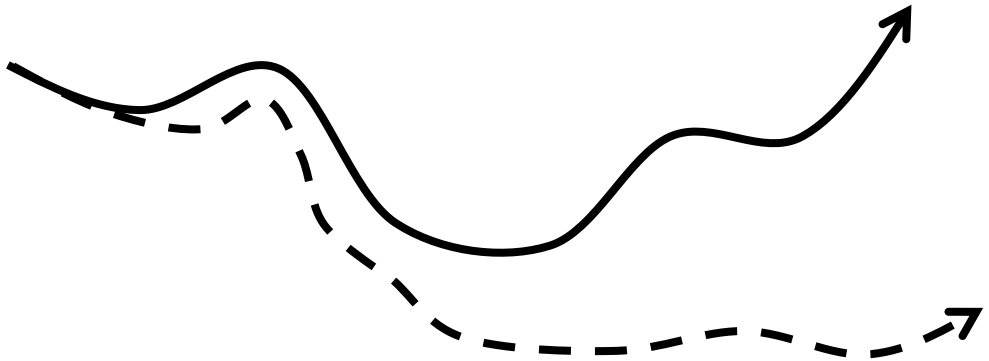
Rollout in environment

π_θ

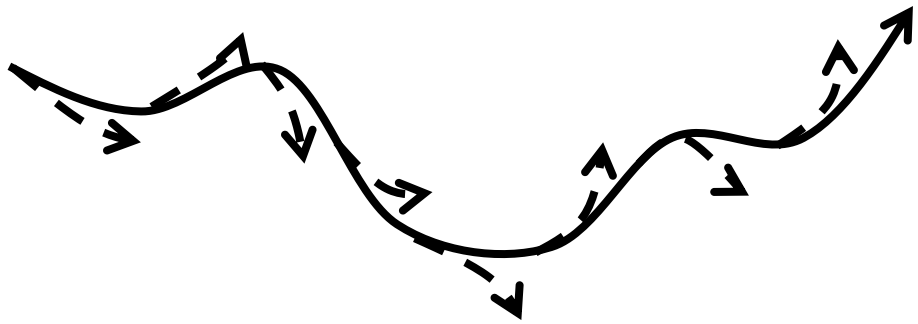
Test time

π_θ

What matters in generating data from models?



Long horizon rollouts can deviate

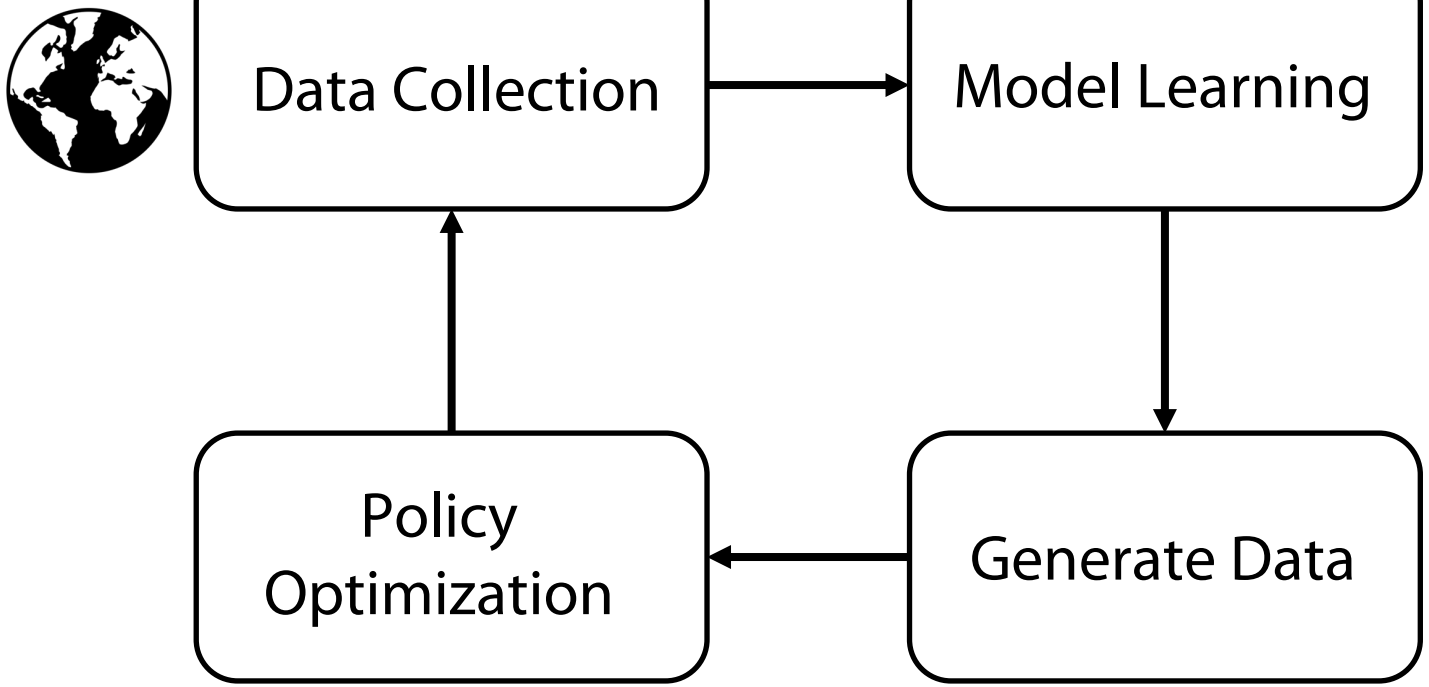


Short horizon rollouts deviate far less

Balance between off-policy coverage and compounding error

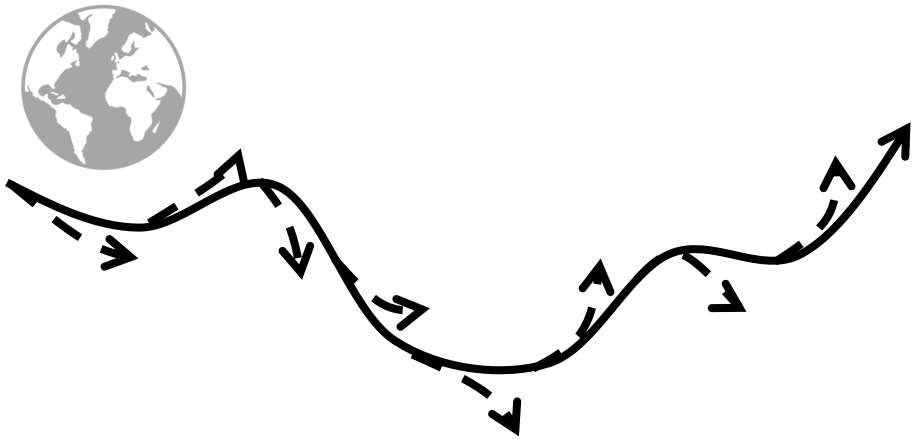
More at https://arxiv.org/abs/1906.08253

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

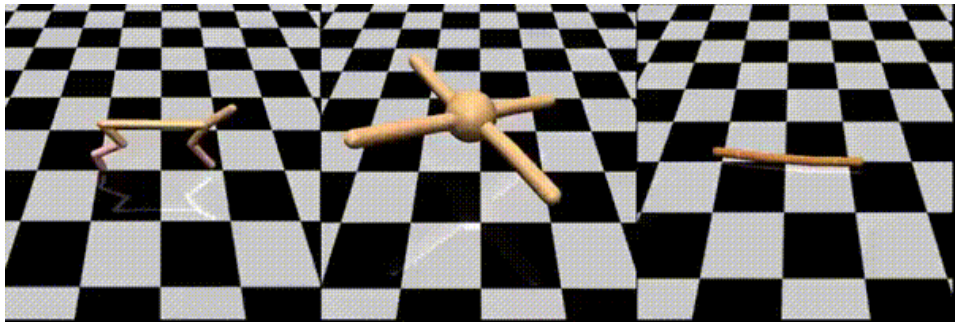
↓

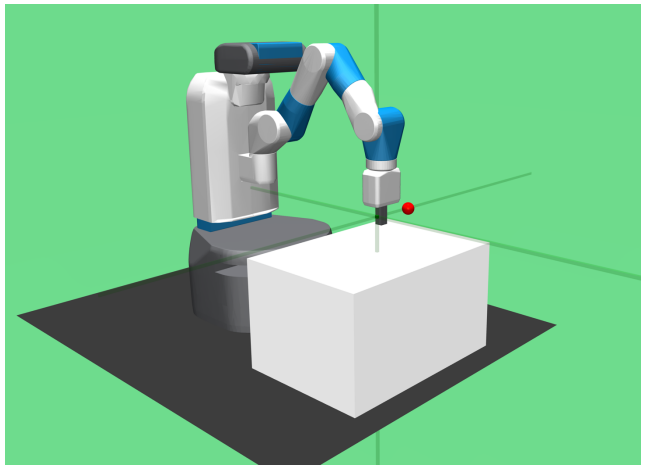
Model based RL v4 → latent space models with images

