The Differentiable Cross-Entropy Method

Brandon Amos[1] Denis Yarats[1,2]
[1]Facebook AI Research [2]New York University

ICML 2020

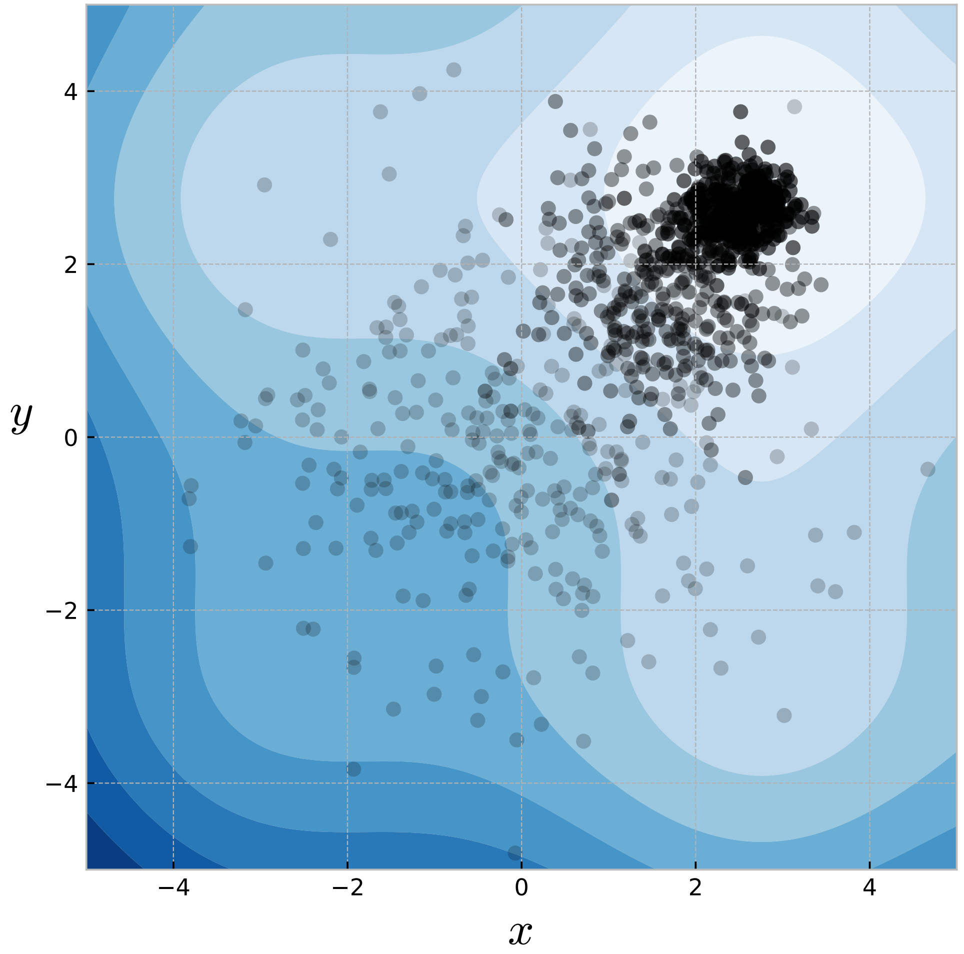
brandondamos
bamos.github.io
denisyarats
cs.nyu.edu/~dy1042

The cross-entropy method is a powerful optimizer

Iterative sampling-based optimizer that:

1. **Samples** from the domain
2. **Observes** the function's values
3. **Updates** the sampling distribution

