Model-Based Deep Reinforcement Learning

Joseph Marino

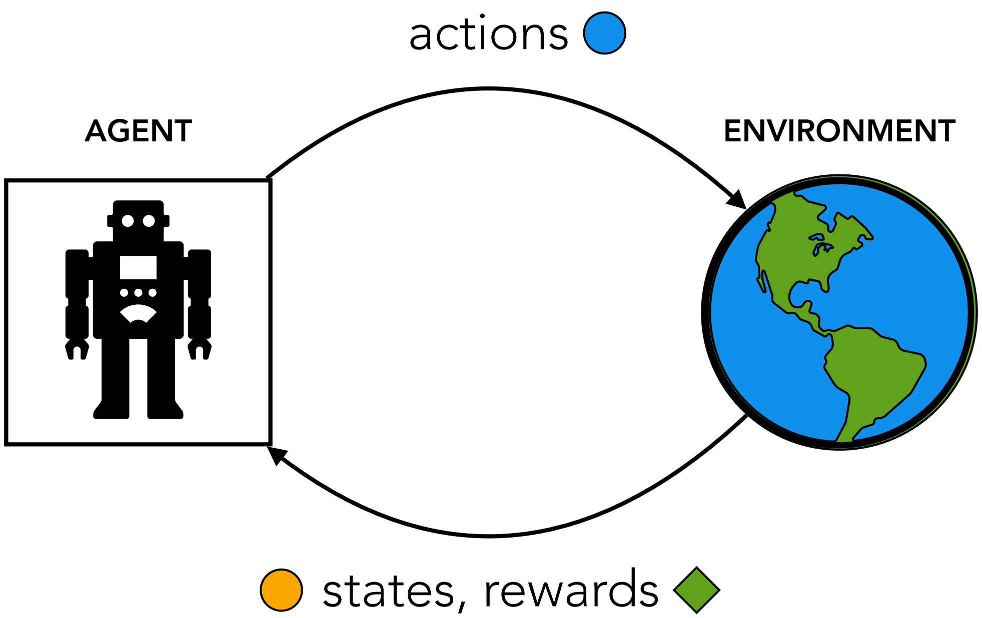
May 11, 2021

Caltech

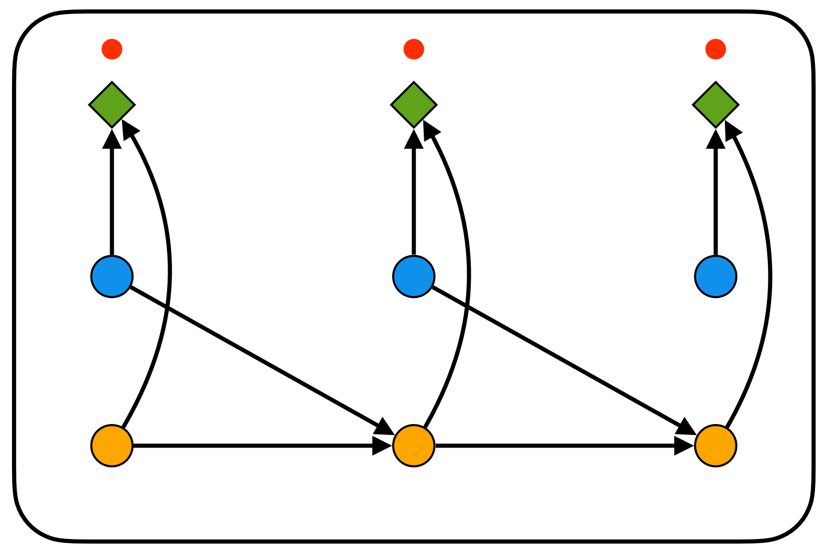
REINFORCEMENT LEARNING

Reinforcement Learning

sequential decision making
by maximizing expected future reward



maximize rewards w.r.t. actions



maximize $\sum_t \bullet$ w.r.t. $\bullet, \bullet, \ldots$

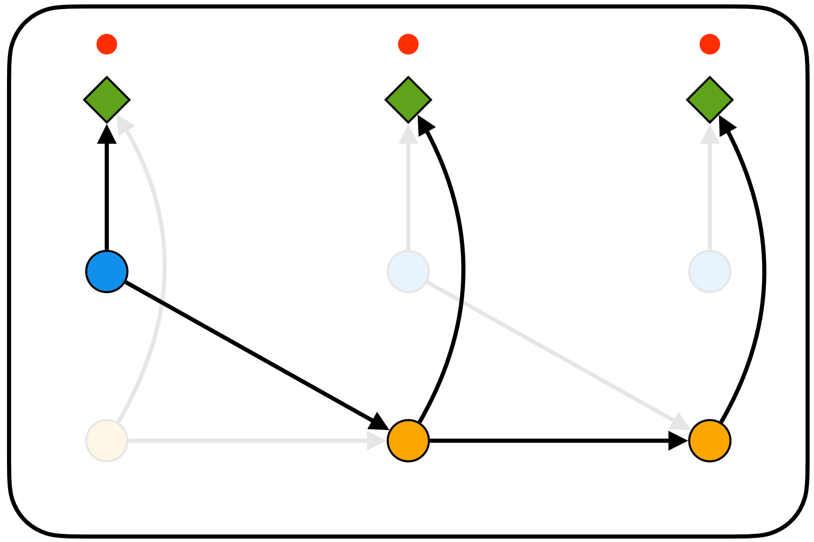
Reinforcement Learning

sequential decision making
by maximizing expected future reward

requires some way of estimating or evaluating future outcomes

- environment itself
- simulator
- learned value function
- **learned simulator, i.e. a model**

maximize rewards w.r.t. actions



maximize $\sum_t$ ● w.r.t. ●, ●, …

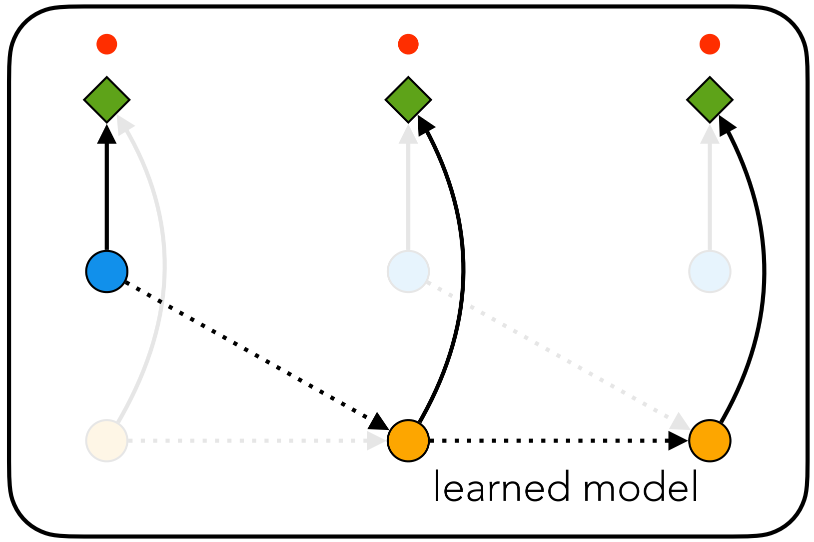
Reinforcement Learning

sequential decision making
by maximizing expected future reward

requires some way of estimating or evaluating future outcomes

- environment itself
- simulator
- learned value function
- **learned simulator, i.e. a model**

maximize rewards w.r.t. actions



maximize $\sum_t$ ● w.r.t. ●, ●, …

Reinforcement Learning

a model of what?

proprioception/kinematics

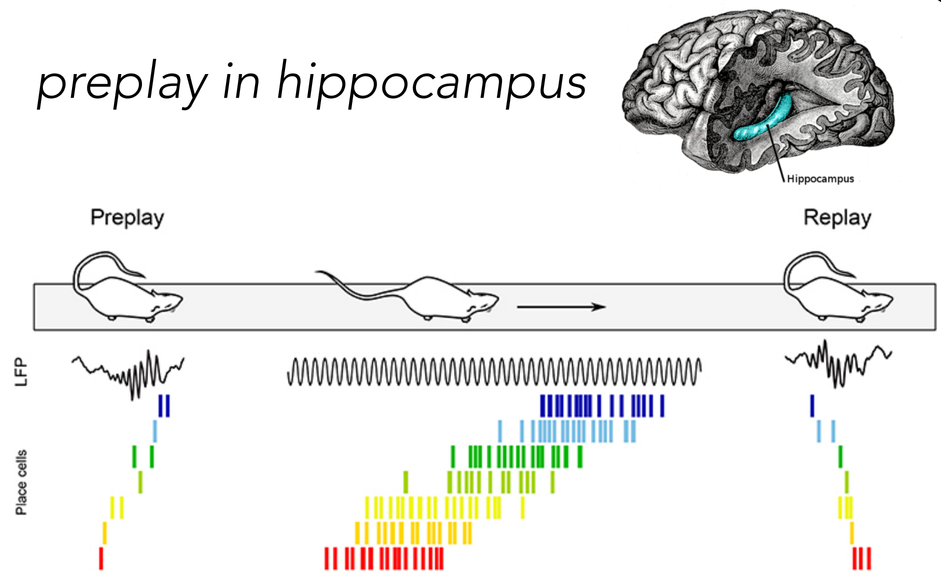
object manipulation

travel

can be anything, as long as you define the state / action spaces

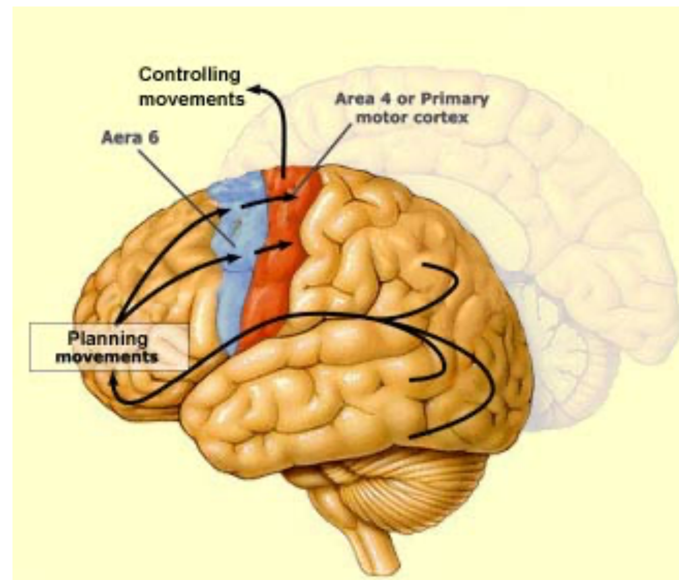
Reinforcement Learning

preplay in hippocampus



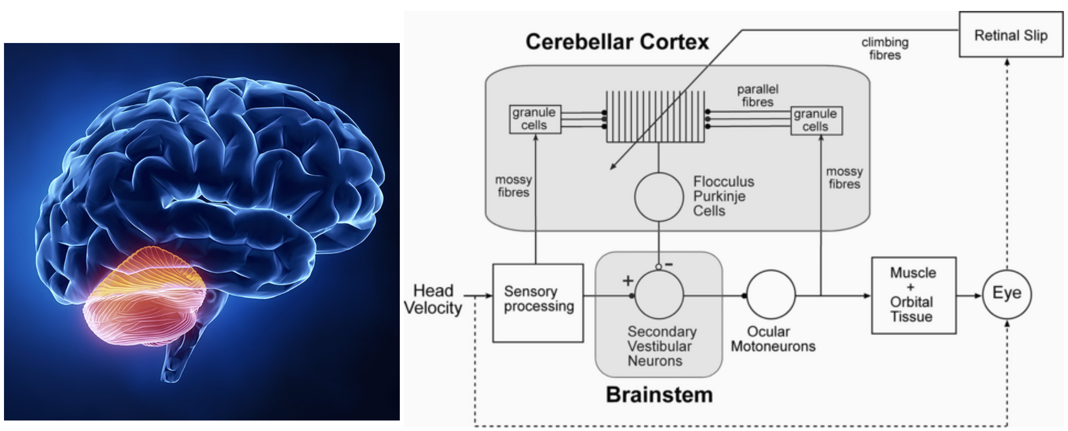
Drieu & Zugaro, 2019

goals & plans in prefrontal cortex



e.g., Badre 2020

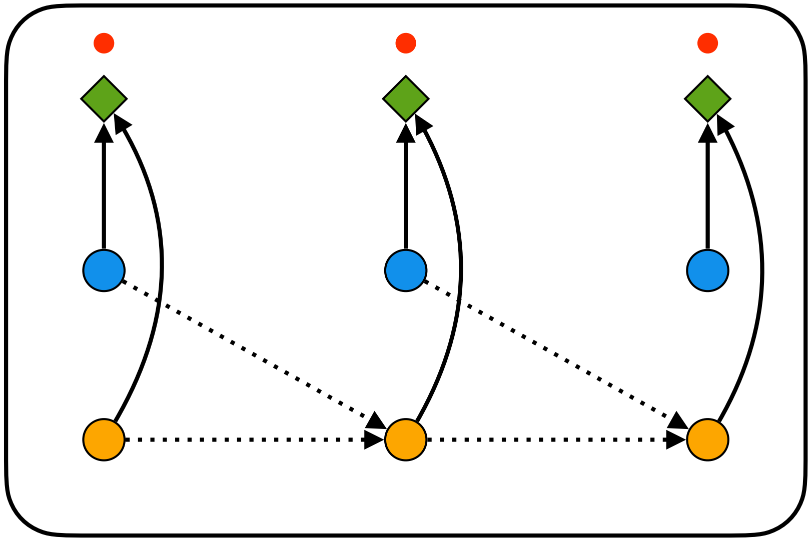
low-level models in cerebellum

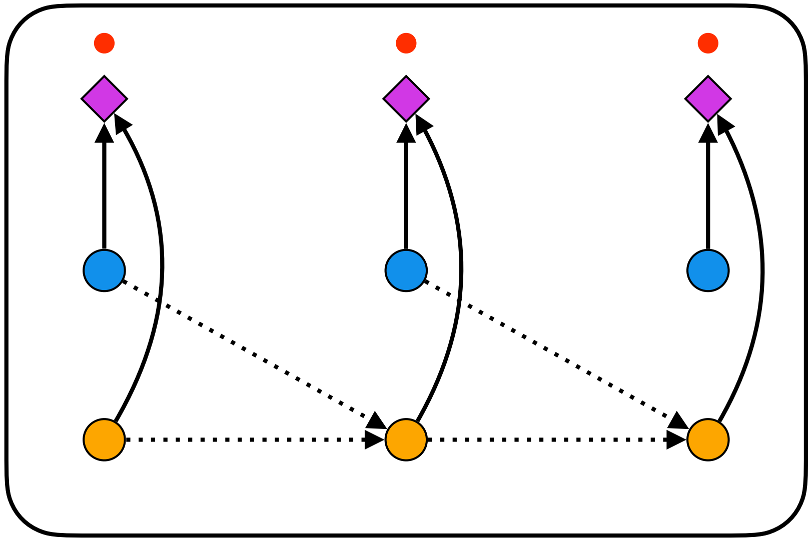


Porrill & Dean, 2007

Motivation

models are general (not task specific)





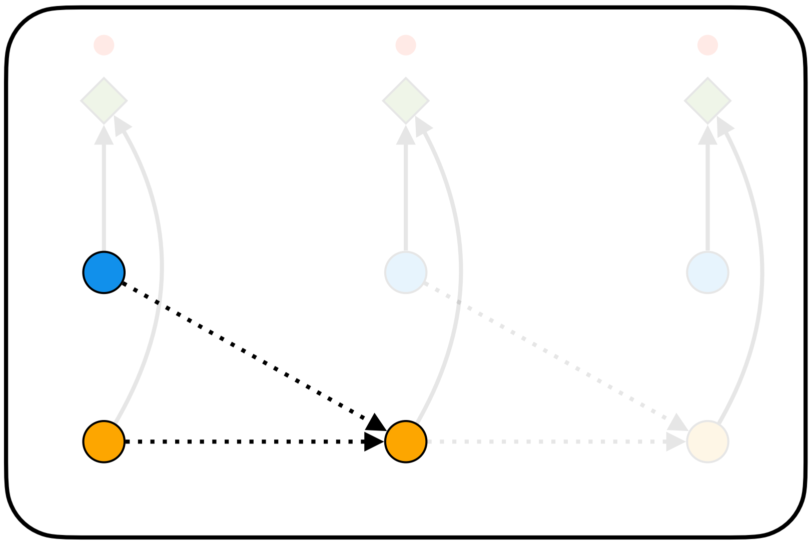
reward for task A

reward for task B

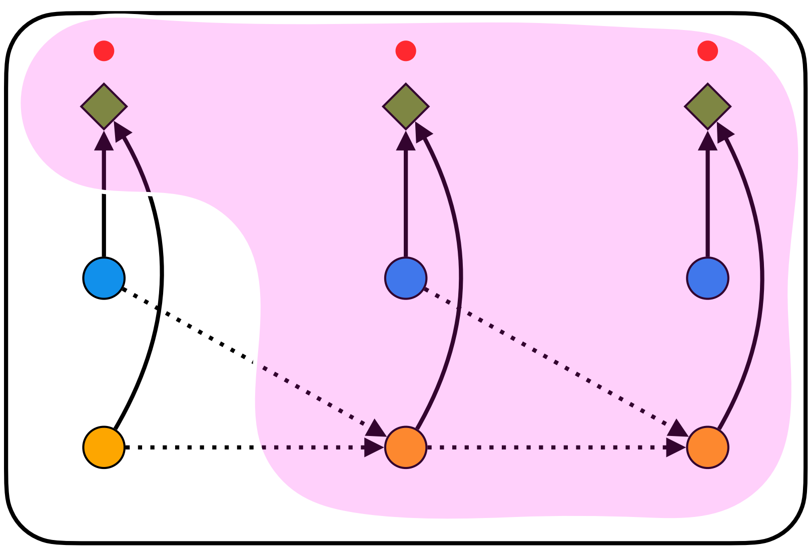
if the reward is known, can use the ***same*** model!