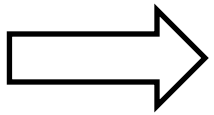
↓

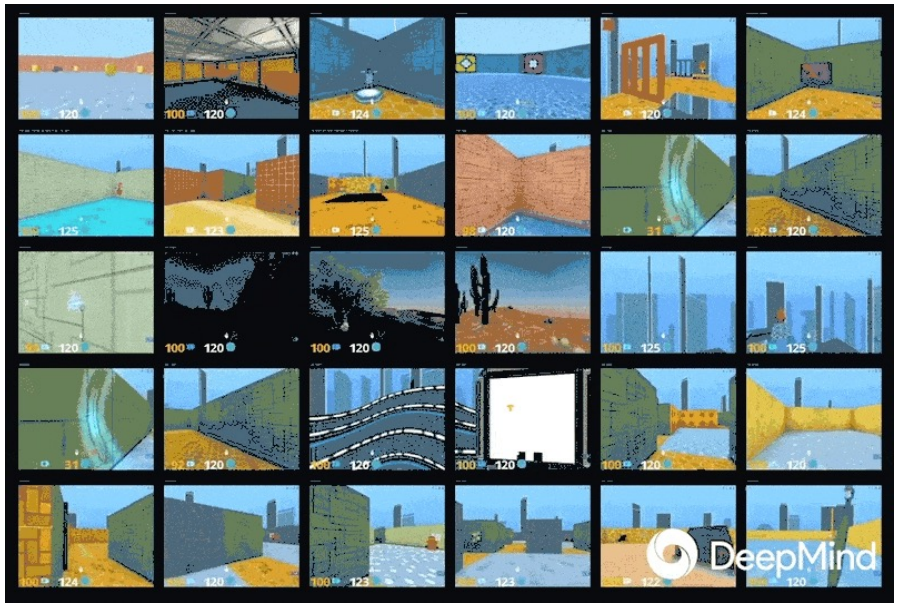
Model based RL v5 → From MPPI to MCTS

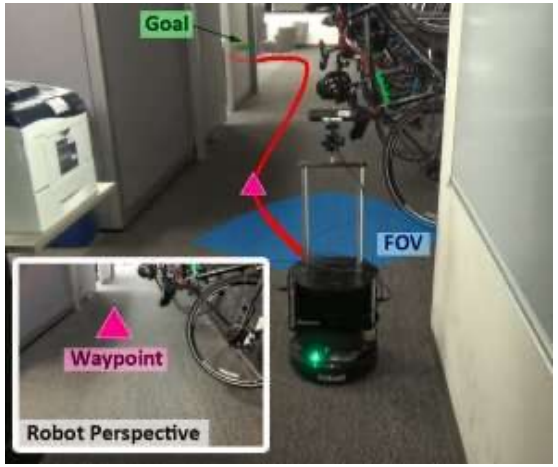
What about images?









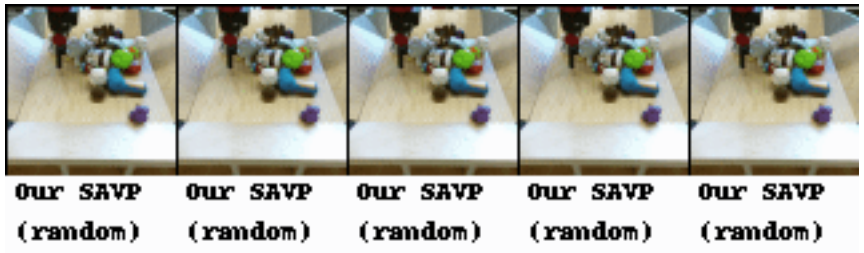


State based domains

Image based domains

Why is learning from images hard?

Generative modeling is videos, challenging to model multimodal correlated predictions

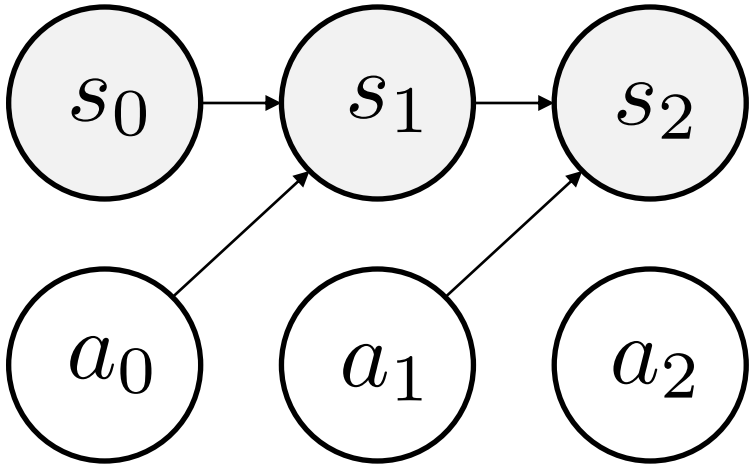


Partially observable!

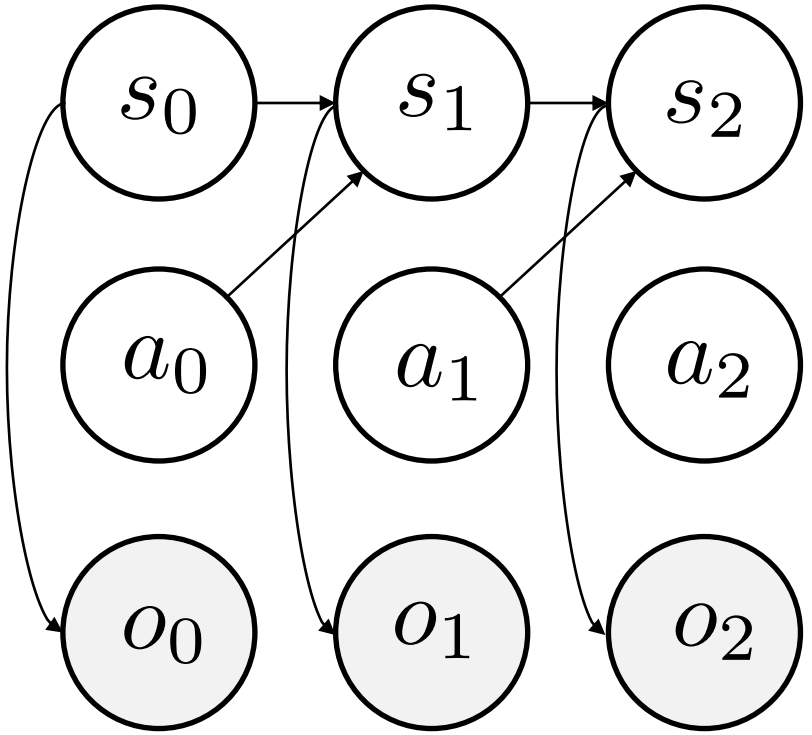
Long horizon predictions in video space can be challenging!