Widely used in **control** and **model-based RL**

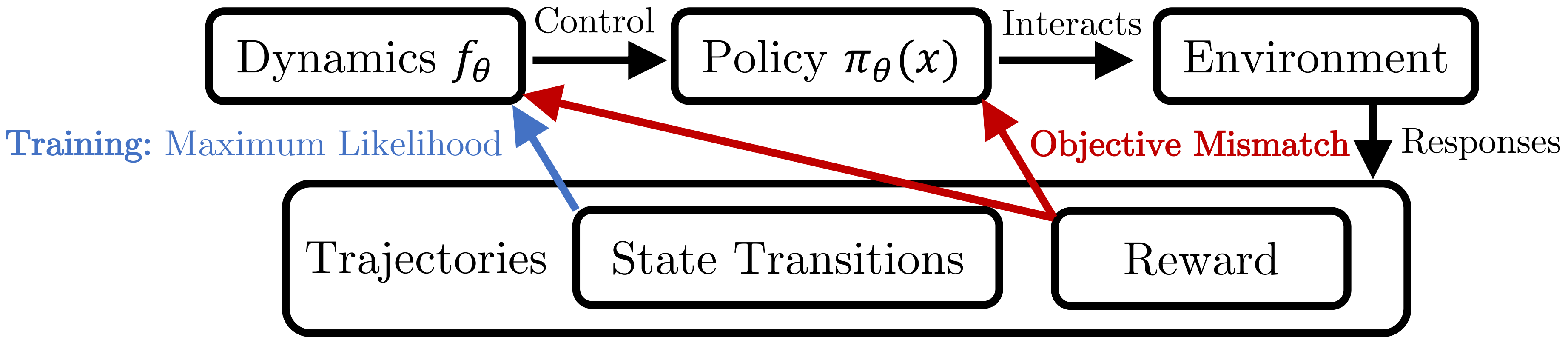


Problem: CEM breaks end-to-end learning

A common **learning pipeline**, e.g. for control, is

1. Fit models with **maximum likelihood**
2. **Run CEM** on top of the learned models
3. Hope CEM induces **reasonable downstream performance**

Objective mismatch issue: models are unaware of downstream performance

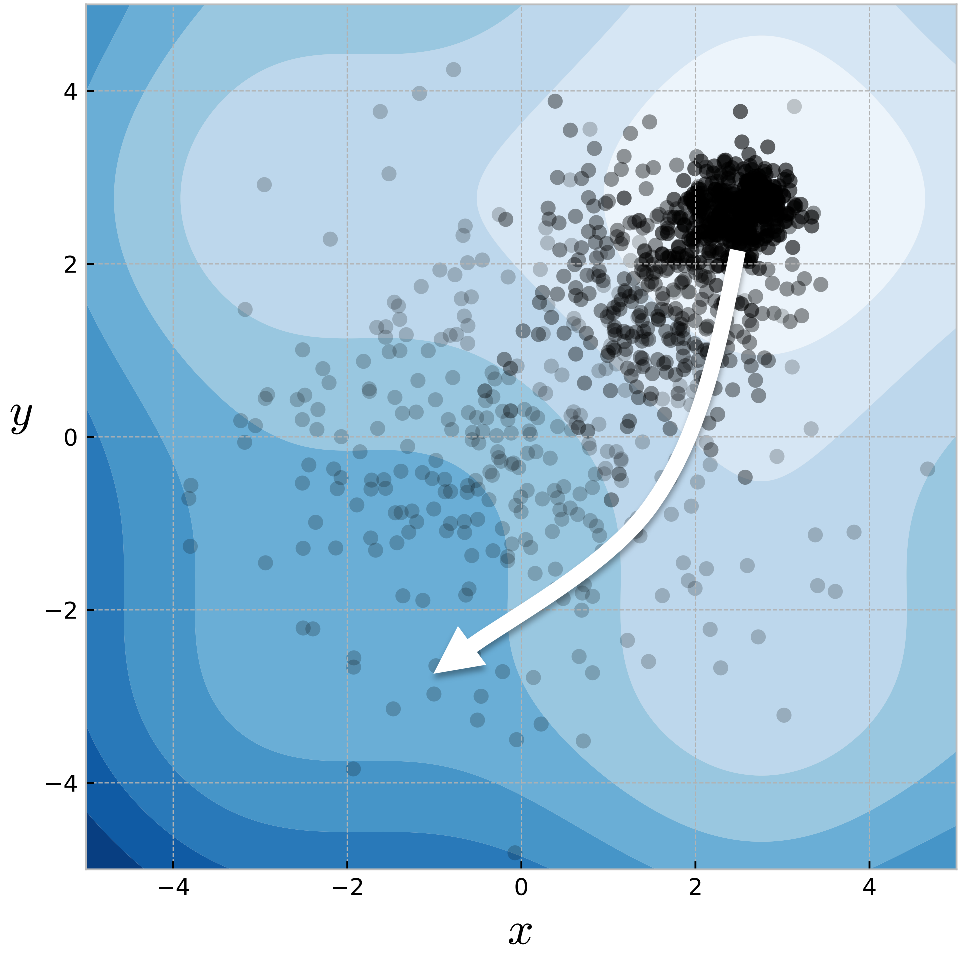


The Differentiable Cross-Entropy Method (DCEM)