Motivation

models can be easier to learn



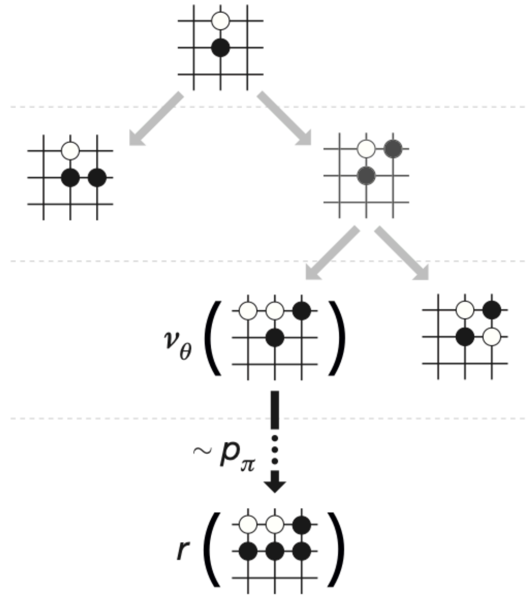
can just estimate 1-step dynamics



…whereas learning a value function
requires an expectation over future steps

Motivation

may be better suited for certain environments



e.g., board games have simple dynamics,
but a large number of possible outcomes

Silver et al., 2016

Motivation

easier to incorporate expert knowledge

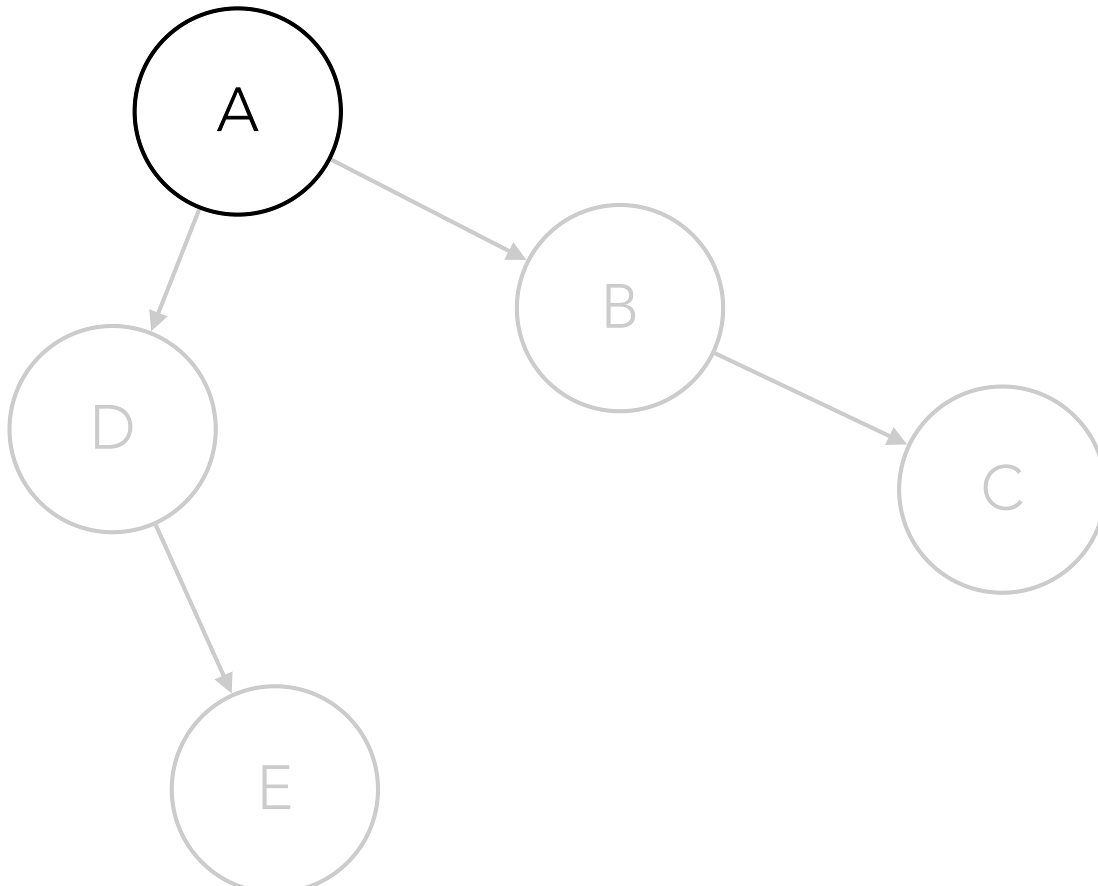
Neural Lander

model dynamics with physics,
and only use learning for cases that are difficult to model (e.g., near the ground)