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

Fully observed – Markovian case



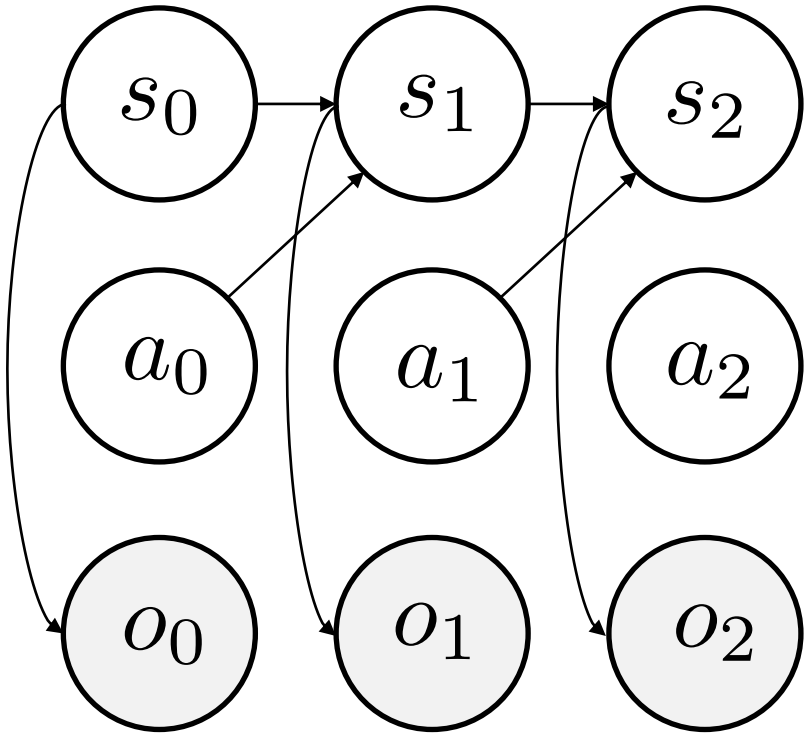
Partially observed – Non-Markovian case



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 $p_\eta(s_{t+1}|s_t, a_t)$

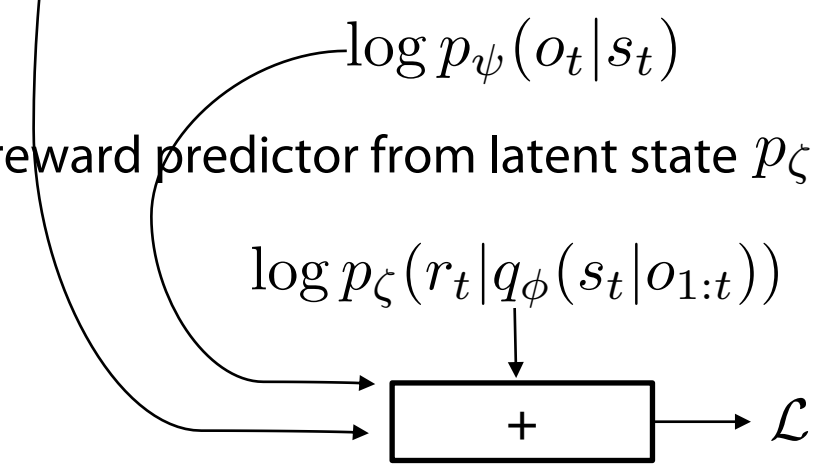
$$\log p_\eta(q_\phi(s_{t+1}|o_{1:t+1})|q_\phi(s_t|o_{1:t}), a_t)$$

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

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

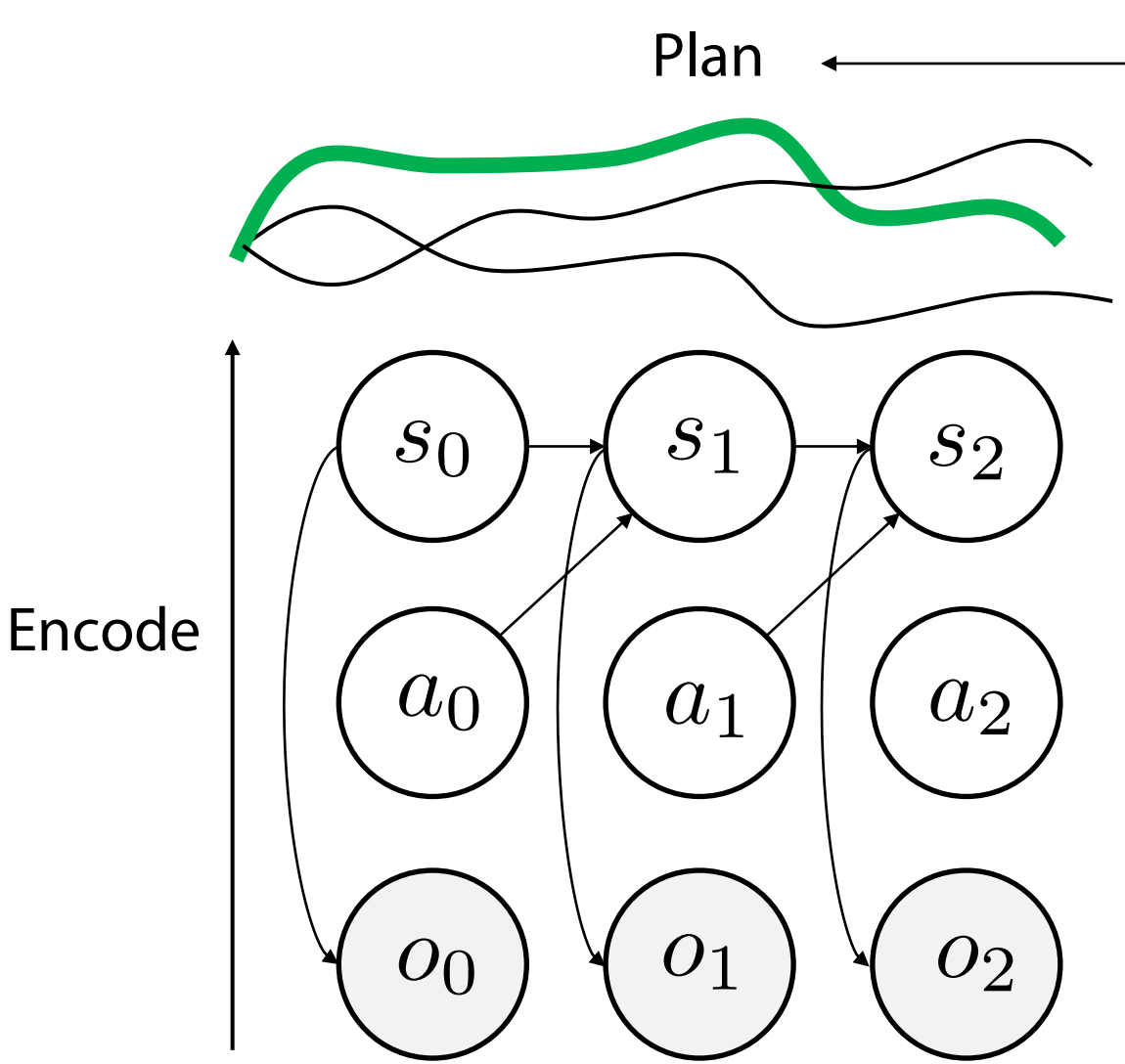
Learn reward predictor from latent state $p_\zeta(r_t|s_t)$

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



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

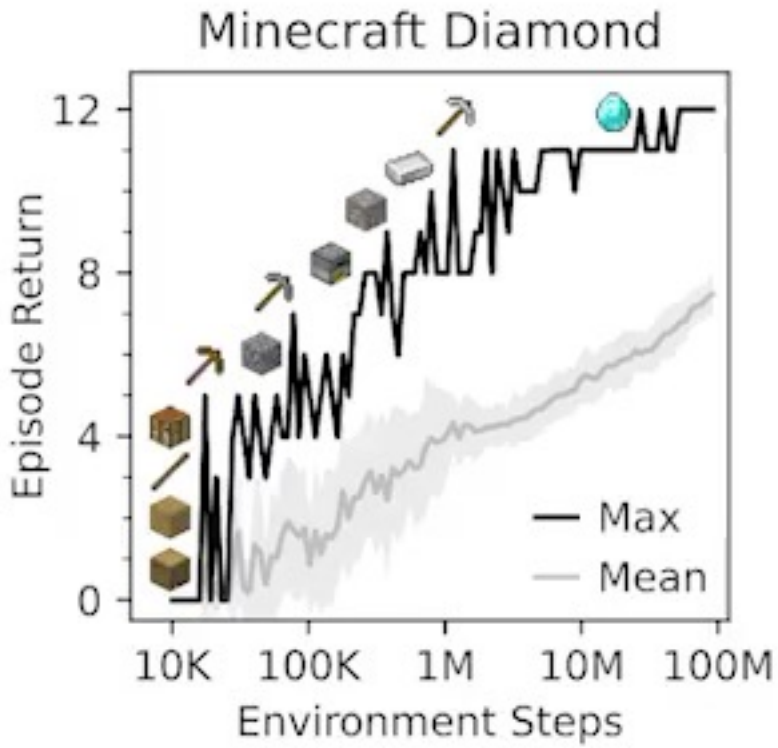
How do we **use** latent space models?