Differentiate backwards through the sequence of samples
Using **differentiable top-k** (LML) and **reparameterization**

Useful when a fixed point is **hard to find,** or when unrolling gradient descent hits a local optimum

A **differentiable controller** in the RL setting



This Talk

Method: The differentiable-cross entropy method

Applications
- Learning deep energy-based models
- Learning embedded optimizers
- Control

Foundation: The Implicit Function Theorem

[Dini 1877, Dontchev and Rockafellar 2009]

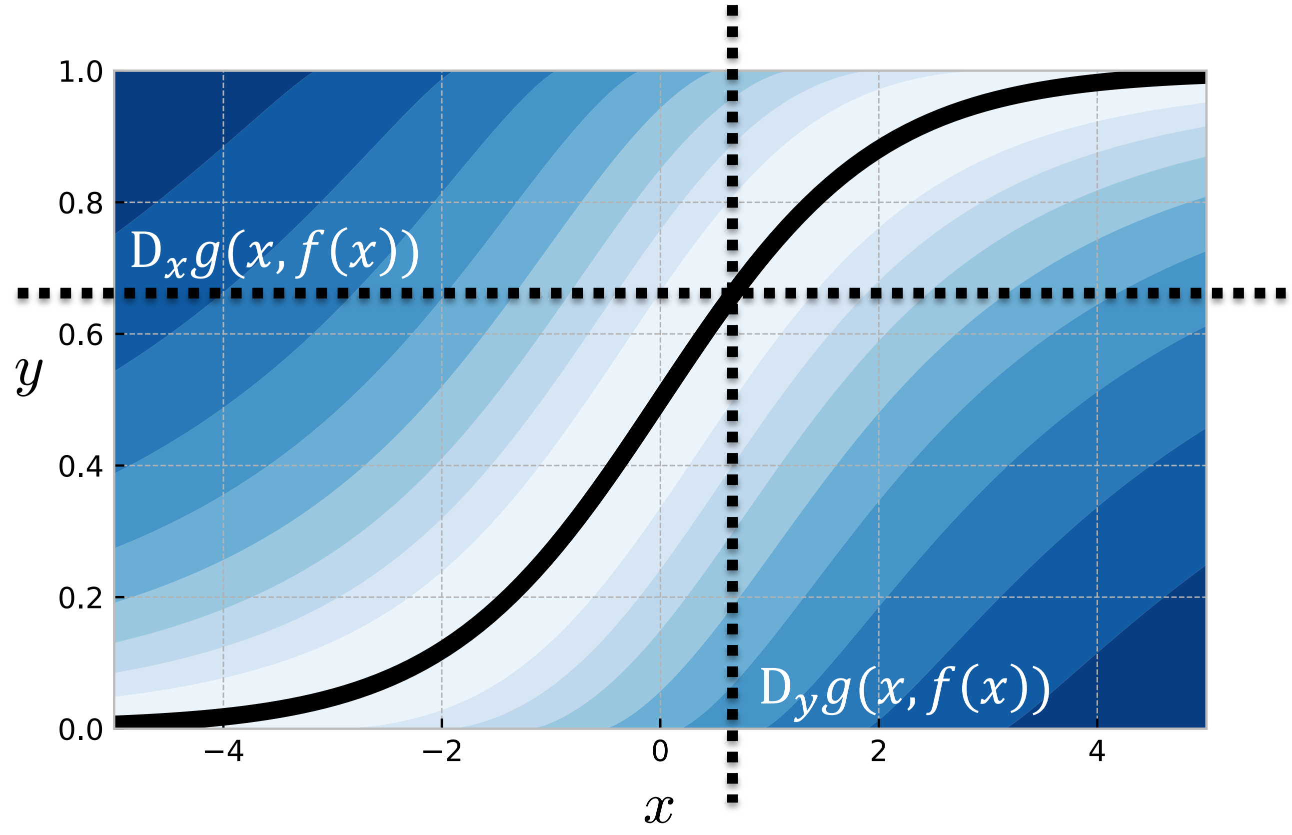
Given $g(x, y)$ and $f(x) = g(x, y')$, where $y' \in \{y : g(x, y) = 0\}$

How can we compute $\mathrm{D}_x f(x)$?