Shi et al., 2019

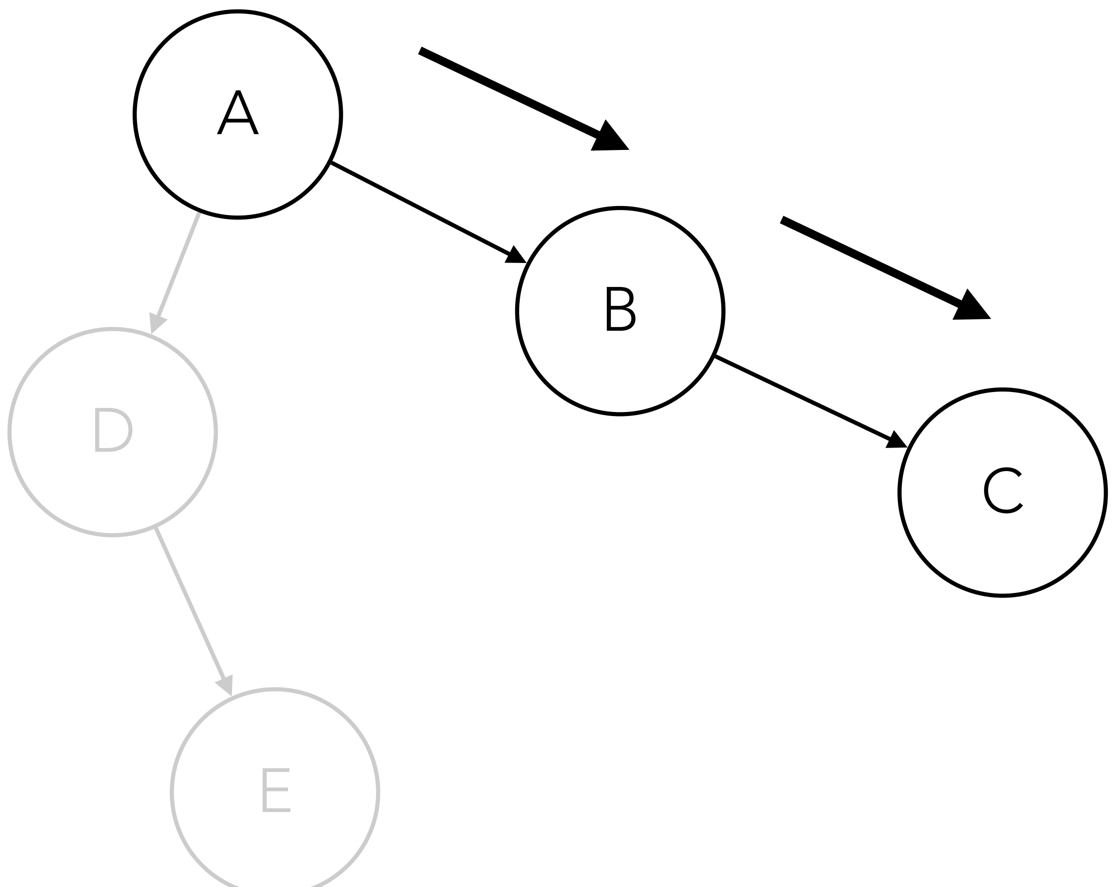
Motivation

can use new information more immediately



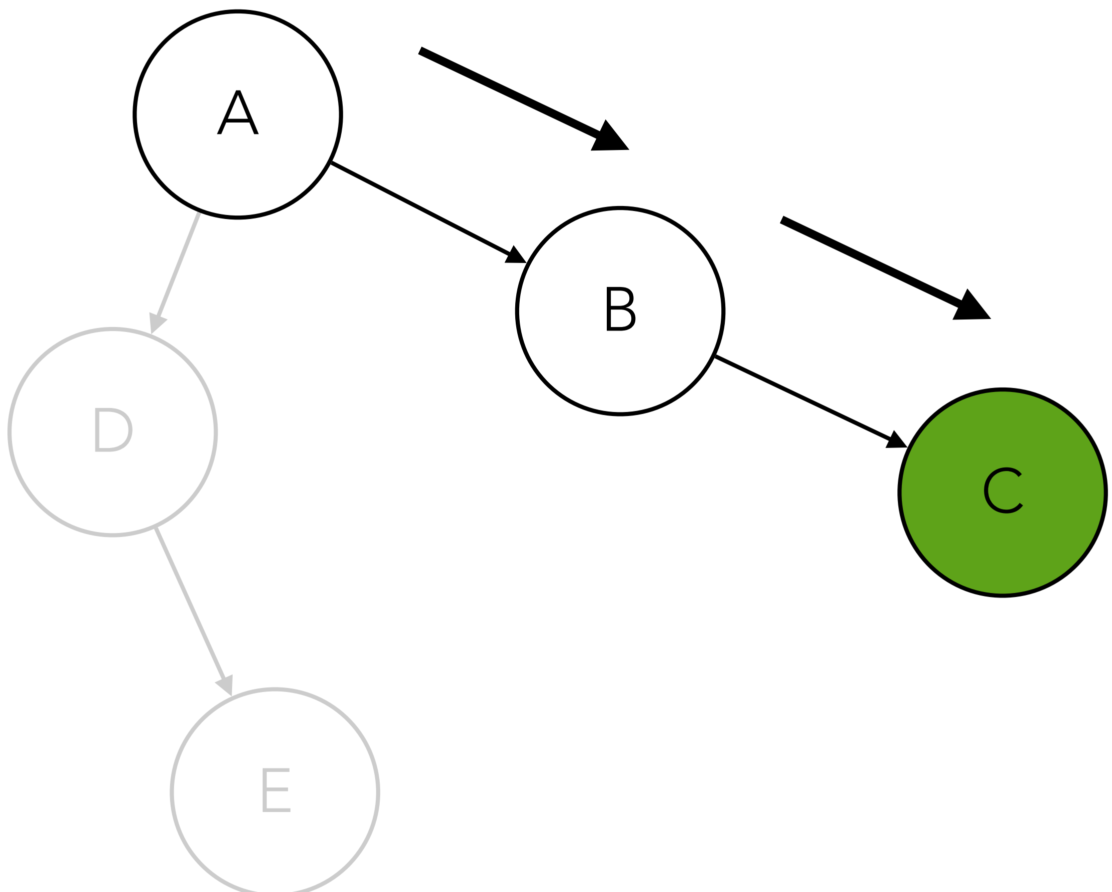
Motivation

can use new information more immediately



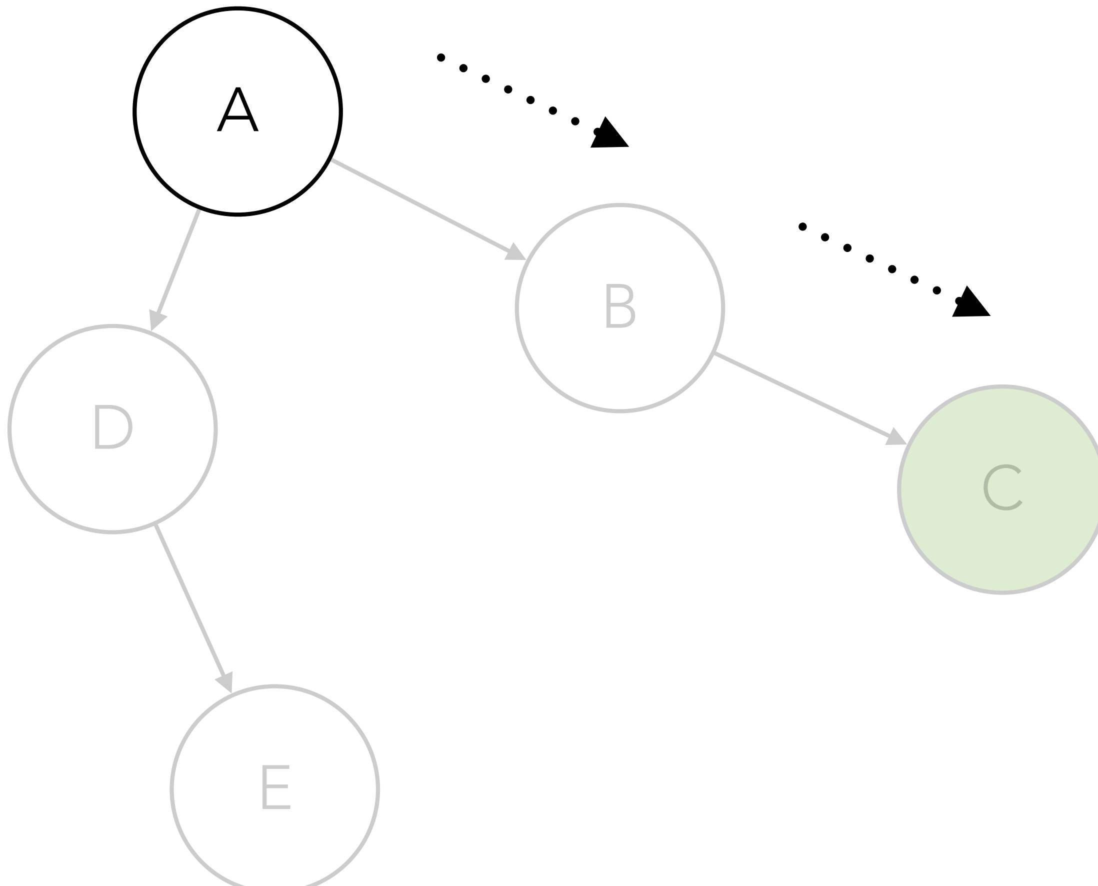
Motivation

can use new information more immediately



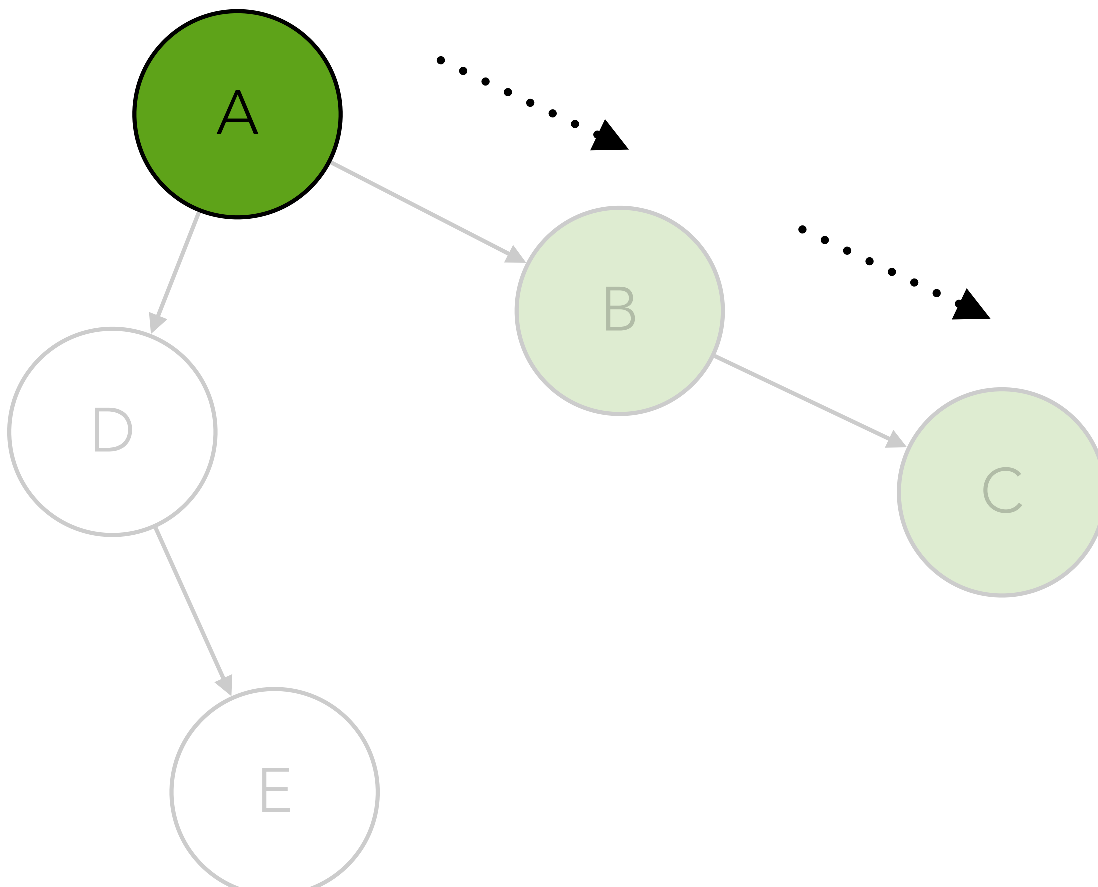
Motivation

can use new information more immediately



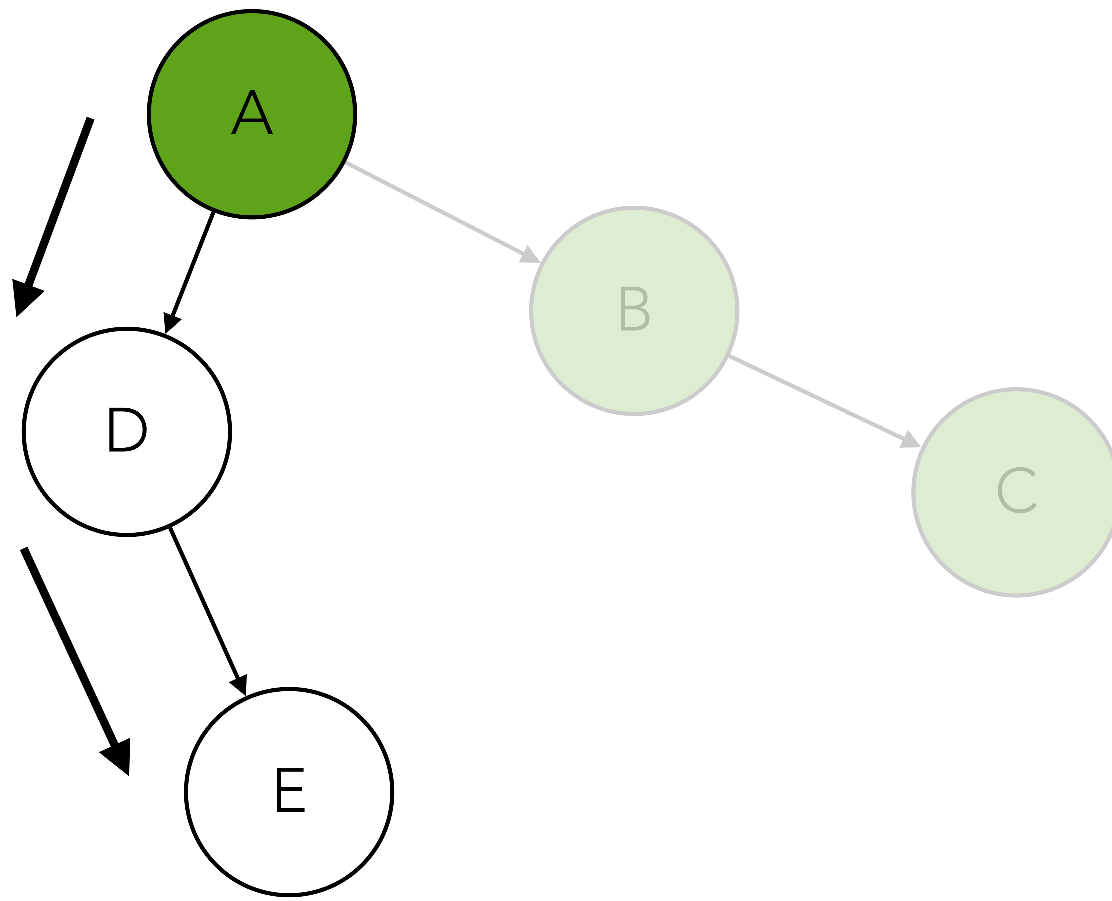
Motivation

can use new information more immediately



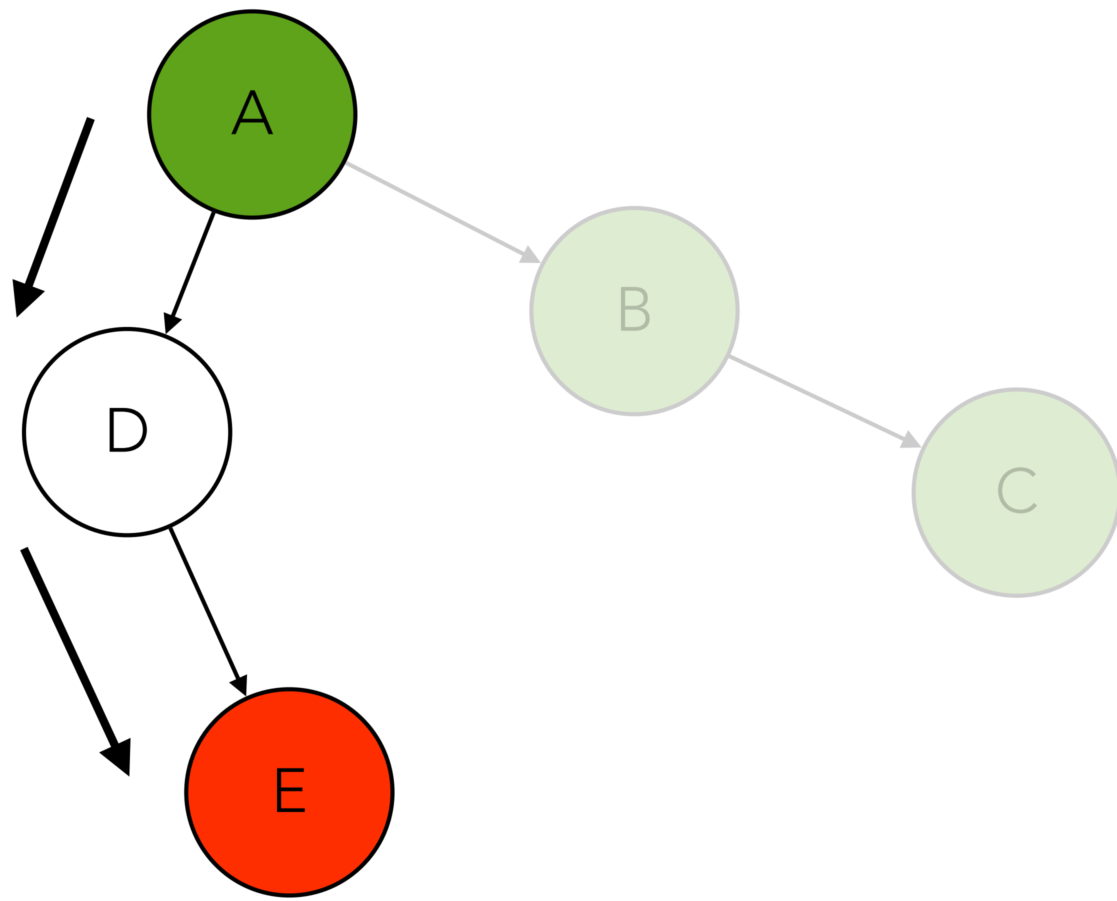
Motivation

can use new information more immediately



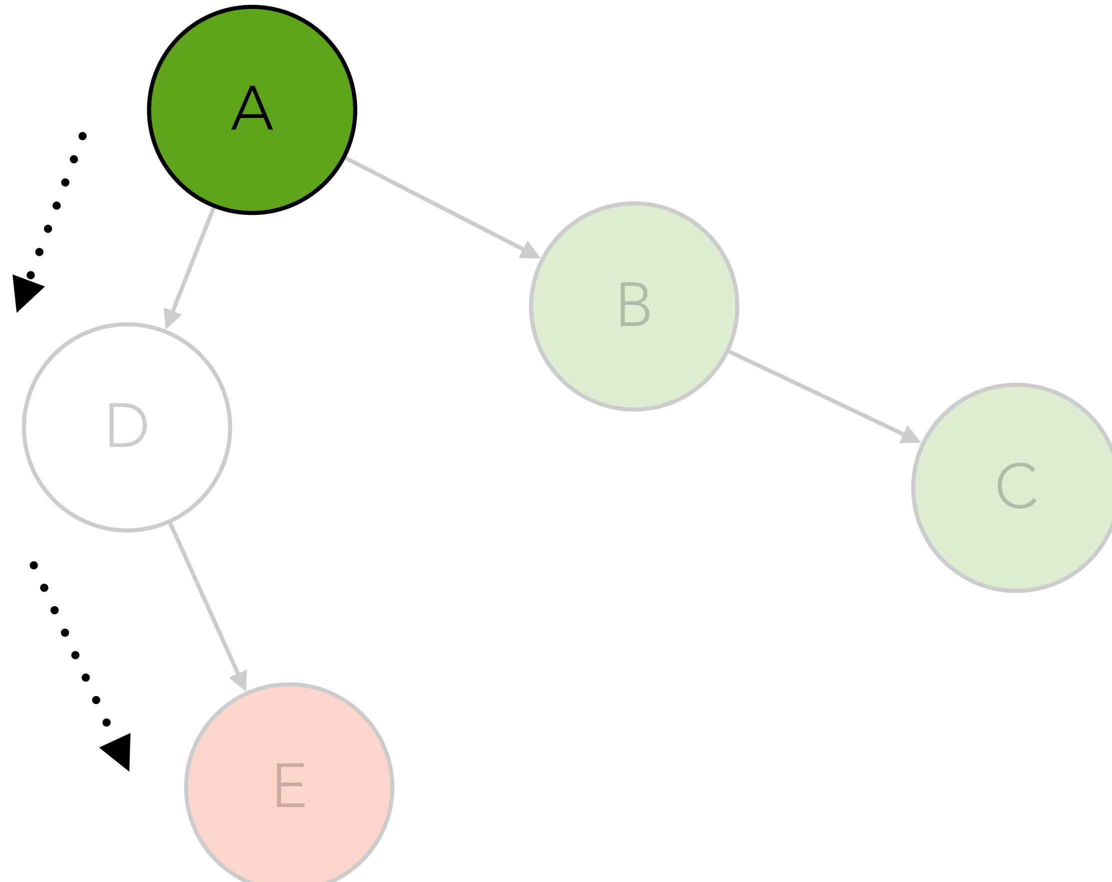
Motivation

can use new information more immediately



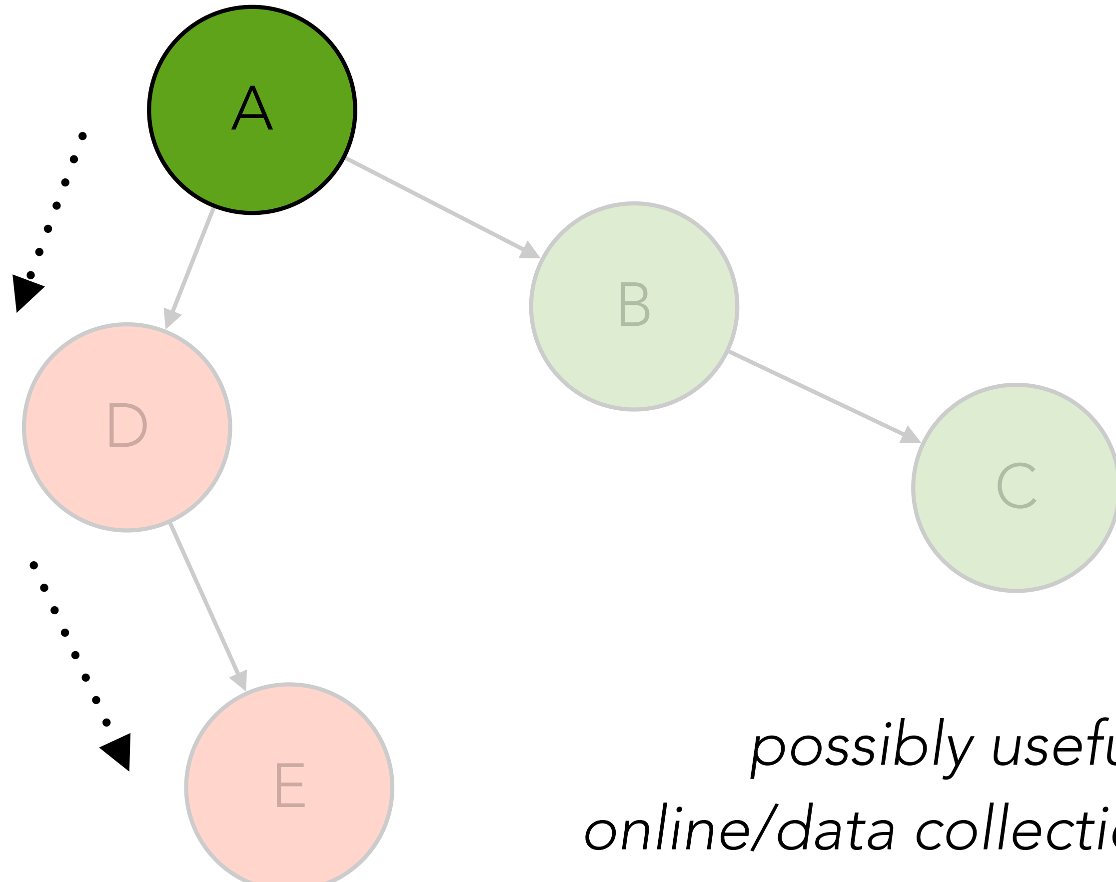
Motivation

can use new information more immediately



Motivation

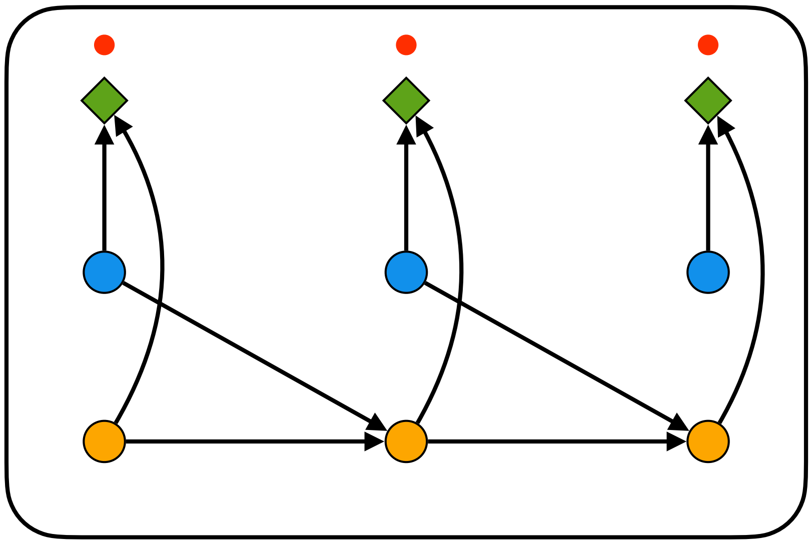
can use new information more immediately



possibly useful in
online/data collection setting

MODEL-BASED REINFORCEMENT LEARNING

Model-Based RL



$r(\mathbf{s}_t, \mathbf{a}_t) \in \mathbb{R}$

$\pi(\mathbf{a}_t|\cdot)$

$p_{\text{env}}(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})$

$$p_{\text{env}}(\mathbf{s}_{1:T}|\mathbf{a}_{1:T}) = \prod_t p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

a model is an approximation of $p_{\text{env}}(\mathbf{s}_{1:T}|\mathbf{a}_{1:T})$

and maybe $r(\mathbf{s}_t, \mathbf{a}_t)$

we will refer to the model as $p_\theta(\mathbf{s}_{1:T}|\mathbf{a}_{1:T})$

Model-Class & Learning

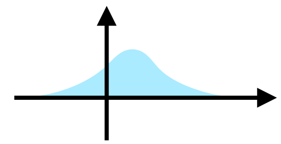
two main considerations for a generative model

- distribution
  - family & dependency structure
  - function(s)

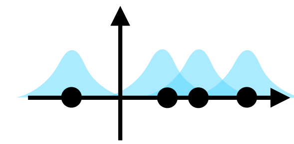
- learning objective
  - typically cross-entropy

family & dependency structure

e.g.,



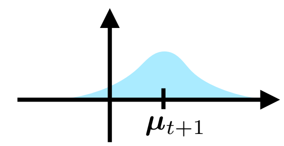
parametric



non-parametric

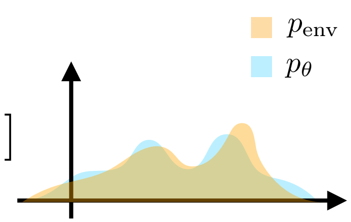
function(s)

e.g., $\boldsymbol{\mu}_{t+1} = \mathrm{NN}_\theta(\mathbf{s}_t, \mathbf{a}_t)$



learning objective

$$\max_\theta \mathbb{E}_{p_{\text{env}}(\mathbf{s}|\cdot)} \left[\log p_\theta(\mathbf{s}|\cdot)\right]$$



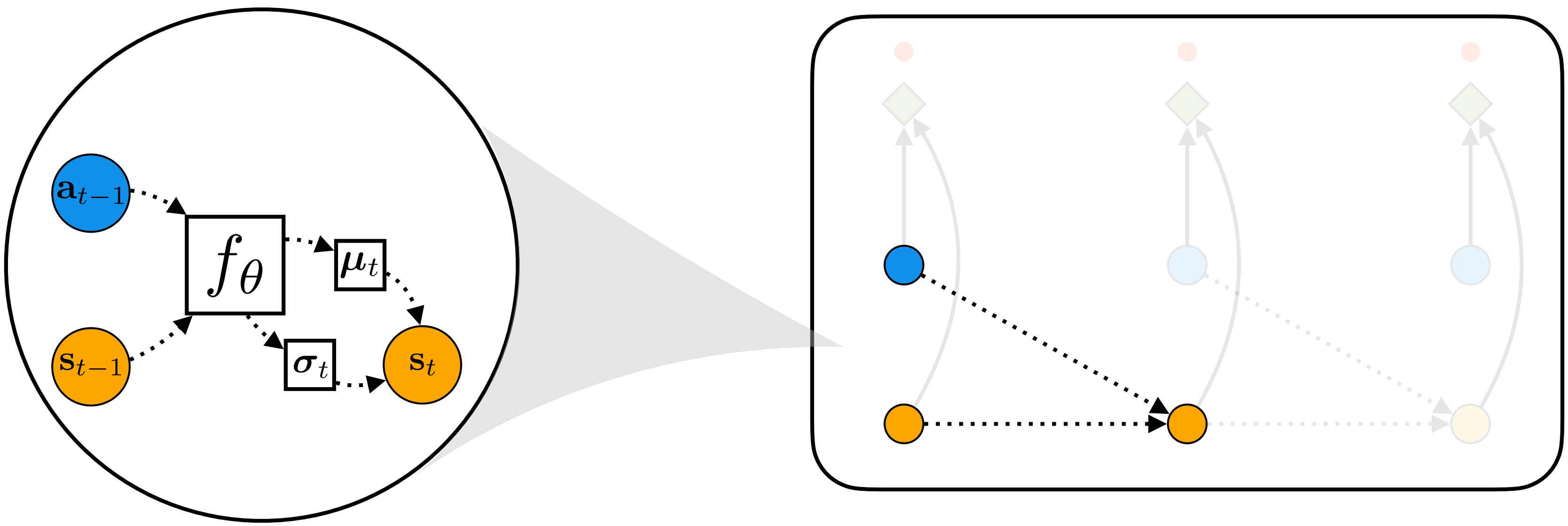
The 1-Step Model

factorize into a product of 1-step transition probabilities

$$p_\theta(\mathbf{s}_{1:T}|\mathbf{a}_{1:T}) = \prod_t p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})$$

parameterize each 1-step transition with a simple distribution

$$p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}) = \mathcal{N}(\mathbf{s}_t; \boldsymbol{\mu}_t, \mathrm{diag}(\boldsymbol{\sigma}_t^2))$$