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

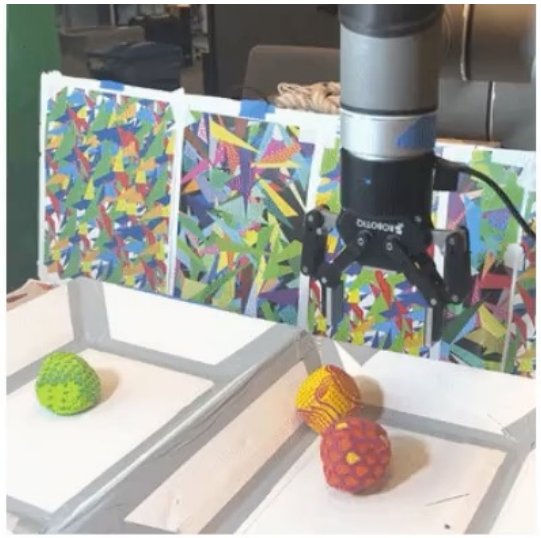
1. Avoids predicting image frames at planning time
2. Scales much better than image prediction
3. Allows for longer horizon predictions

Does this work?

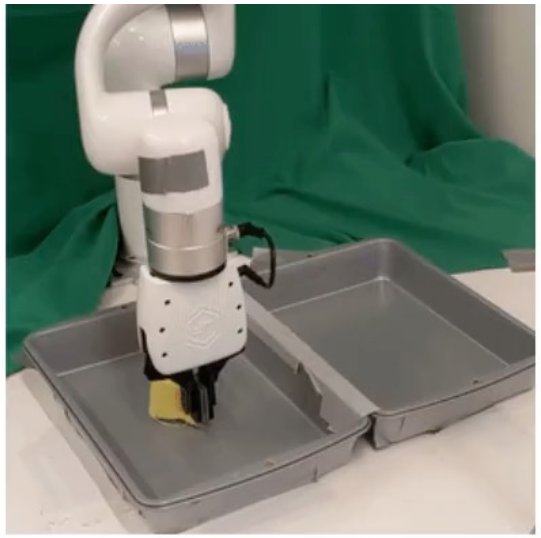


Does this work?

A1 Quadruped Walking



UR5 Multi-Object Visual Pick Place



XArm Visual Pick and Place

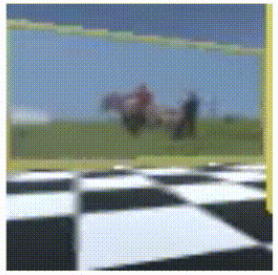
Sphero Ollie Visual Navigation

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

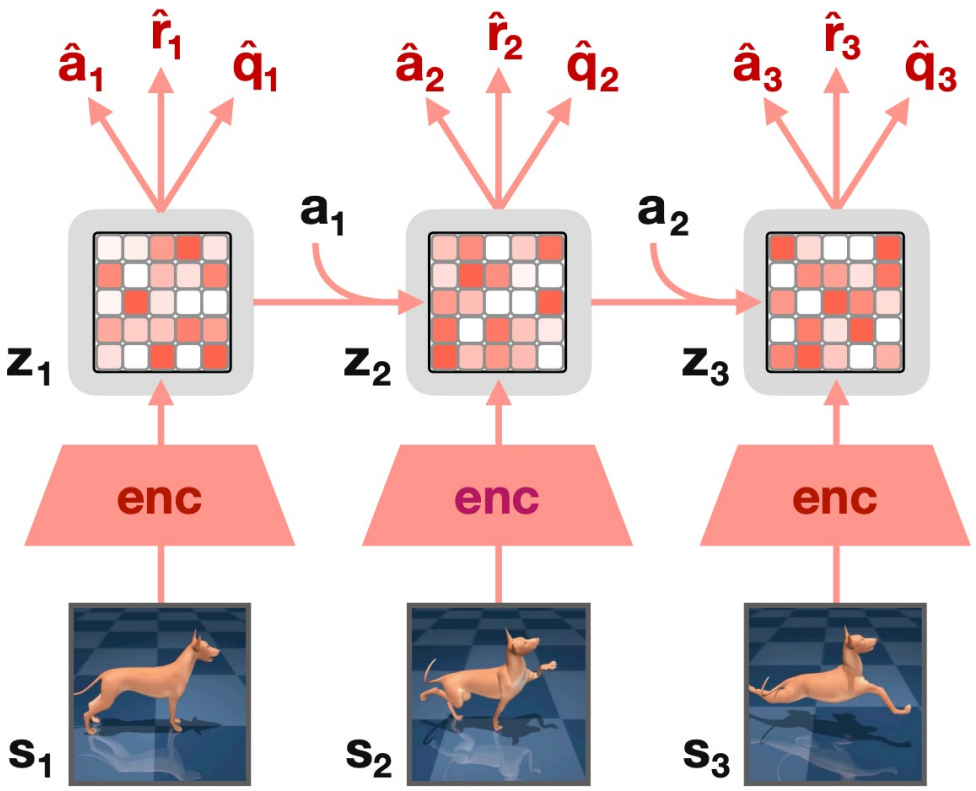
Reconstruction overindexes on pixel accuracy