The **Implicit Function Theorem** gives

$$\mathrm{D}_x f(x) = -\mathrm{D}_y g\big(x, f(x)\big)^{-1} \mathrm{D}_x g\big(x, f(x)\big)$$

under mild assumptions



Foundation: Differentiable top-k operations

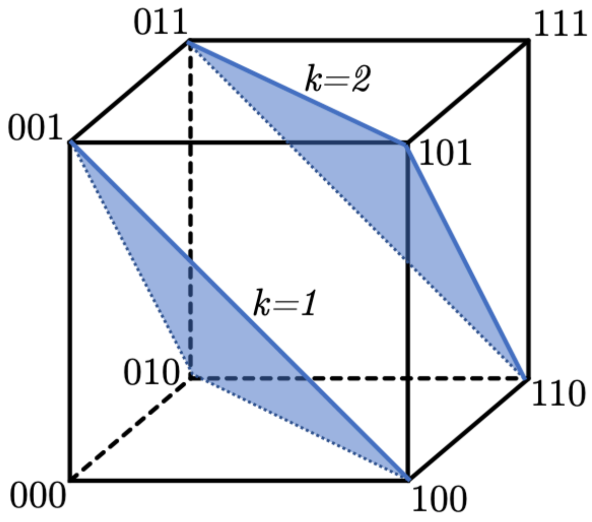
[Constrained softmax, constrained sparsemax, Limited Multi-Label Projection]

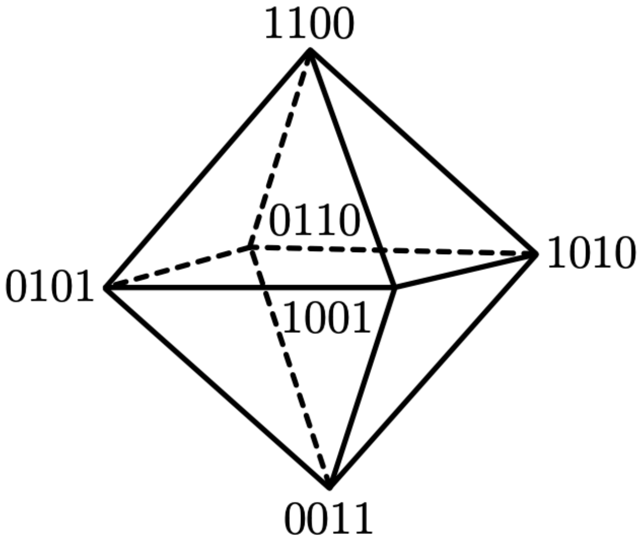
Optimization perspective of the softmax

$$\begin{aligned} y^\star = \ & \underset{y}{\operatorname{argmin}} \ -y^\top x - H(y) \\ & \text{subject to} \ 0 \le y \le 1 \\ & \qquad\qquad\ 1^\top y = 1 \end{aligned}$$

Limited Multi-Label Projection

$$\begin{aligned} y^\star = \ & \underset{y}{\operatorname{argmin}} \ -y^\top x - H_b(y) \\ & \text{subject to} \ 0 \le y \le 1 \\ & \qquad\qquad\ 1^\top y = k \end{aligned}$$