Policy Optimization

maximize expected sum of rewards w.r.t. policy

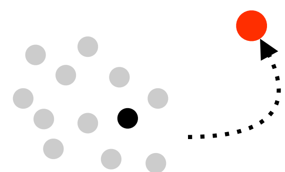
(from model)

OPTIMIZERS:

gradient-free

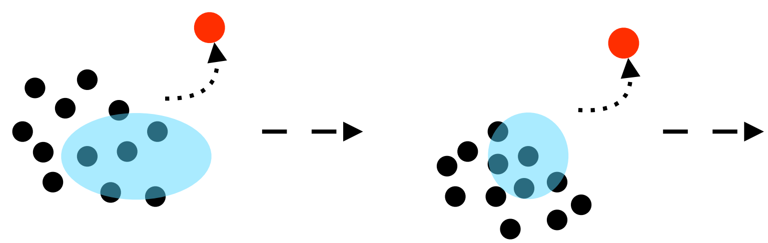
Random Shooting

choose best random sample



Cross Entropy Method (CEM)

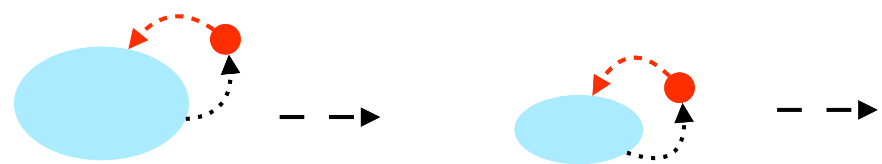
iteratively re-fit policy to best samples



gradient-based

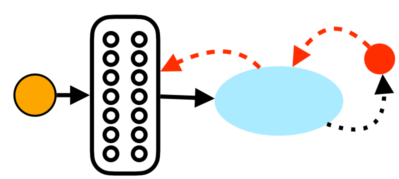
Gradient Ascent / Descent

optimize policy directly



Policy Network

optimize parameters of a network that outputs policy

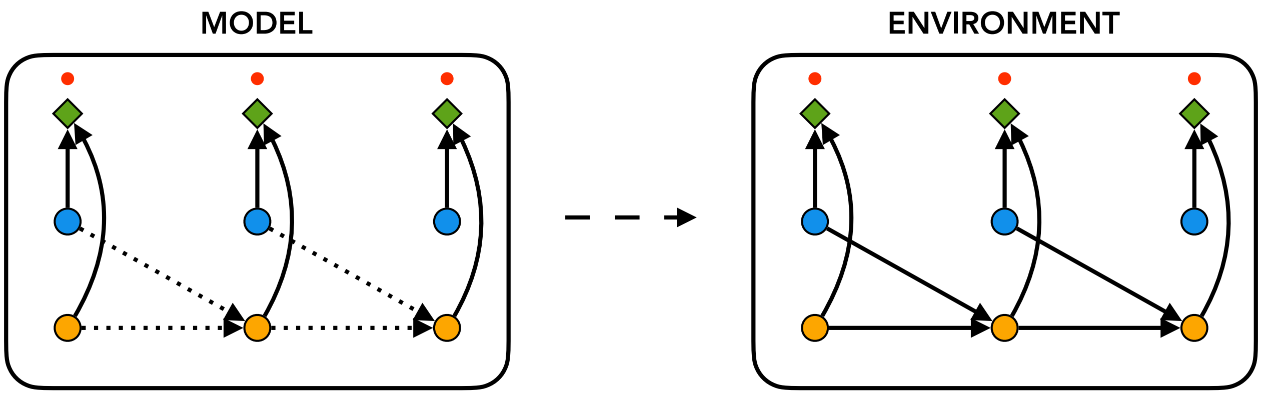


. . .

Open vs. Closed Loop Optimization

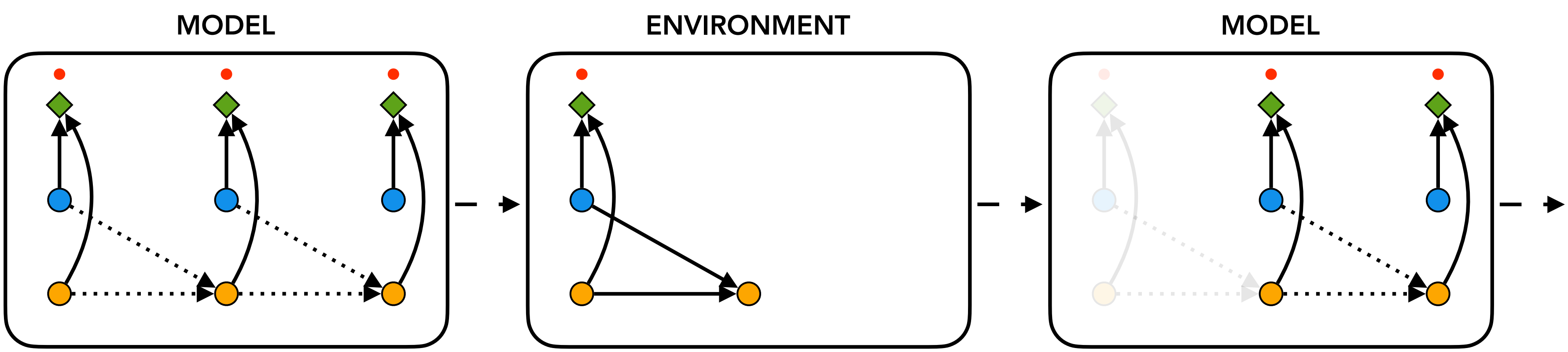
open loop

plan once, then execute



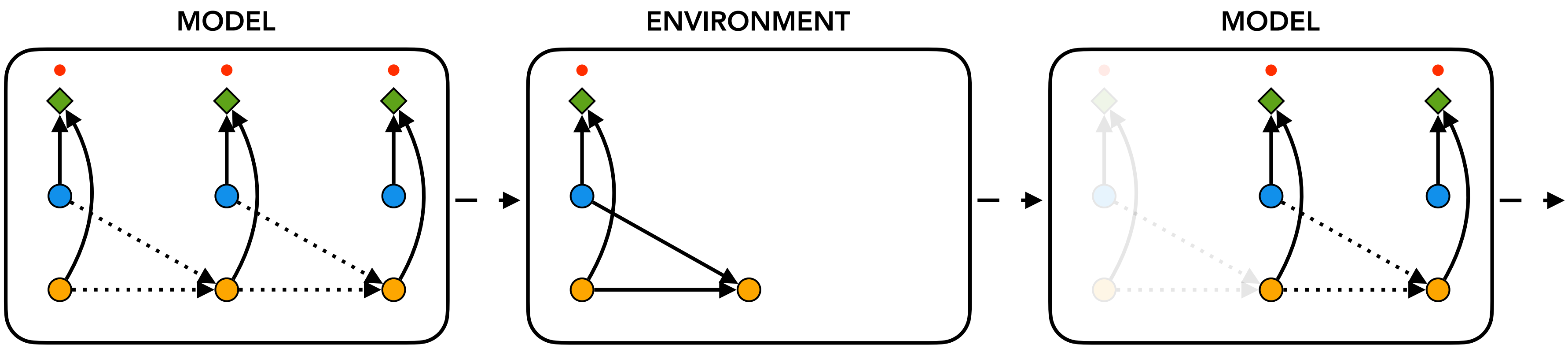
closed loop (model predictive control)

re-plan / execute at each time step

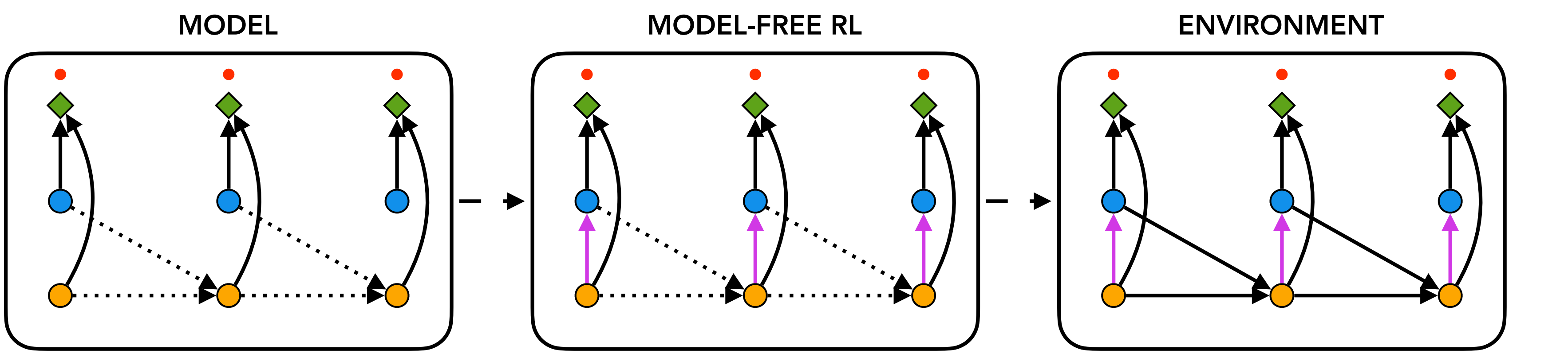


Online vs. Offline

Planning: *model as* ***online*** *simulator*



DYNA: *model as* ***offline*** *simulator* (Sutton, 1991)



can use any model-free RL algorithm

DEEP MODEL-BASED REINFORCEMENT LEARNING

Deep Model-Based RL

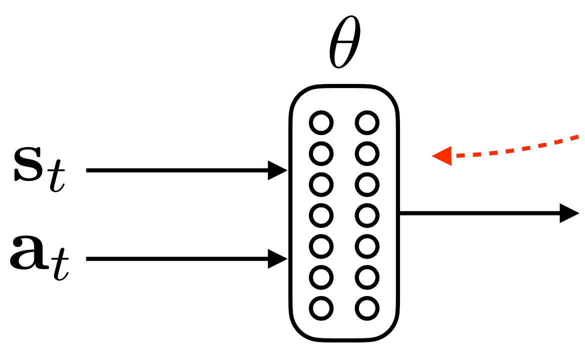
parameterize the model using a ***deep* neural network**

typical example: 1-step model

∇_θ $\mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\text{env}}} \left[\log p_\theta(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t) \right]$

maximum log-likelihood

e.g., $||\mathbf{s}_{t+1} - \boldsymbol{\mu}_\theta||_2^2$

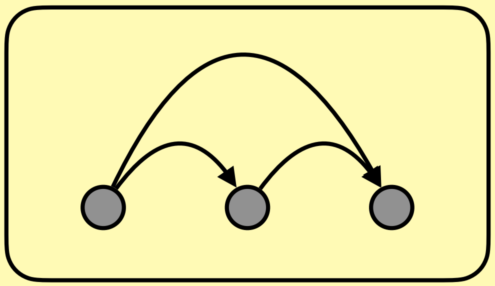


$$\underbrace{p_\theta(\mathbf{s}_{t+t} | \mathbf{s}_t, \mathbf{a}_t)}$$

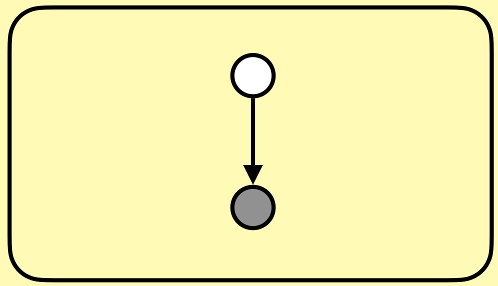
e.g., $\mathcal{N}(\mathbf{s}_{t+1}; \boldsymbol{\mu}_\theta, \text{diag}(\boldsymbol{\sigma}_\theta^2))$

Generative Modeling

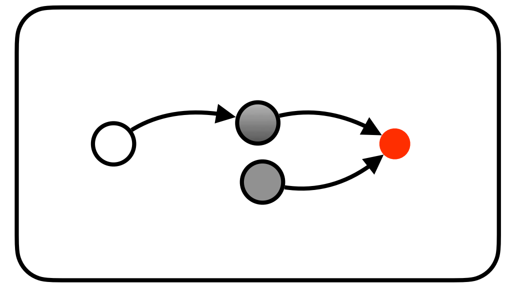
recent advances in generative models



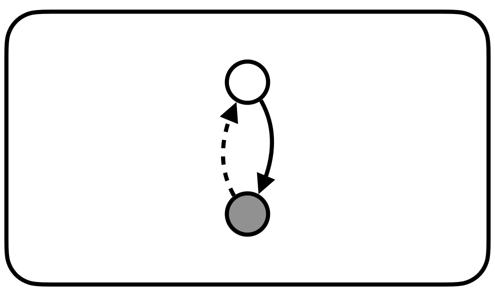
AUTOREGRESSIVE



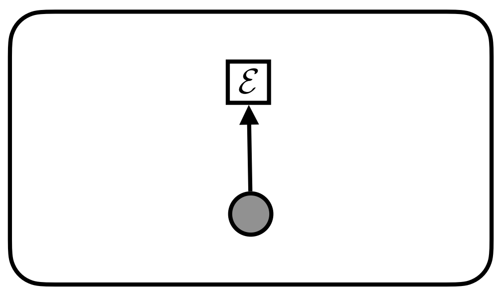
EXPLICIT
LATENT VARIABLE MODELS



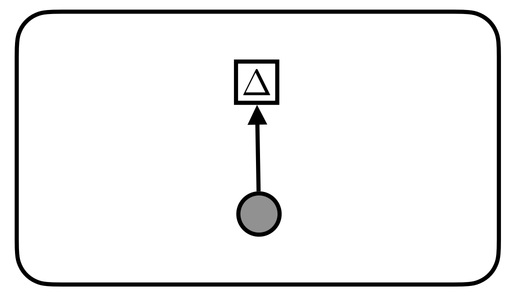
IMPLICIT
LATENT VARIABLE MODELS



FLOW-BASED MODELS



ENERGY-BASED MODELS

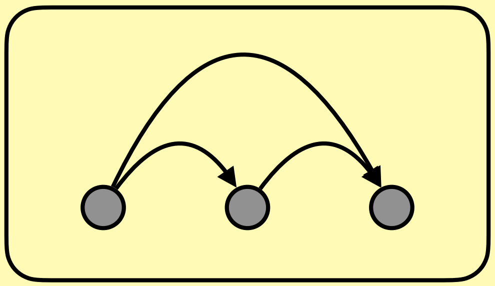


SCORE-BASED MODELS

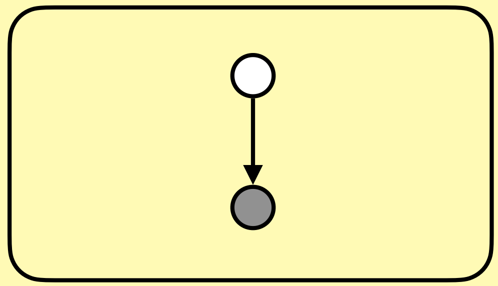
• • •

Generative Modeling

recent advances in generative models



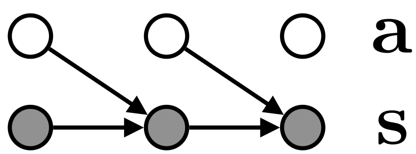
AUTOREGRESSIVE



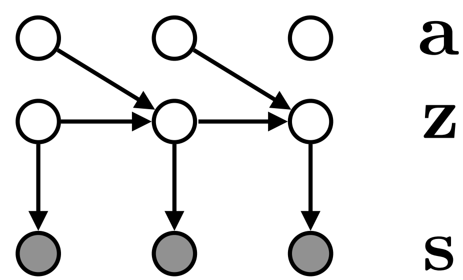
EXPLICIT
LATENT VARIABLE MODELS

in either case:

$$p_\theta(\mathbf{s}_{1:T}|\mathbf{a}_{1:T})$$



$$\prod_{t=1}^{T} p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1},\mathbf{a}_{t-1})$$



$$\prod_{t=1}^{T} p_\theta(\mathbf{s}_t|\mathbf{z}_t)p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{t-1})$$