Large-objects dominate reconstruction objective

Some solutions to avoid reconstruction

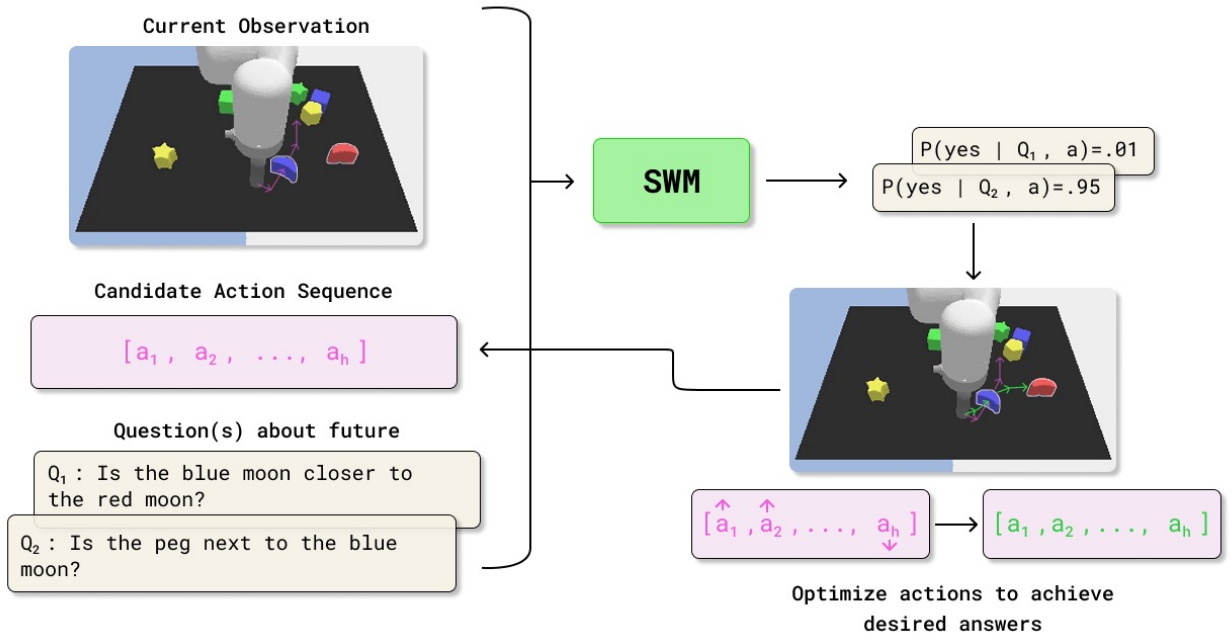
Predict “task-relevant” objectives

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

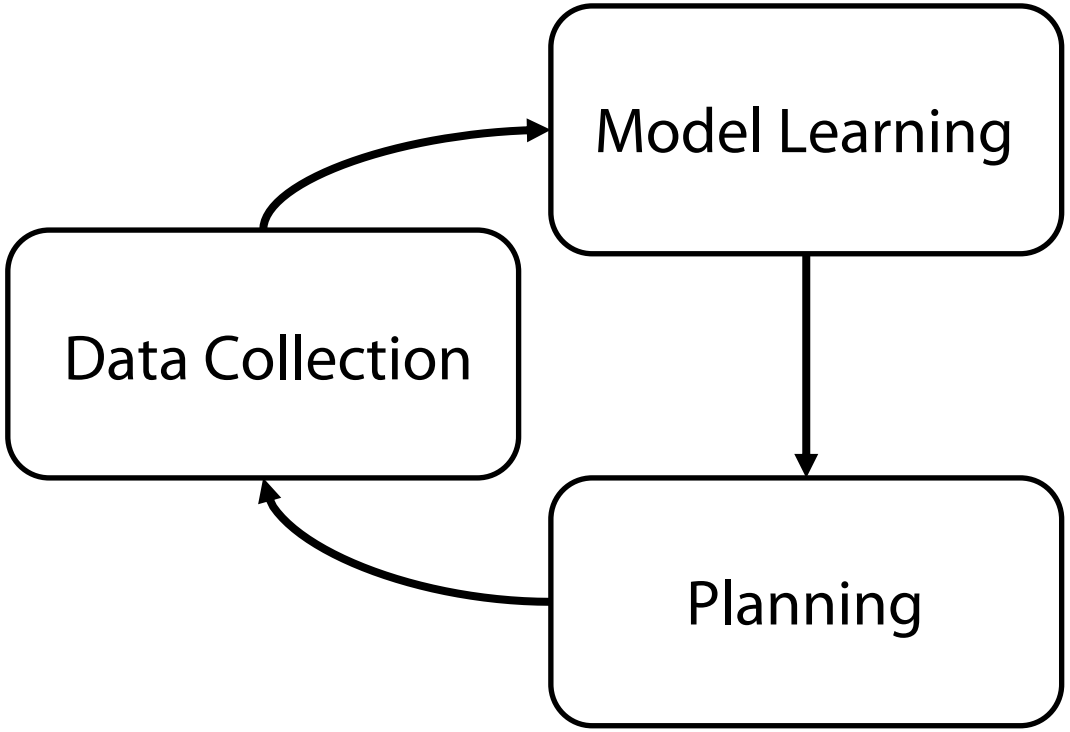
↓

Model based RL v4 → latent space models with images

↓

Model based RL v5 → From MPPI to MCTS

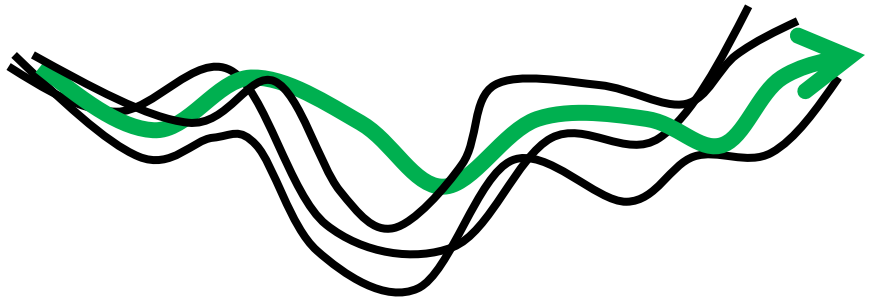
Can we do better than MPPI?



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

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

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

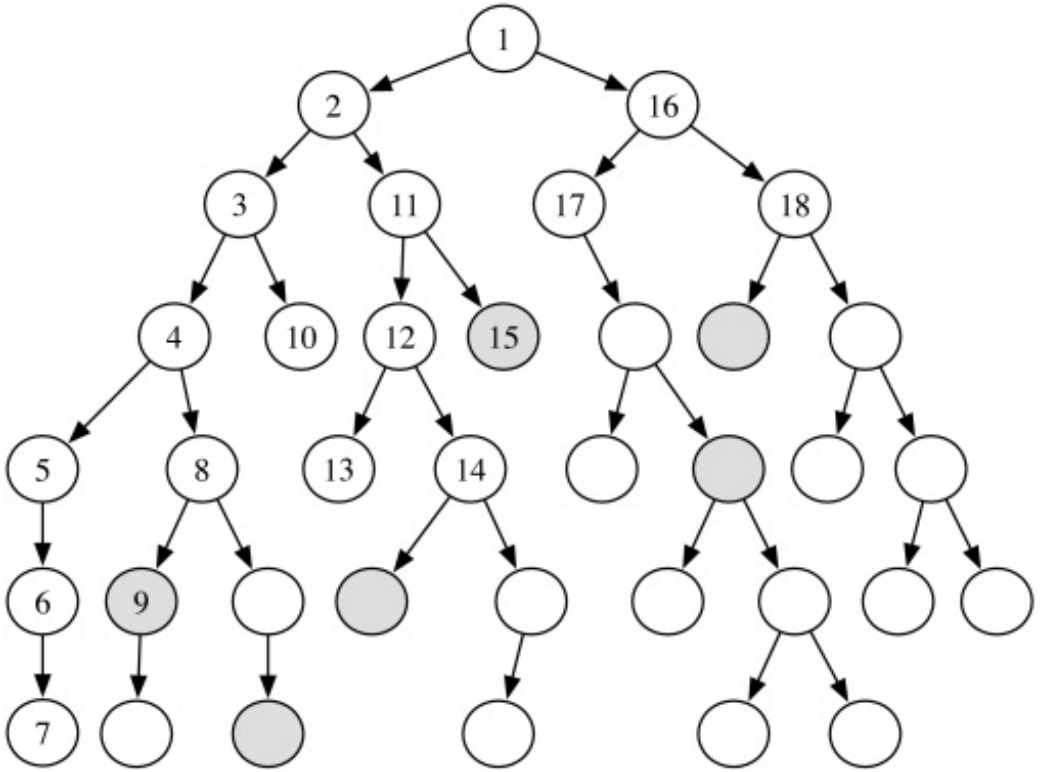


Can we improve on this planning strategy for discrete action spaces?

Discrete action spaces allow for better data structures

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

Are we wasting computation?