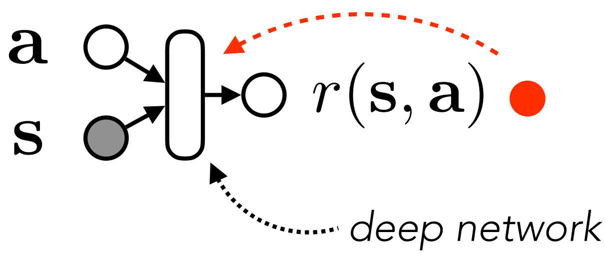
note: these are only possible examples

Modeling Reward

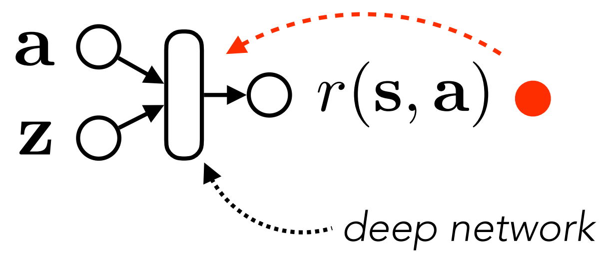
to estimate value, ***need some estimate of future reward/value***

can use the same maximum likelihood approach

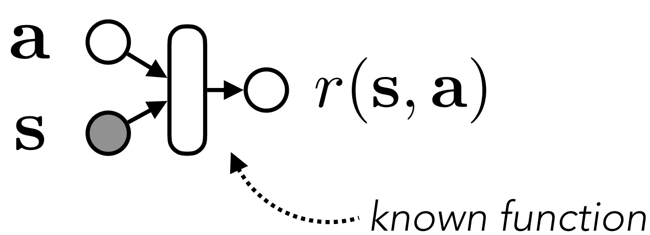
STANDARD SETTING



LATENT STATE SETTING



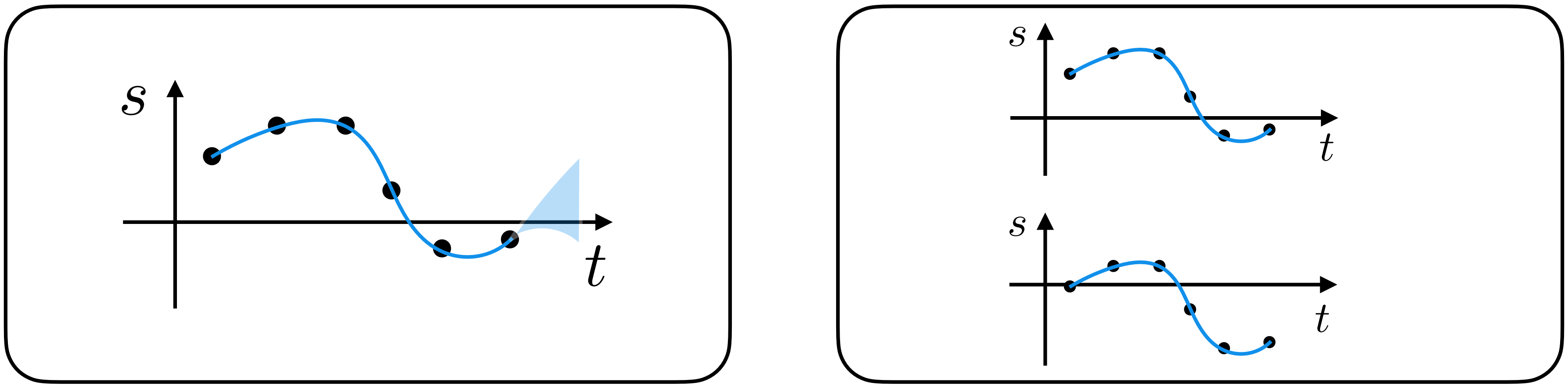
or assume we have access to the reward function



Practical Aspects of Deep MBRL

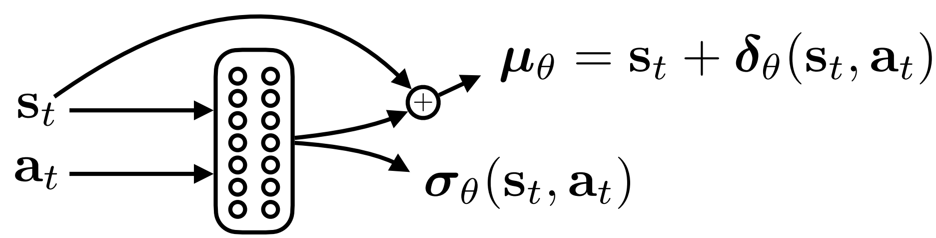
modeling state *changes*

states often change smoothly + dynamics generalize across states



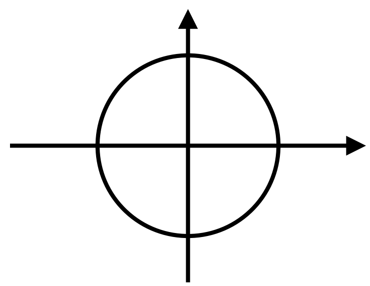
e.g., with $\mathcal{N}(\mathbf{s}_{t+1}; \boldsymbol{\mu}_\theta, \mathrm{diag}(\boldsymbol{\sigma}_\theta^2))$

estimate <u>*change*</u> in state:

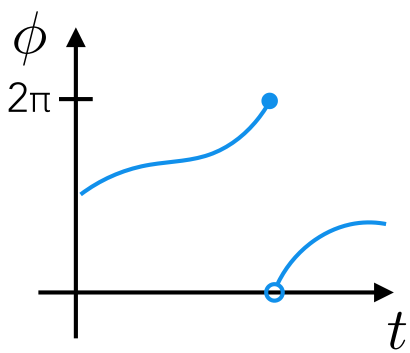


Practical Aspects of Deep MBRL

many robotic applications involve joint *angles*



restricted to 0 to 2π



results in discontinuities in state trajectories

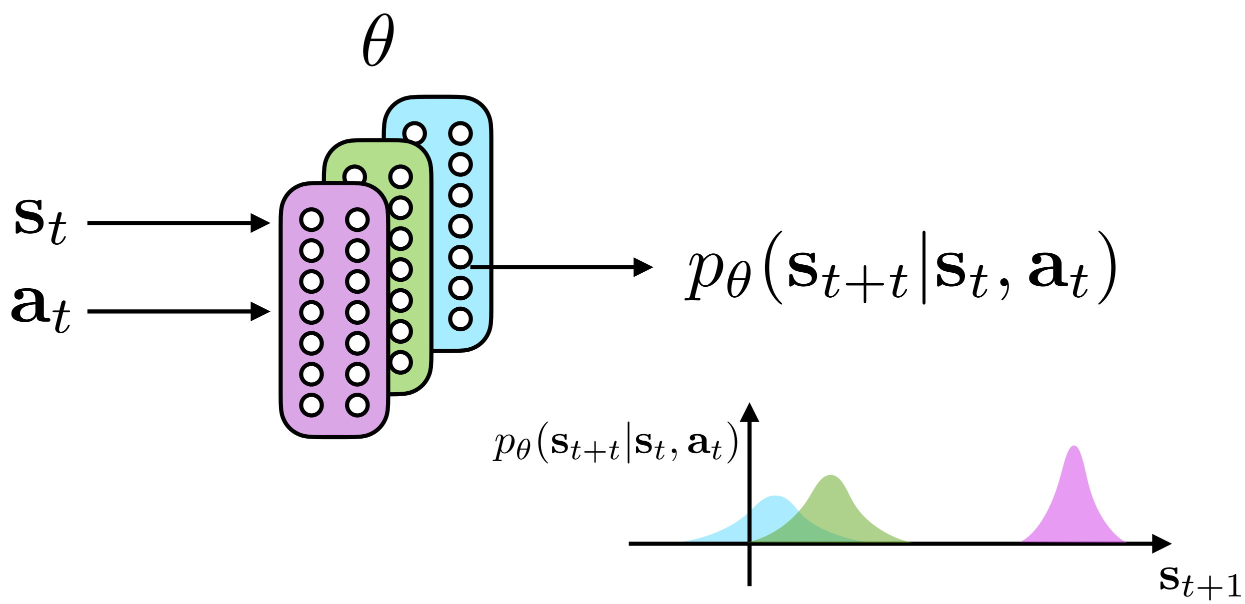
one approach: $\phi \rightarrow [\sin\phi, \cos\phi]$

see Chua et al., 2018)

Practical Aspects of Deep MBRL

a single distribution may not capture the
uncertainty in the model's estimate

ensemble of networks *(see Chua et al., 2018)*



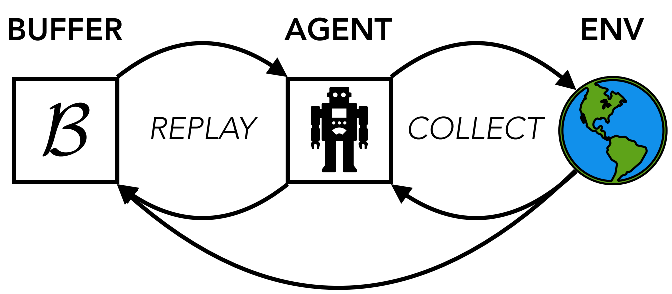
*epistemic* (knowledge) uncertainty may be multi-modal,
even if _*aleatoric*_ (inherent) uncertainty is not

Practical Aspects of Deep MBRL

issues with training on collected data

catastrophic forgetting

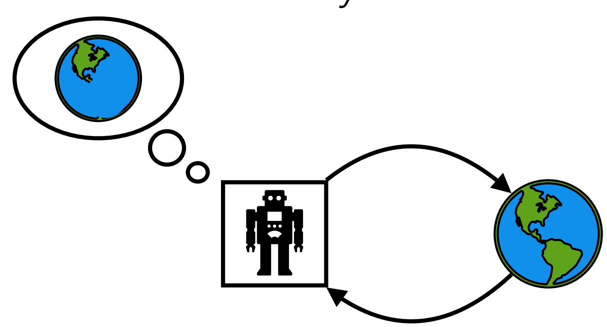
deep networks struggle with non-I.I.D. data,
forget *earlier examples when training online*



use a large replay buffer of recent samples

exploration / uncertainty

may avoid states with inaccurate
reward / dynamics estimates



initially collect large amount of random data
+ use action, value, and/or state exploration

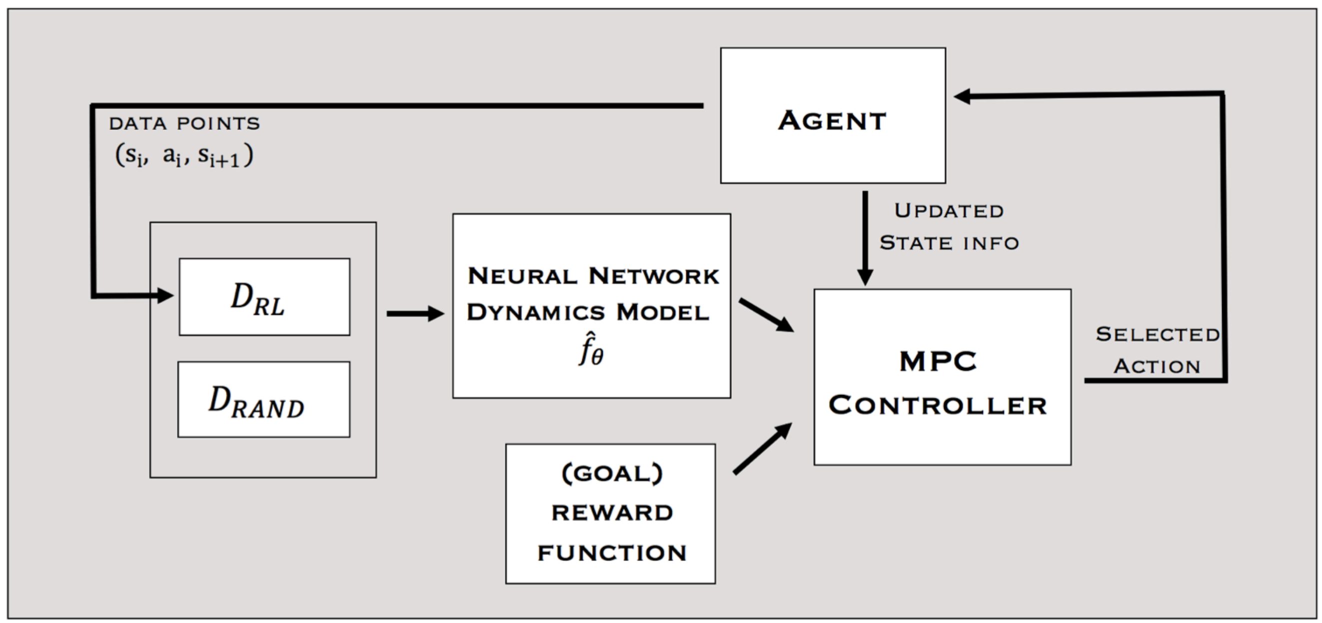
SURVEY

Nagabandi et al. 2017

single 1-step model

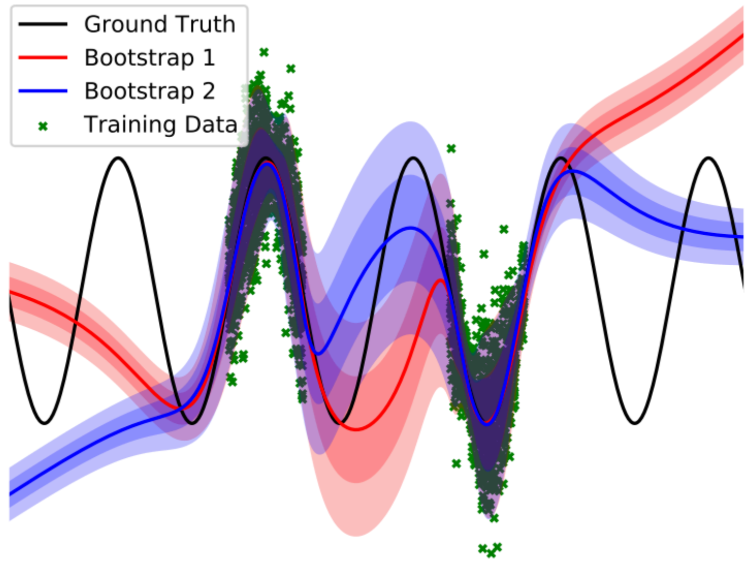
planning (MPC) with random shooting

perform imitation learning on planned actions to initialize model-free agent

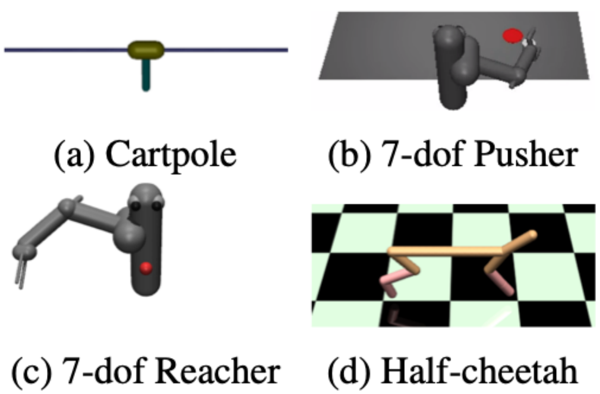


PETS (Probabilistic Ensembles with Trajectory Sampling)

uses ensembles (bootstraps) of 1-step models



planning + CEM + various sampling strategies



(a) Cartpole (b) 7-dof Pusher

(c) 7-dof Reacher (d) Half-cheetah

continuous control

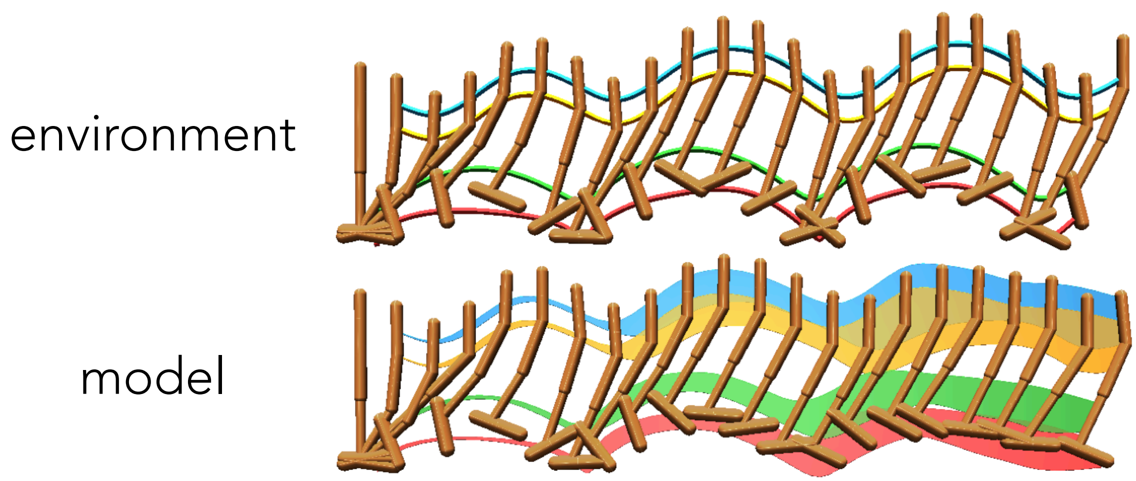
Chua et al., 2018

MBPO (Model-Based Policy Optimization)

Dyna-style training with model ensemble and (model-free) actor-critic setup

Very short rollouts for model-based value estimation

Continuous control



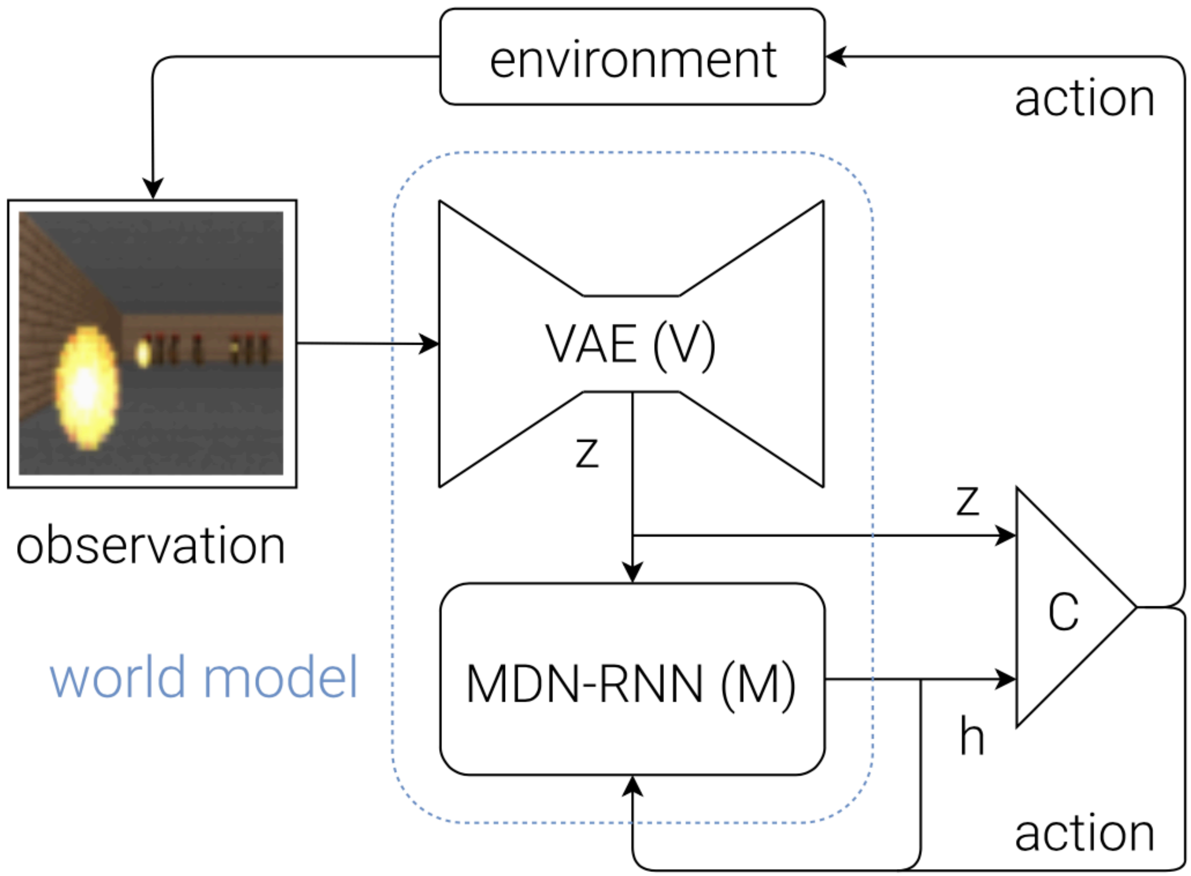
Janner et al., 2019

World Models

Dyna-style training with evolutionary policy

Uses a sequential latent variable model

Discrete actions

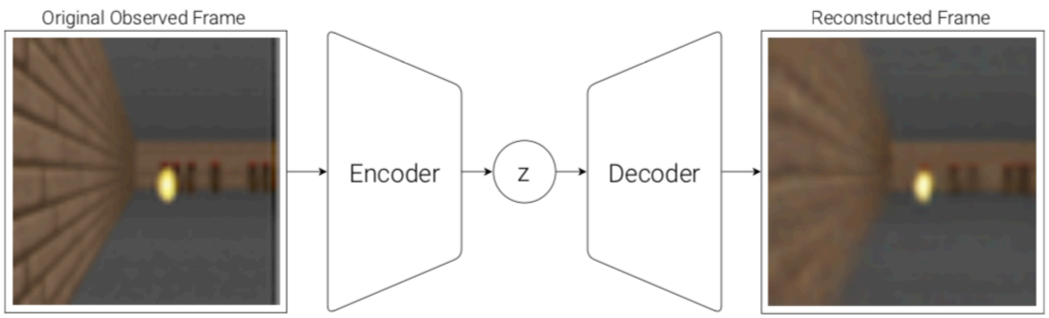


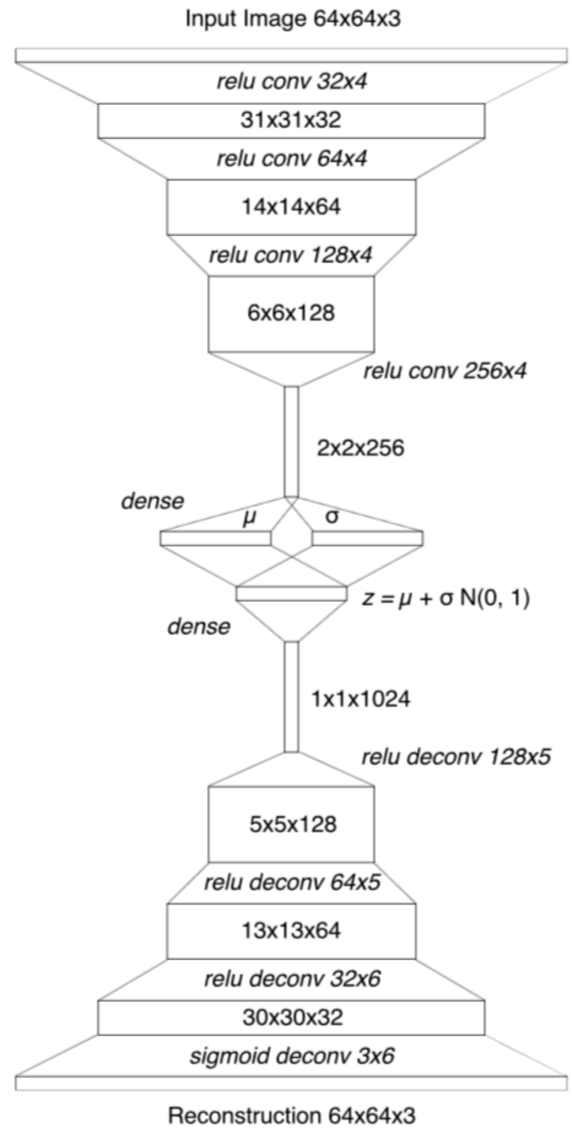
Ha & Schmidhuber, 2018

World Models

the model (vision):

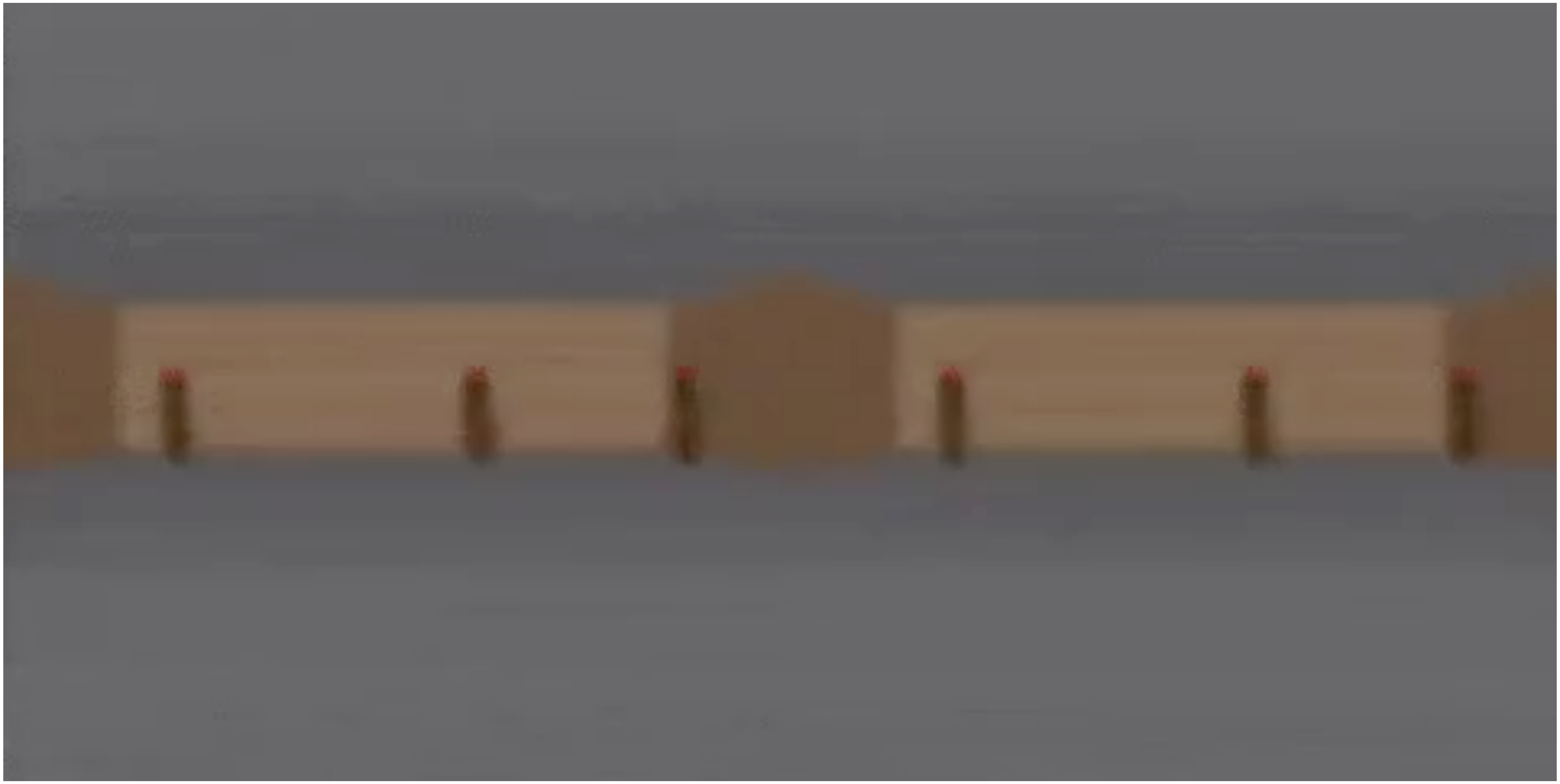
compress the observations





Ha & Schmidhuber, 2018

World Models



observations reconstructions

Ha & Schmidhuber, 2018

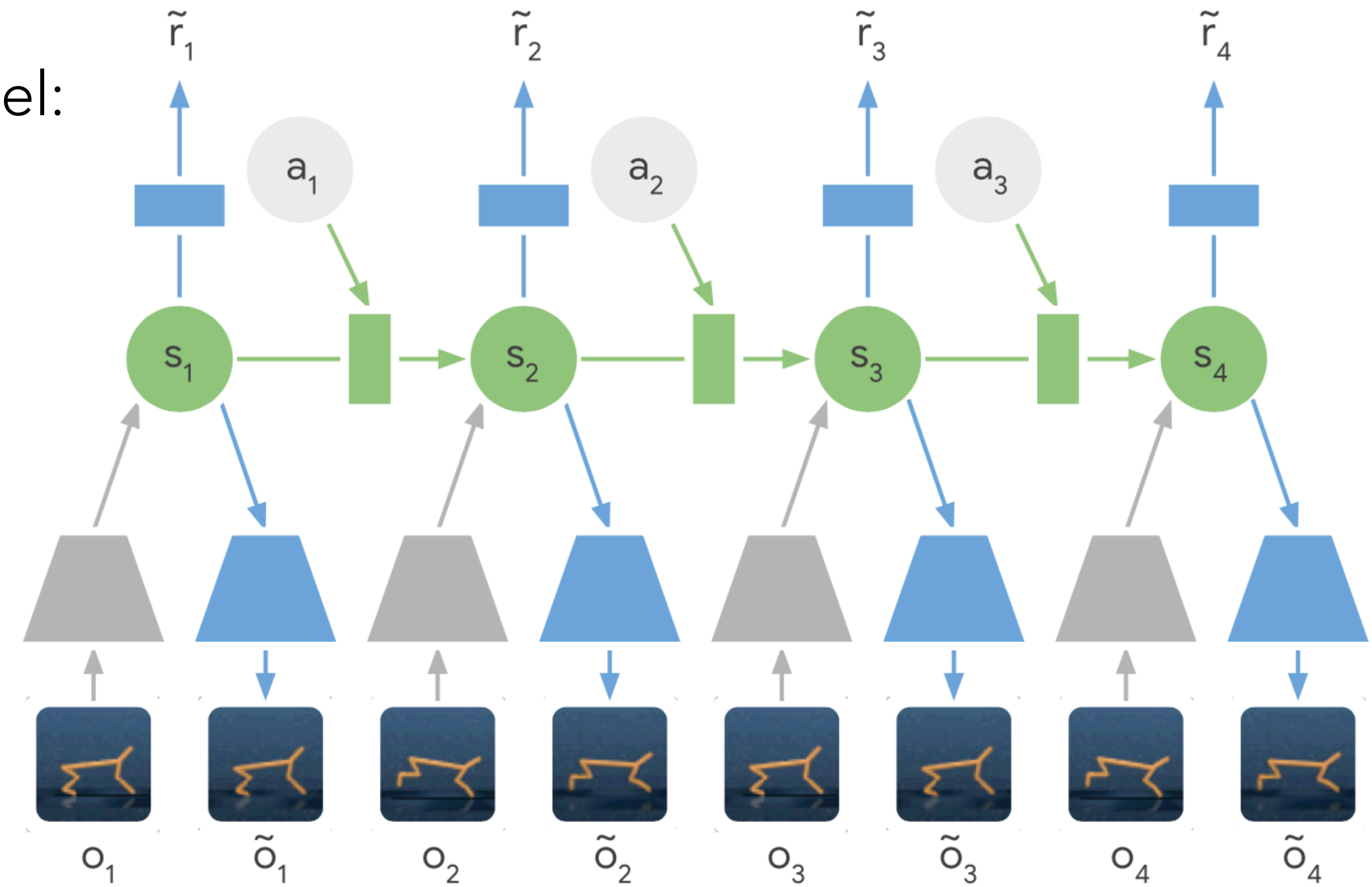
PlaNet

Similar model as World Models

Uses planning with CEM

Continuous control from visual inputs

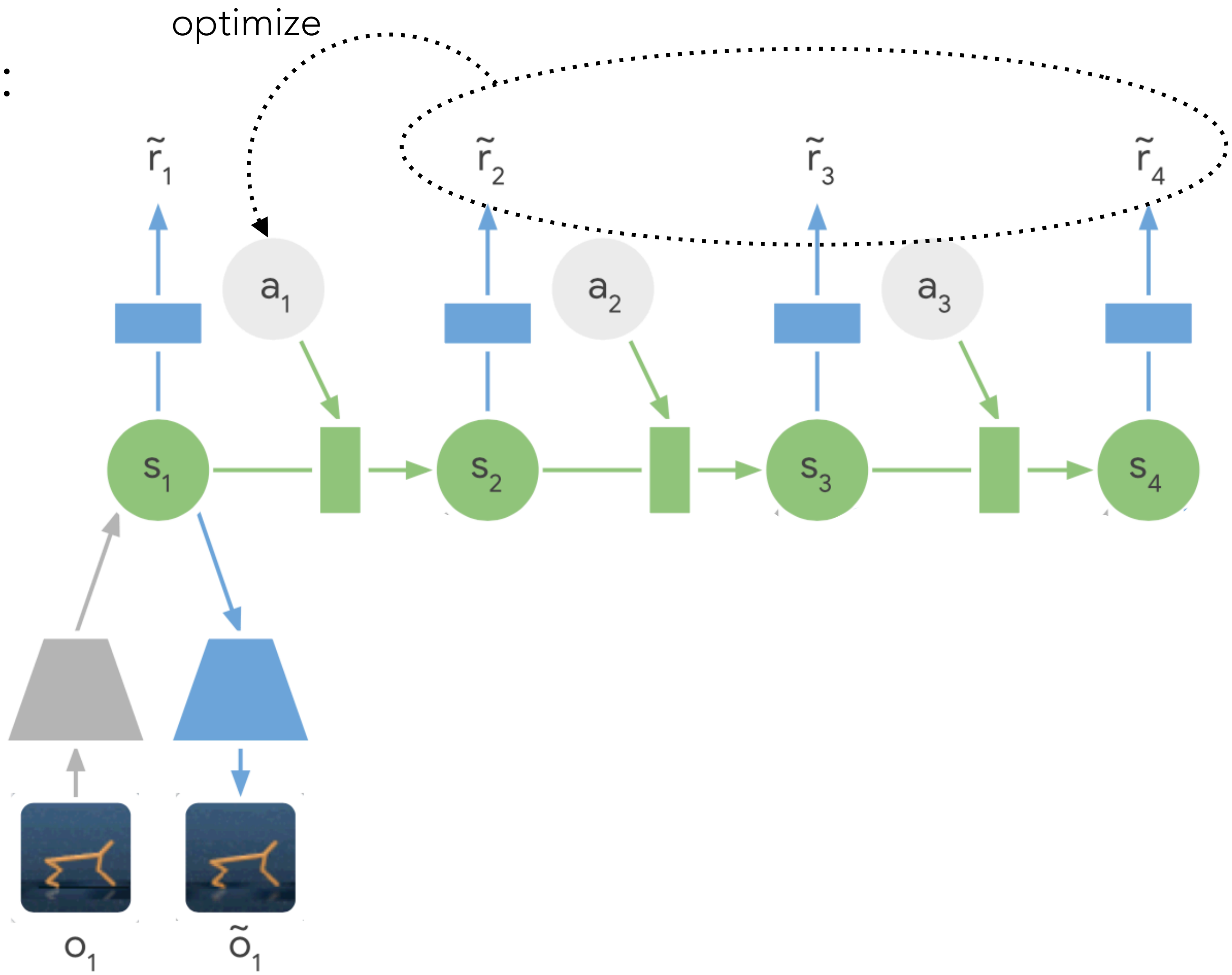
the model:



Hafner *et al.*, 2019

PlaNet

planning:

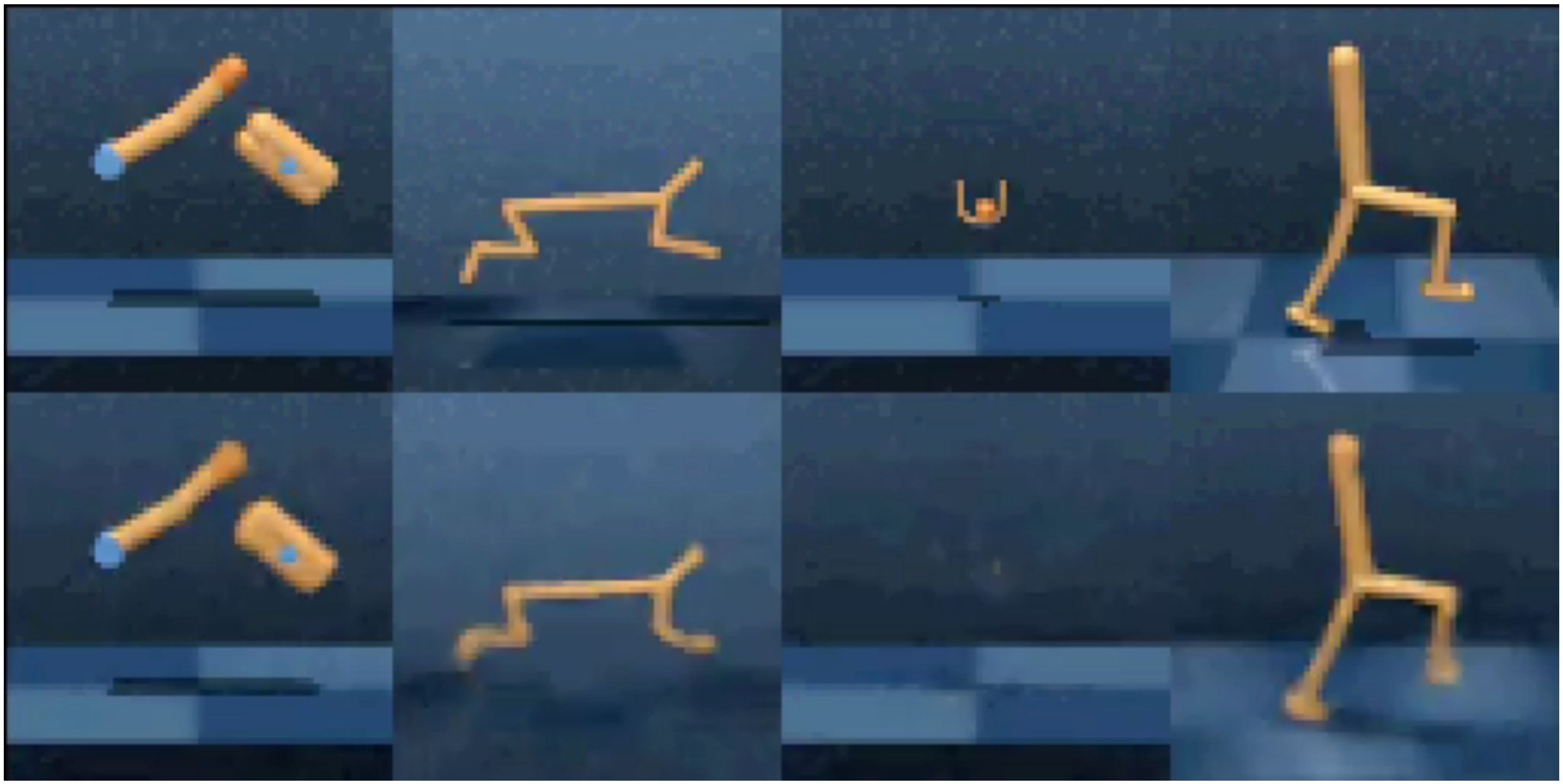


Hafner *et al.*, 2019

PlaNet

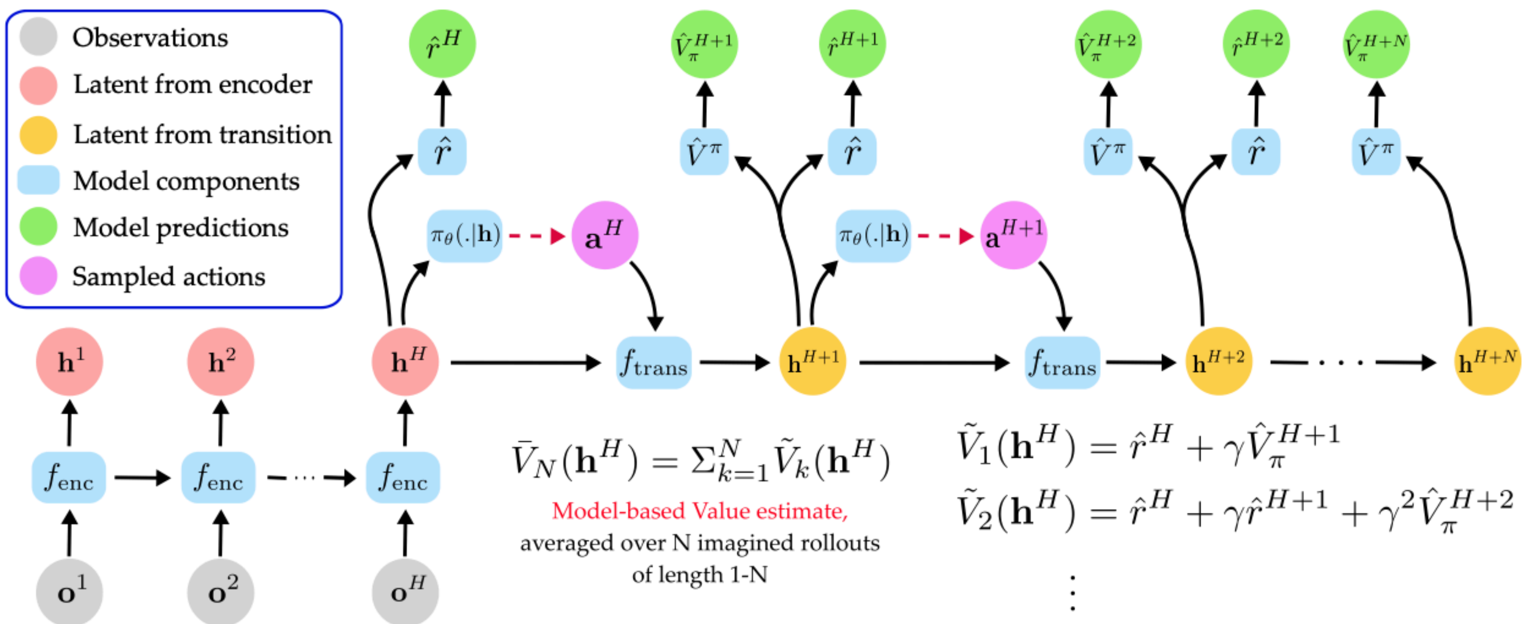
observations

predictions



Hafner *et al.*, 2019

Imagined Value Gradient



Uses a latent space learned through reconstruction/prediction

Uses a policy network for policy optimization

Continuous control

Byravan *et al.*, 2019

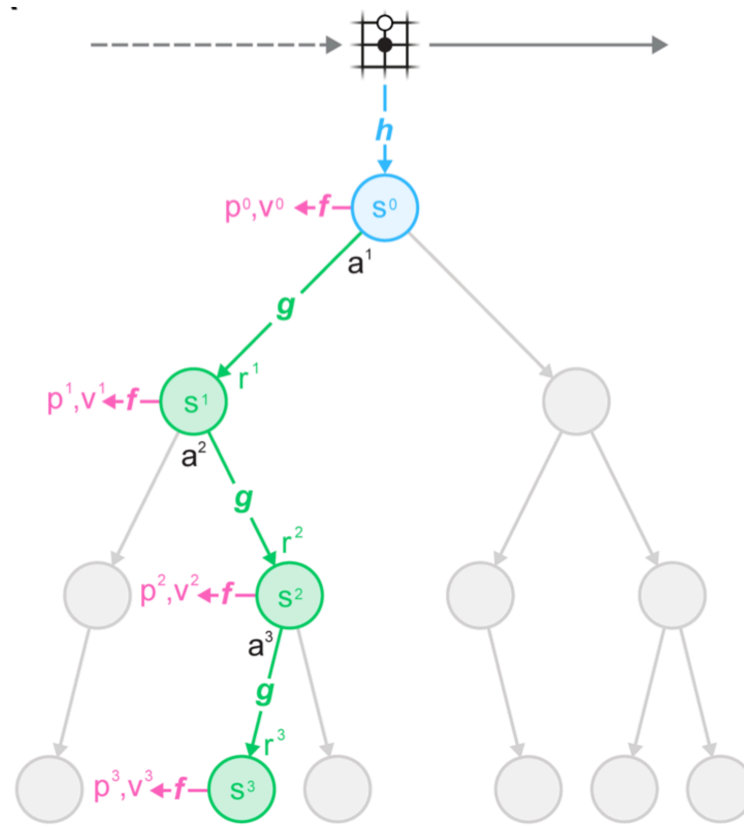
MuZero

Just predict the future reward, actions, and values

- mapping from observations to latent state (h)

- latent dynamics (g)

- mapping from latent state to predictions (f)

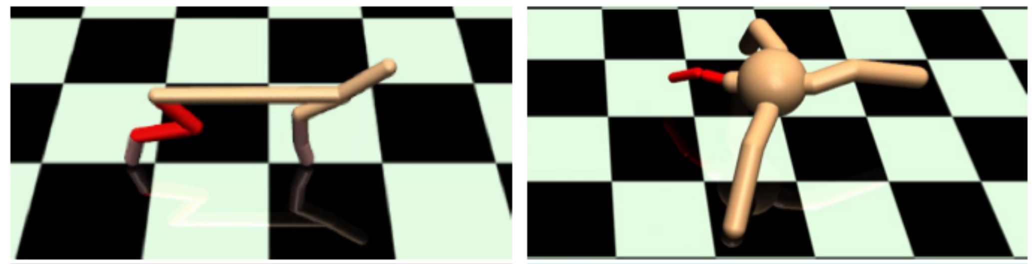


Monte Carlo Tree Search for policy optimization

discrete actions spaces

Schrittwieser *et al.*, 2019

Model-Based Meta-Learning



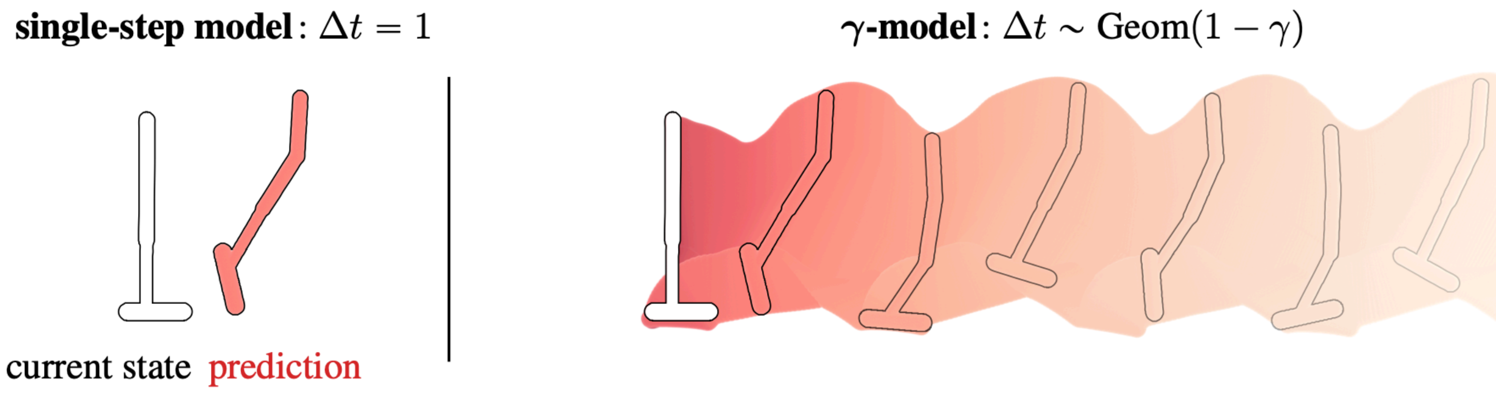
dynamically allocate new models as the environment dynamics change

can adapt to changes online

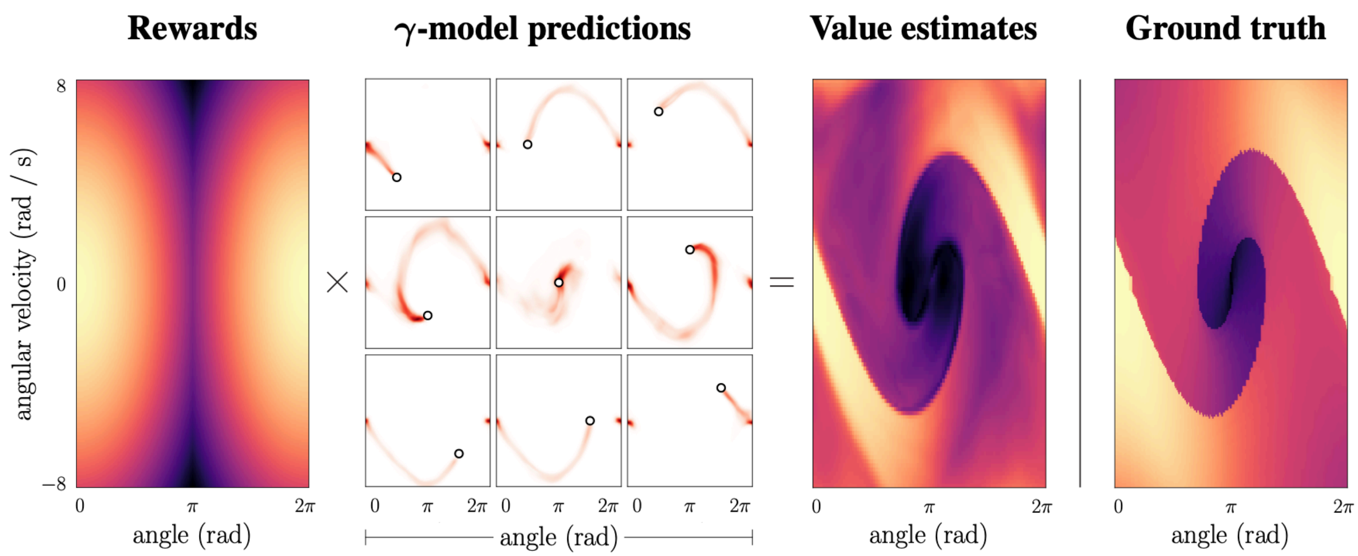
1-step models with MPC planning

Nagabandi *et al.*, 2019

Gamma Models



predict the future state distribution instead of just the next state



see also successor representation (Dayan, 1993)

Janner *et al.*, 2020

Other papers…

PILCO (Diesenroth, et al., 2013)

stochastic value gradient (Heess, Wayne, et al., 2015)

AlphaGo (Silver et al., 2016)

Imagination Augmented Agents (Weber et al., 2017)

Predictron (Silver et al., 2017)

POPLIN (Wang & Ba, 2019)

on the model-based stochastic value gradient (Amos et al, 2020)

…

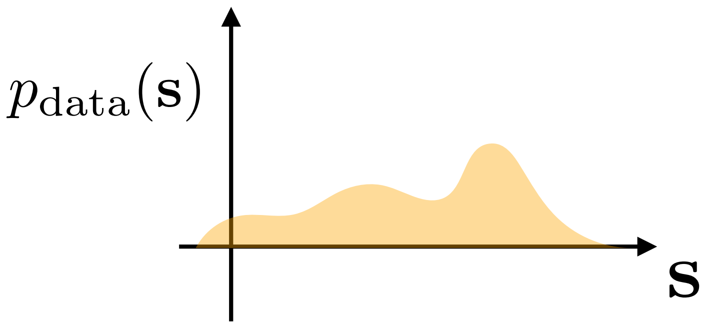
see list of references from Hamrick / Mordatch tutorial

Janner *et al.*, 2020

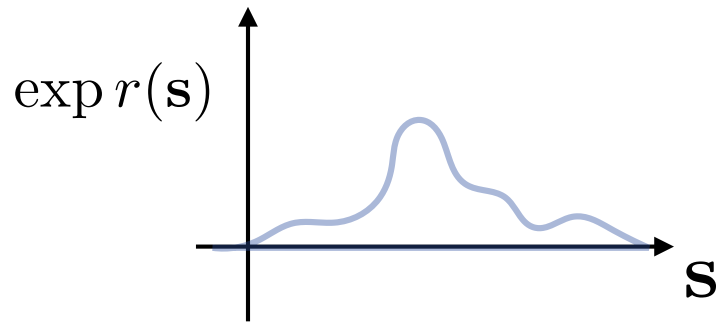
CONSIDERATIONS & OPEN ISSUES

Objective Mismatch

generative modeling ≠ reward maximization



generative modeling weights states according to their frequency



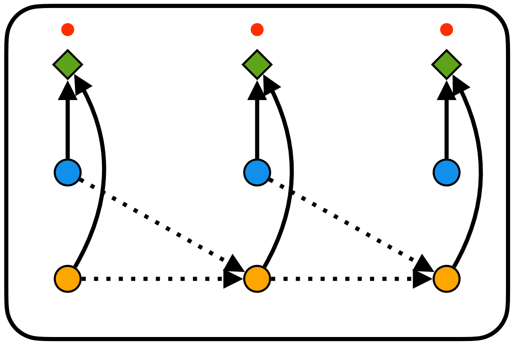
but not every state has the same importance for the overall task

this ***objective mismatch*** can result in sub-optimal final performance

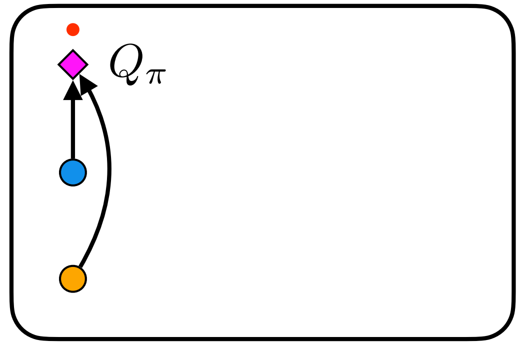
Computation

model-based rollouts are more costly for training / policy optimization

MB



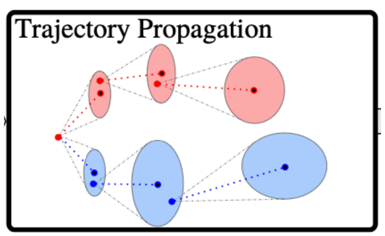
vs.