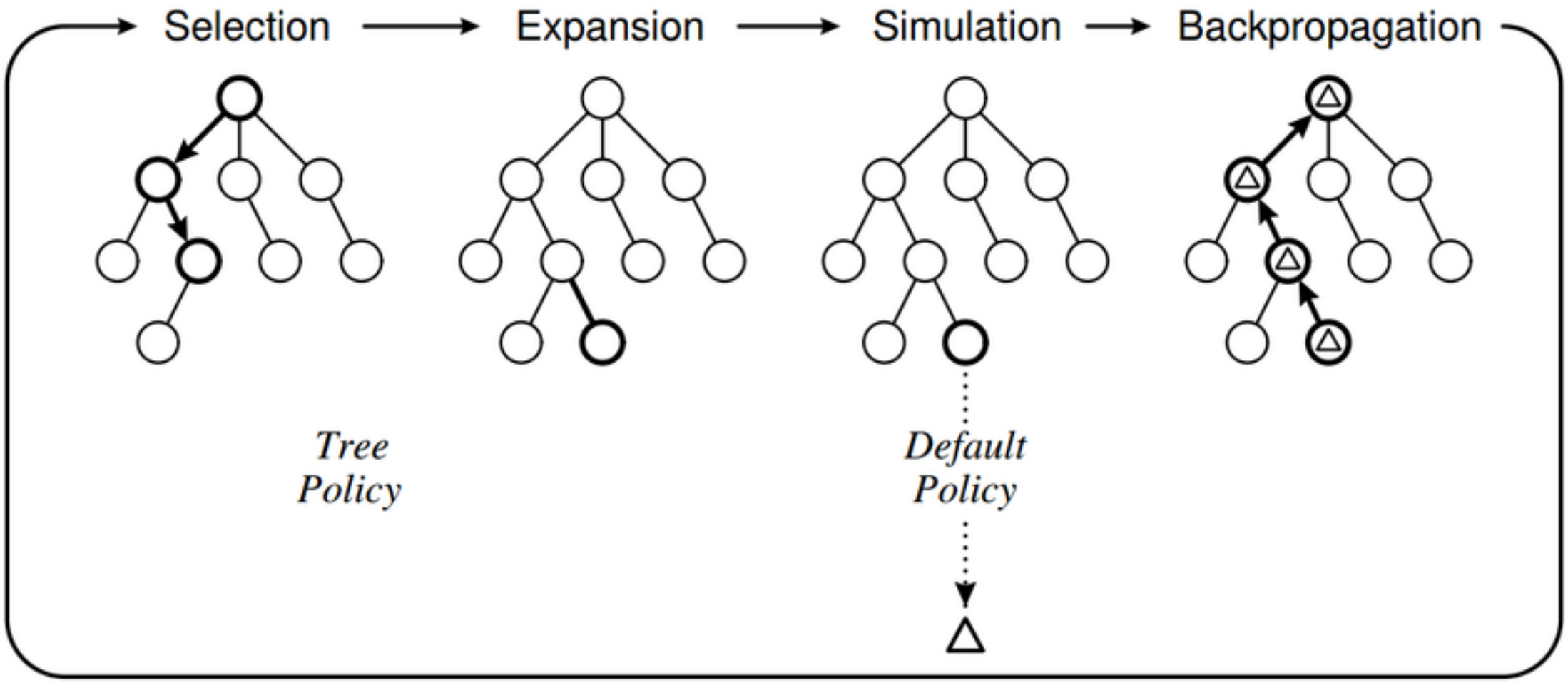
Tree structures can allow you to focus your computational effort

⇩

Can't build whole tree, how to approximate?

Monte-Carlo Tree Search

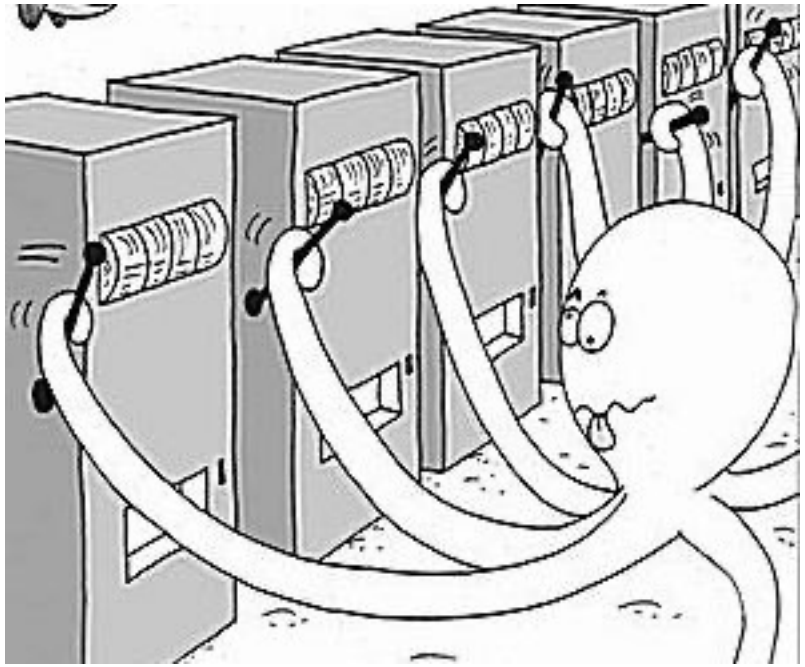
Easy way to approximate tree search for large tree structures, with known/learned models

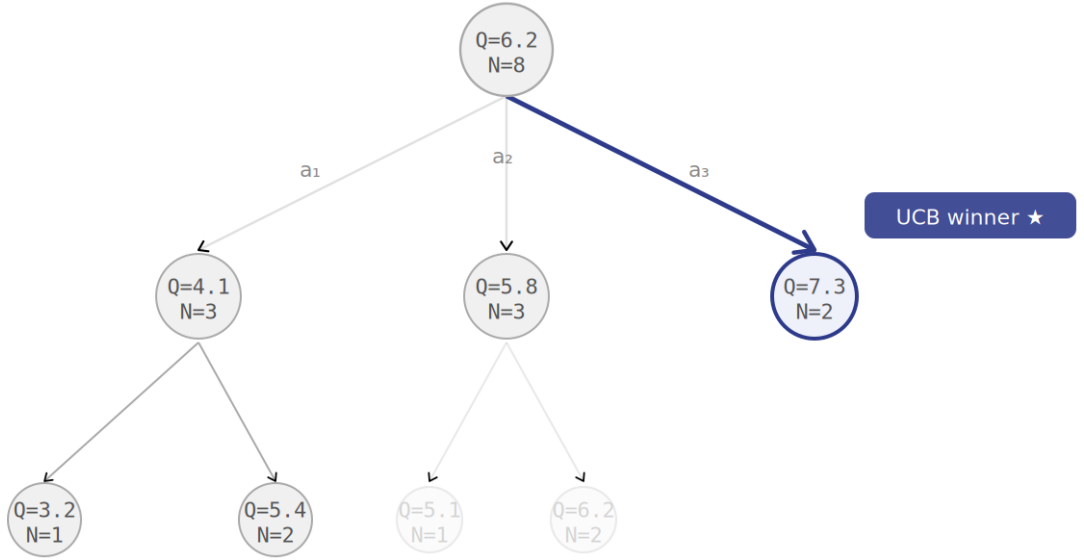


(https://www.cs.swarthmore.edu/~mitchell/classes/cs63/f20/reading/mcts.html)

Monte-Carlo Tree Search: Selection

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

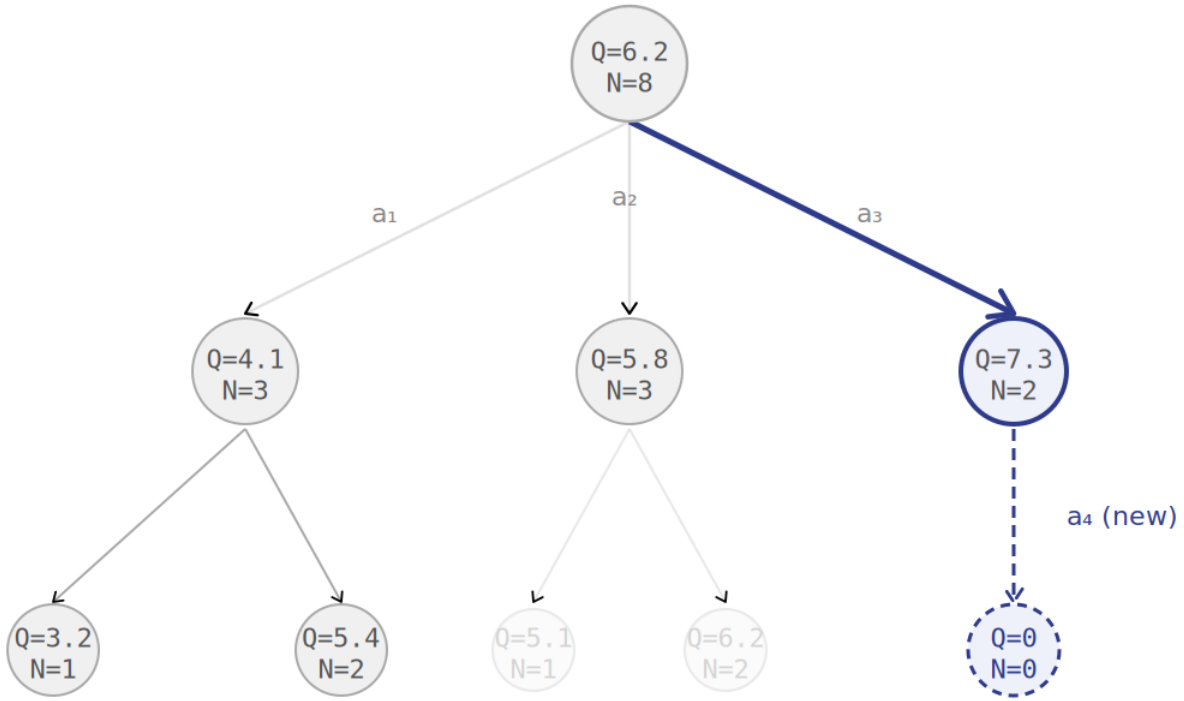




$$a^* = \arg\max_a \left[Q(s,a) + c\sqrt{\frac{\ln N(s)}{N(s,a)}} \right]$$

Monte-Carlo Tree Search: Expansion

Grow the tree by one level

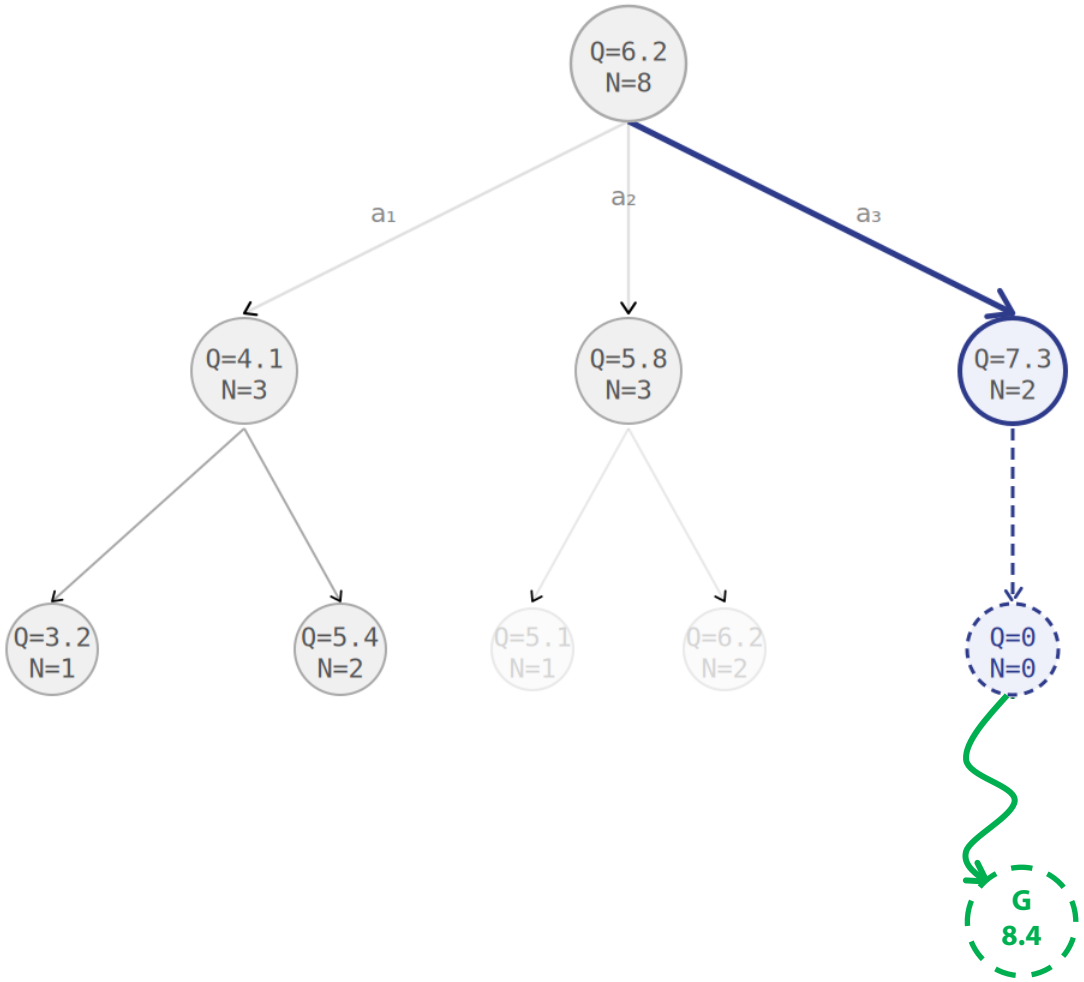


Set Q, N = 0

Updated over time through rollouts

Monte-Carlo Tree Search: Simulation

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

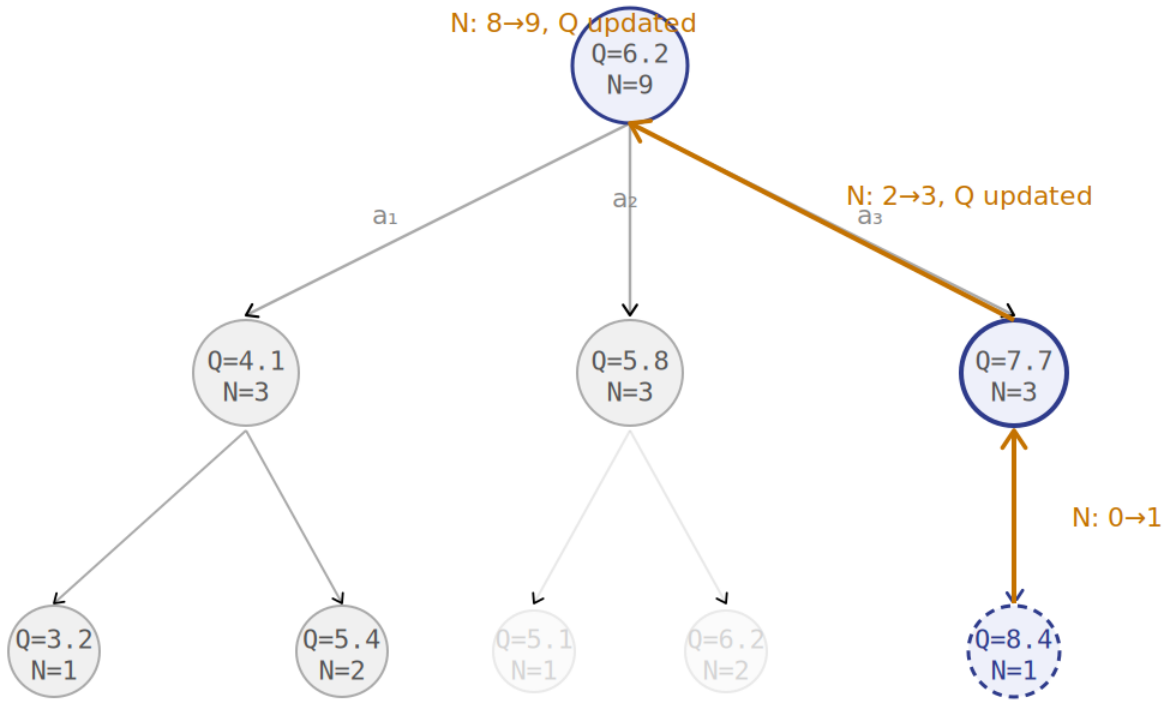


Run samples with policy and approximate value of the remaining tree