MF

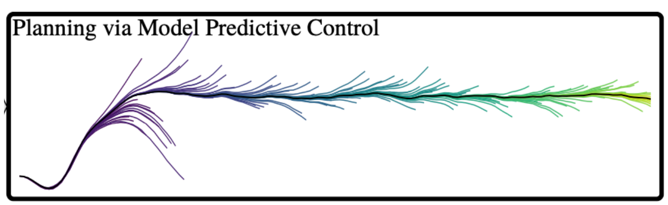
typically require a value function anyway when using short rollout horizon

generally requires more action samples, due to higher variance estimates



MB: ~10s of samples

MF: often 1 sample



Chua et al., 2018

distillation (MB —> MF)

dyna: use model as simulator for entire MF algorithm

e.g., ME-TRPO, World Models, etc.

value estimator: use model to estimate gradients for policy/value network

e.g., MVE / STEVE, Dreamer, etc.

MBPO does both

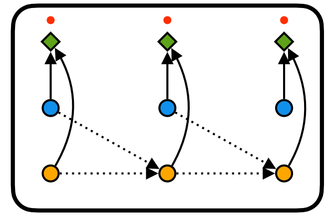
Combining MB + MF

we're still trying to understand where and how models should be used

use model to estimate target values

model-based value expansion (MVE) (Feinberg et al., 2018),
stochastic ensemble value expansion (STEVE) (Buckman et al., 2018)
model-based policy optimization (MBPO) (Janner et al., 2019)

TD loss: $$\left(Q_\pi(\mathbf{s}_t, \mathbf{a}_t) - r(\mathbf{s}_t, \mathbf{a}_t) - \underbrace{\gamma \mathbb{E}_{\pi, p_{\mathrm{env}}}\left[Q_\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})\right]}_{\textit{future value}}\right)^2$$

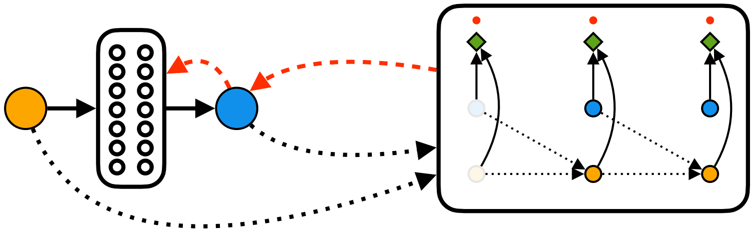


Effectively using model to estimate lower-bias Monte Carlo returns

use model to estimate policy gradients

imagined value gradients (Byravan et al., 2019),
dreamer (Hafner et al., 2020)

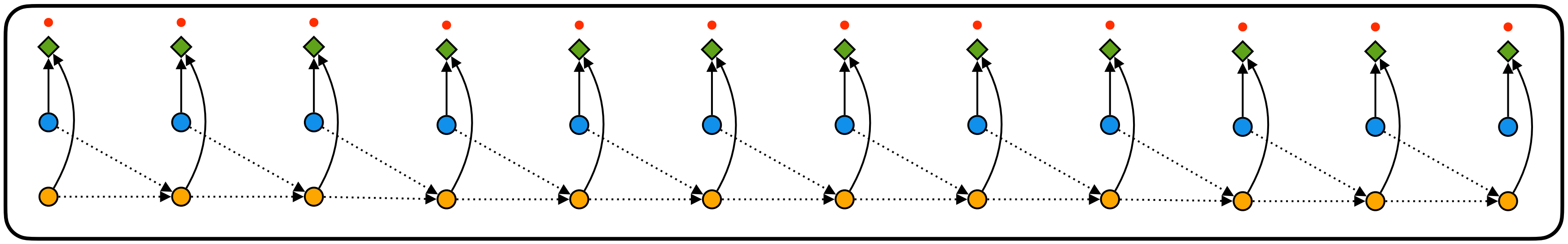
$Q_\pi(\mathbf{s}_t, \mathbf{a}_t)$



again, using model to estimate Monte Carlo returns, but now distilled into a policy network

Temporal Abstraction

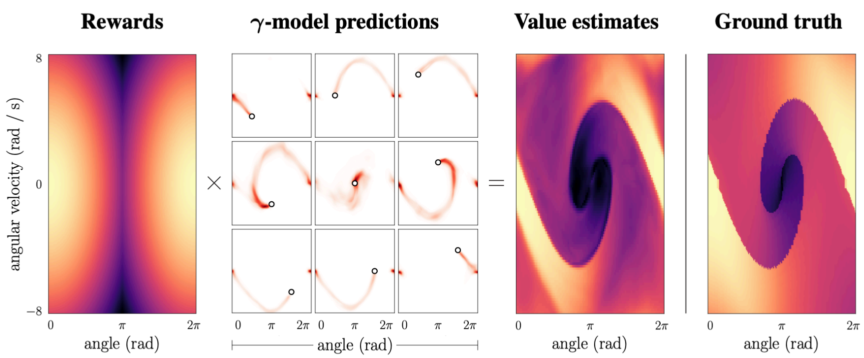
with a 1-step model, we are limited to planning at the sensing/environment frequency



for long-horizon tasks, planning becomes computation infeasible

successor representations

estimate long-term average state distribution



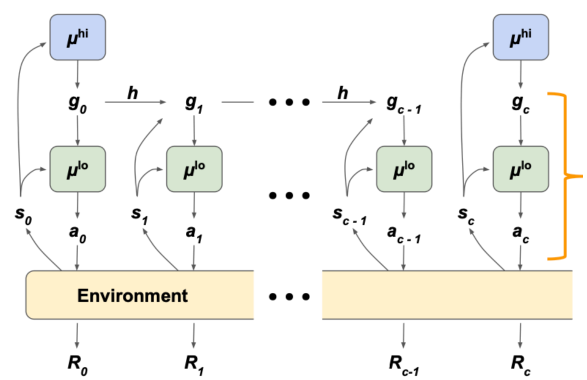
zero-shot long-term predictions…

but restricted to current policy

Janner et al., 2020

options, hierarchical RL

form a temporally sub-sampled latent space



high-level @ 0.1 Hz

low-level @ 1 Hz

…but often inflexible

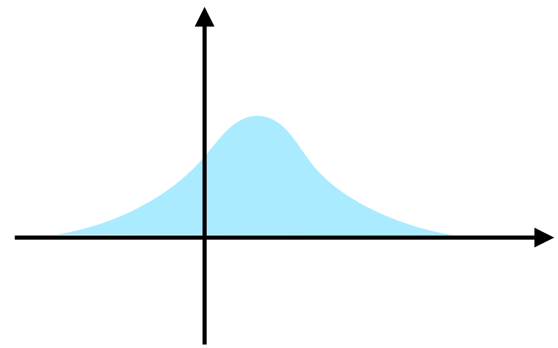
Nachum et al., 2018

PROJECT IDEAS

Dynamics Distribution Family

explore the effect of modifying the distribution family/factorization

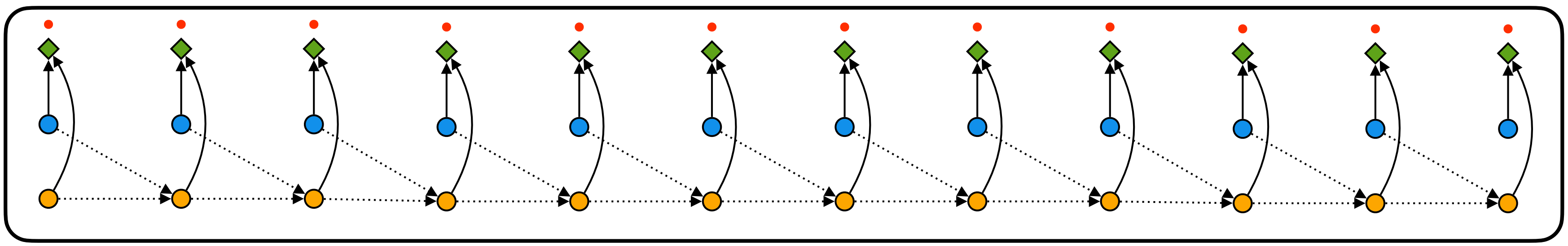
- Gaussian (diagonal or full covariance)
- Other exponential density (Laplace?)
- Mixture of Gaussians
- Flow-Based distribution
- etc.



Rollout Length

explore the effect of changing the rollout length

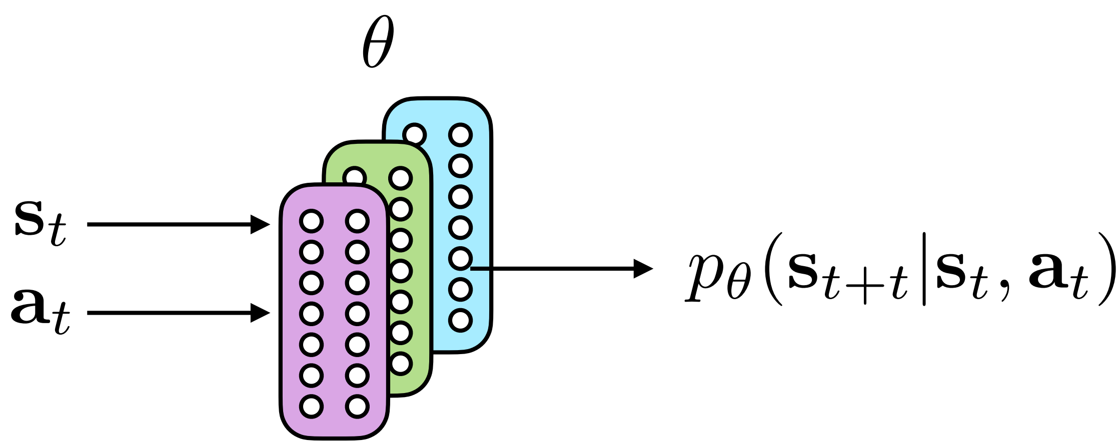
- analyze bias and variance of model's value estimate w.r.t. the true environment
- compare performance
- dynamically set rollout length? (see STEVE (Buckman et al., 2018))



Model Ensembles

explore the effect of using ensembles of models

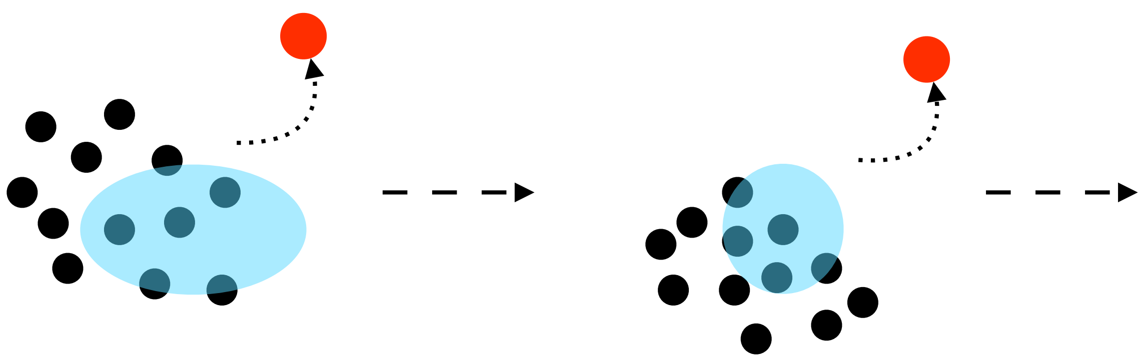
- how does performance vary with ensemble size?
- explore different sampling strategies for rollouts (see PETS (Chua et al., 2018))
- visualize cases where model ensembling helps with estimating uncertainty



Policy Optimizer

compare various policy optimizers in the context of a model-based value estimator

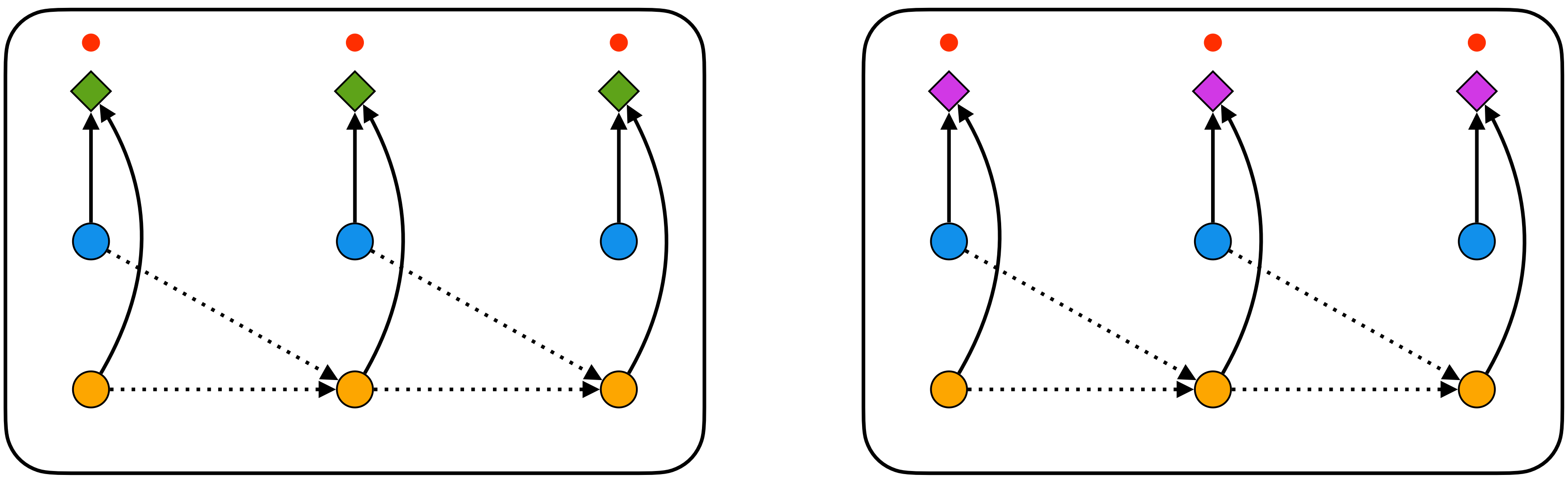
- compare accuracy and efficiency
- does better optimization accuracy lead to better performance?
- can optimizers be combined?



Model-Based Generalization

demonstrate task generalization with a model

- explore a multi-task setting in a particular environment, train a model on a subset of tasks and transfer to other tasks
- how well does the model generalize across tasks with varying similarity?



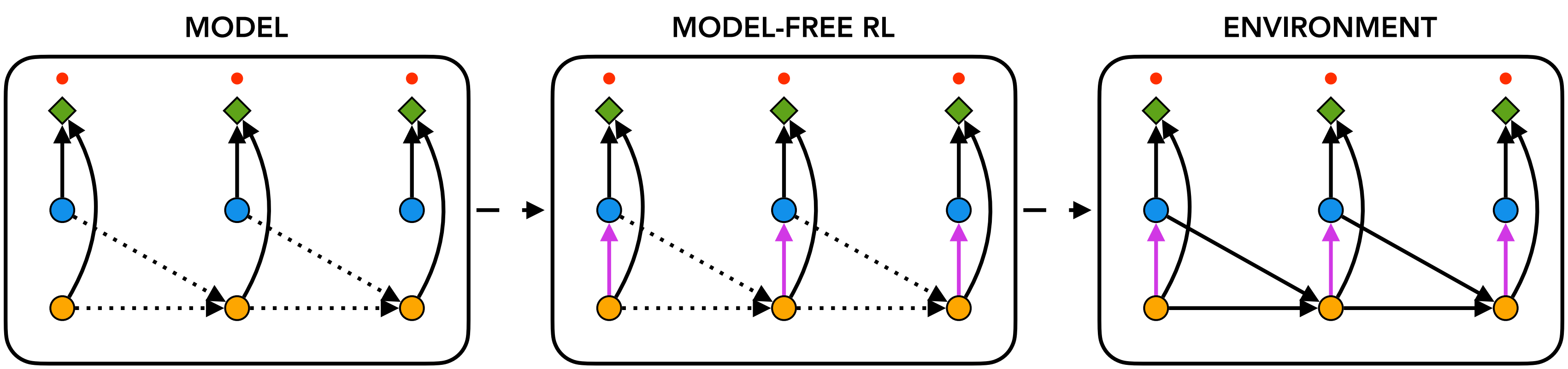
reward for task A

reward for task B

Model-Based + Model-Free

combine model-based and model-free algorithms

- use model-based value targets (MVE, STEVE)
  - explore various target estimation schemes (Monte Carlo, Retrace, etc.)
- use model-based policy gradients (Dreamer)
- use both (Dyna, MBPO)
- some other combination? e.g. distill via imitation learning (Nagabandi et al., 2017)



Additional Resources

Hamrick / Mordatch tutorial on MBRL: https://sites.google.com/view/mbrl-tutorial

→ *very thorough set of references*

On the role of planning in MBRL (Hamrick et al., 2021)

Benchmarking MBRL (Wang et al., 2019)

Lambert blog post on Debugging MBRL: https://www.natolambert.com/writing/debugging-mbrl