Start with random policy
→ Replace with rollouts from π

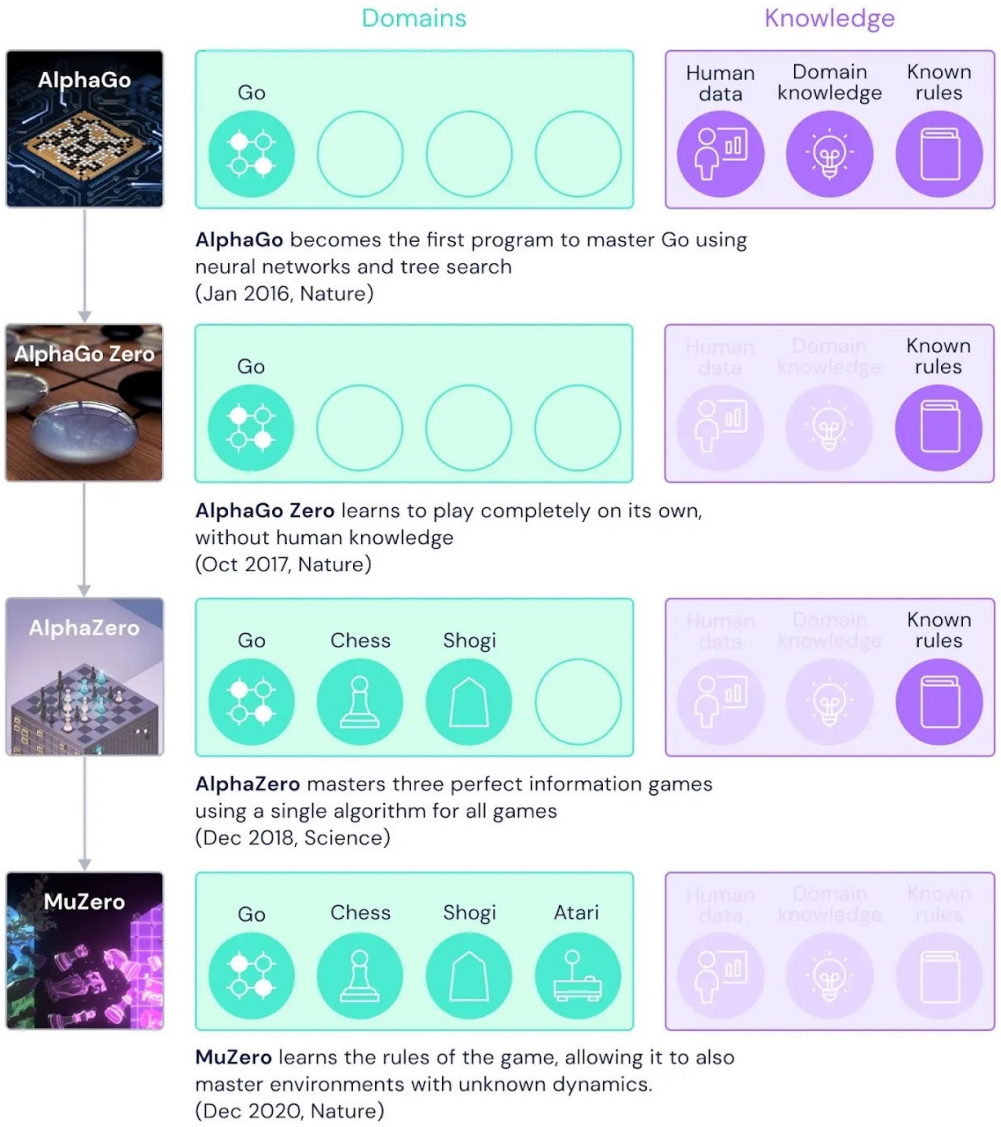
Monte-Carlo Tree Search: Backpropagation

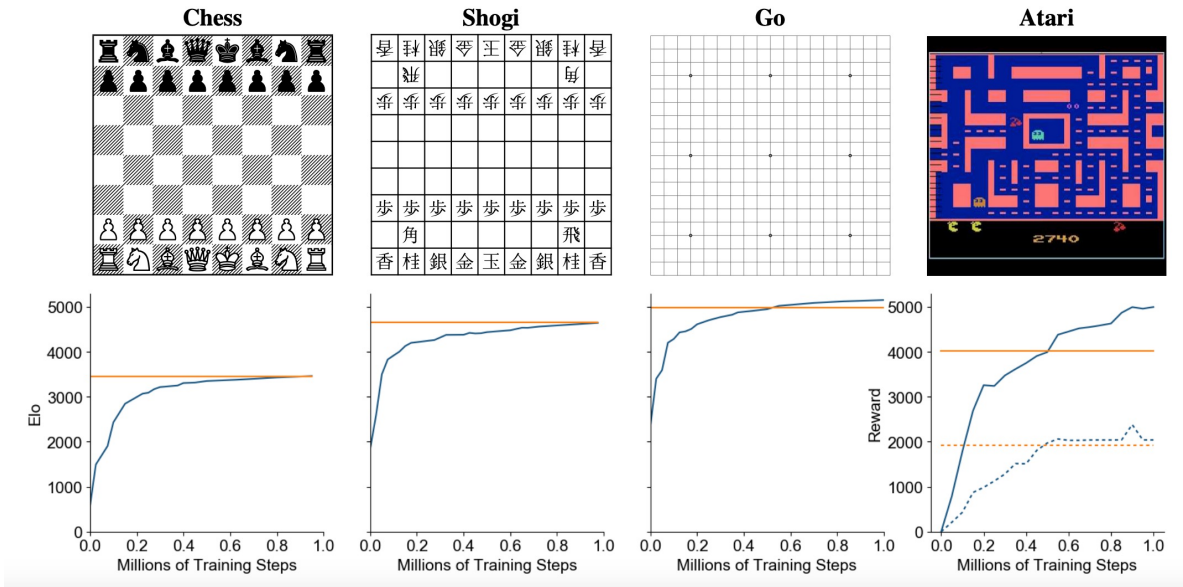
Go backwards and update every node in the tree



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

↓

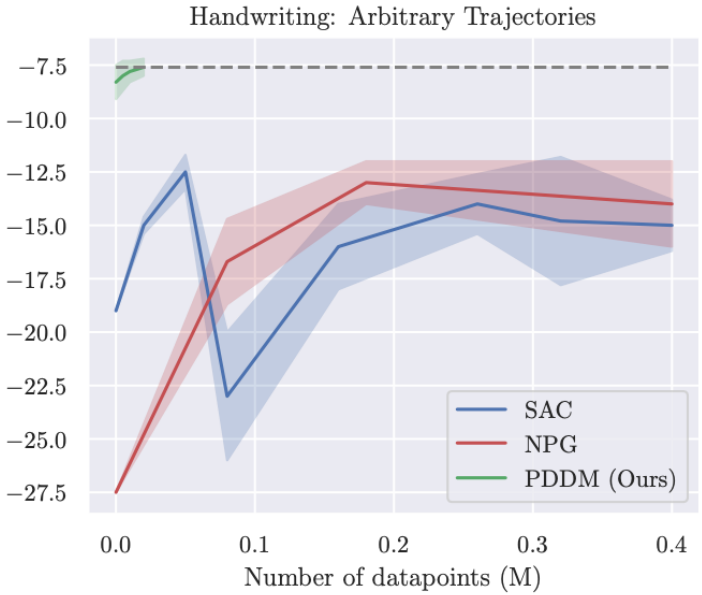
Model based RL v5 → From MPPI to MCTS

Why should you care?

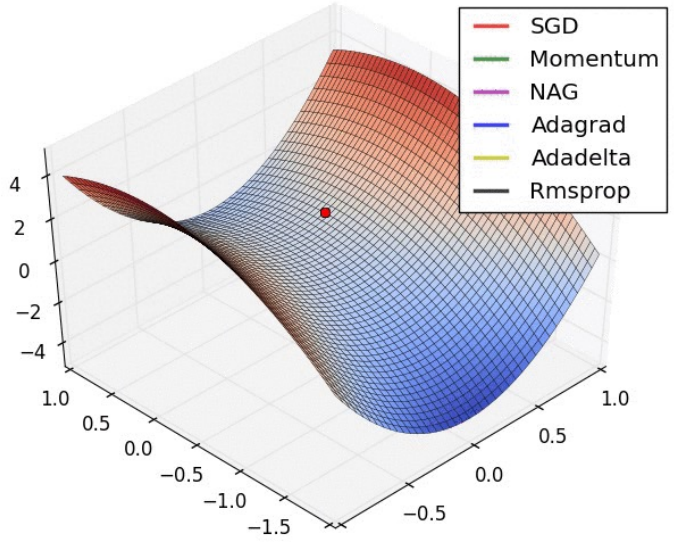
Model based RL **may be** a much more practical path to real world robotics

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