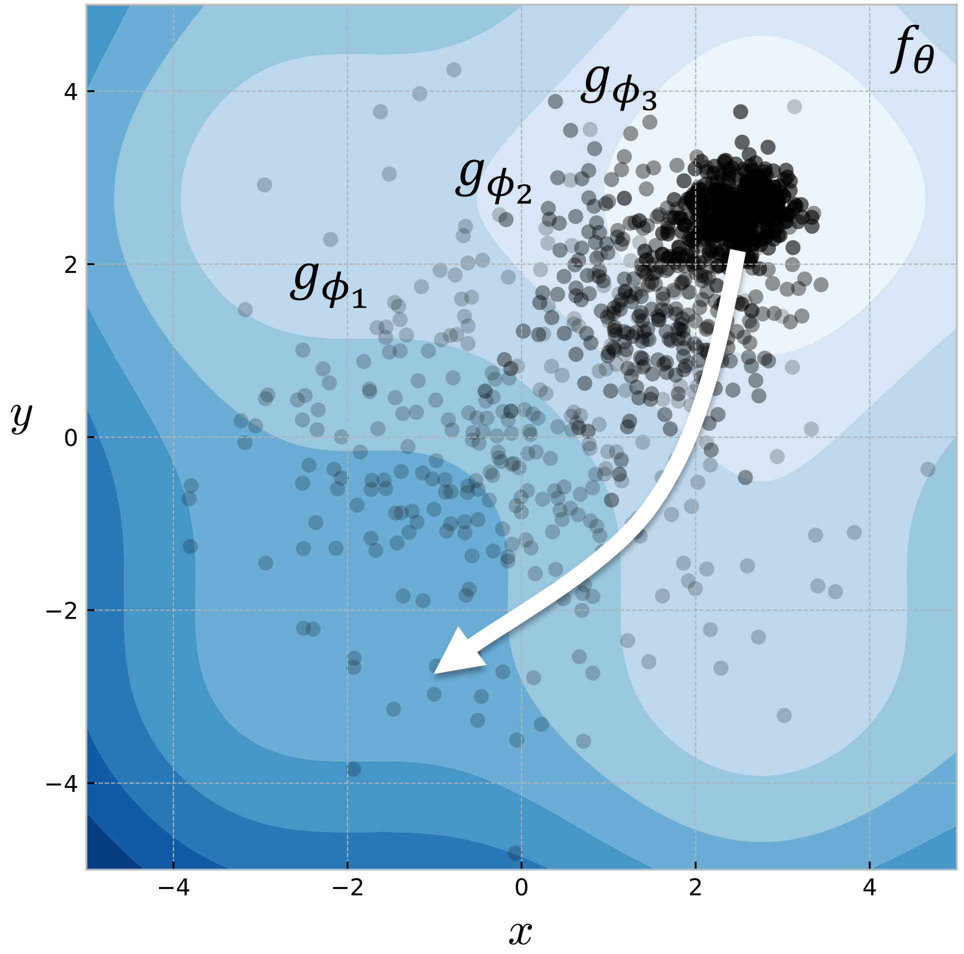
The Differentiable Cross-Entropy Method

In **each iteration**, update a distribution g_ϕ with:

$[X_{t,i}]_{i=1}^{N} \sim g_{\phi_t}(\cdot)$ **Sample** from the domain
$v_{t,i} = f_\theta(X_{t,i})$ **Observe** the function values
$\mathcal{I}_t = \Pi_{\mathcal{L}_k}(v_t/\tau)$ Compute the differentiable **top-k**
Update ϕ_{t+1} with maximum weighted likelihood

And finally return $\mathbb{E}[g_{\phi_{T+1}}(\cdot)]$

Captures vanilla CEM when the soft top-k is hard
Composed of operations with **informative derivatives**



This Talk

Method: The differentiable-cross entropy method

Applications

- Learning deep energy-based models
- Learning embedded optimizers
- Control

Deep Structured Energy Models (SPENs/ICNNs)

[Belanger and McCallum, 2016, Amos, Xu, and Kolter, 2017]

Key idea: Model $\mathrm{p}(x, y) \propto \exp\{-E_\theta(x, y)\}$ where E_θ is a deep energy model

Captures **non-trivial structures** in the output space, while also subsuming feed-forward modes

Feedforward model: $E(x, y) = \|f(x) - y\|_2^2$

Predict with the optimization problem:

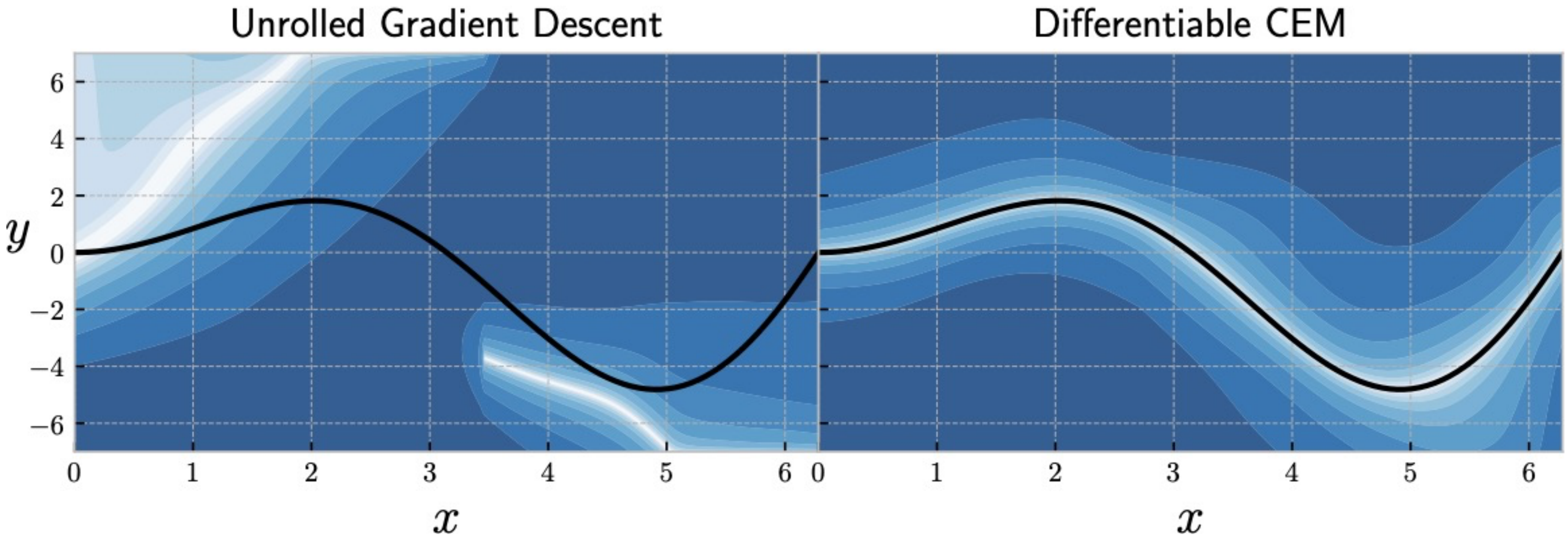
$$\hat{y} = \operatorname*{argmin}_y E_\theta(x, y)$$

Learning can be done by **unrolling optimization** on E_θ using derivative information $\nabla_y E$

Unrolling gradient descent may learn bad energies

Unrolling optimizers **lose the probabilistic interpretation** and can **overfit to the optimizer**

In this regression setting, **GD learns barriers** on the energy surface while **DCEM fits the data**



This Talk

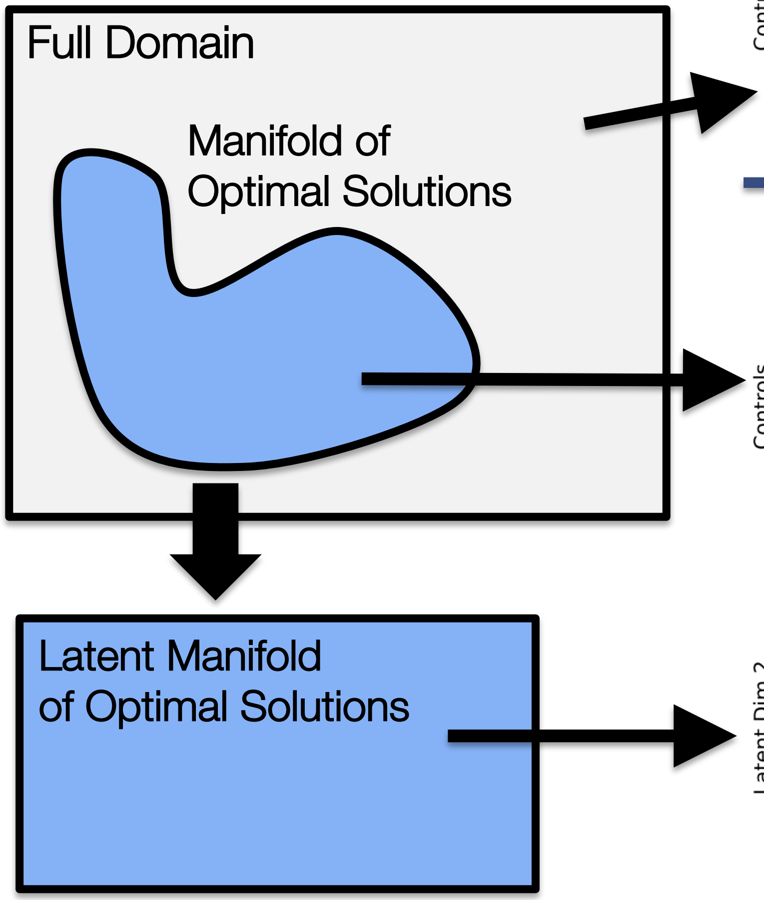
Method: The differentiable-cross entropy method

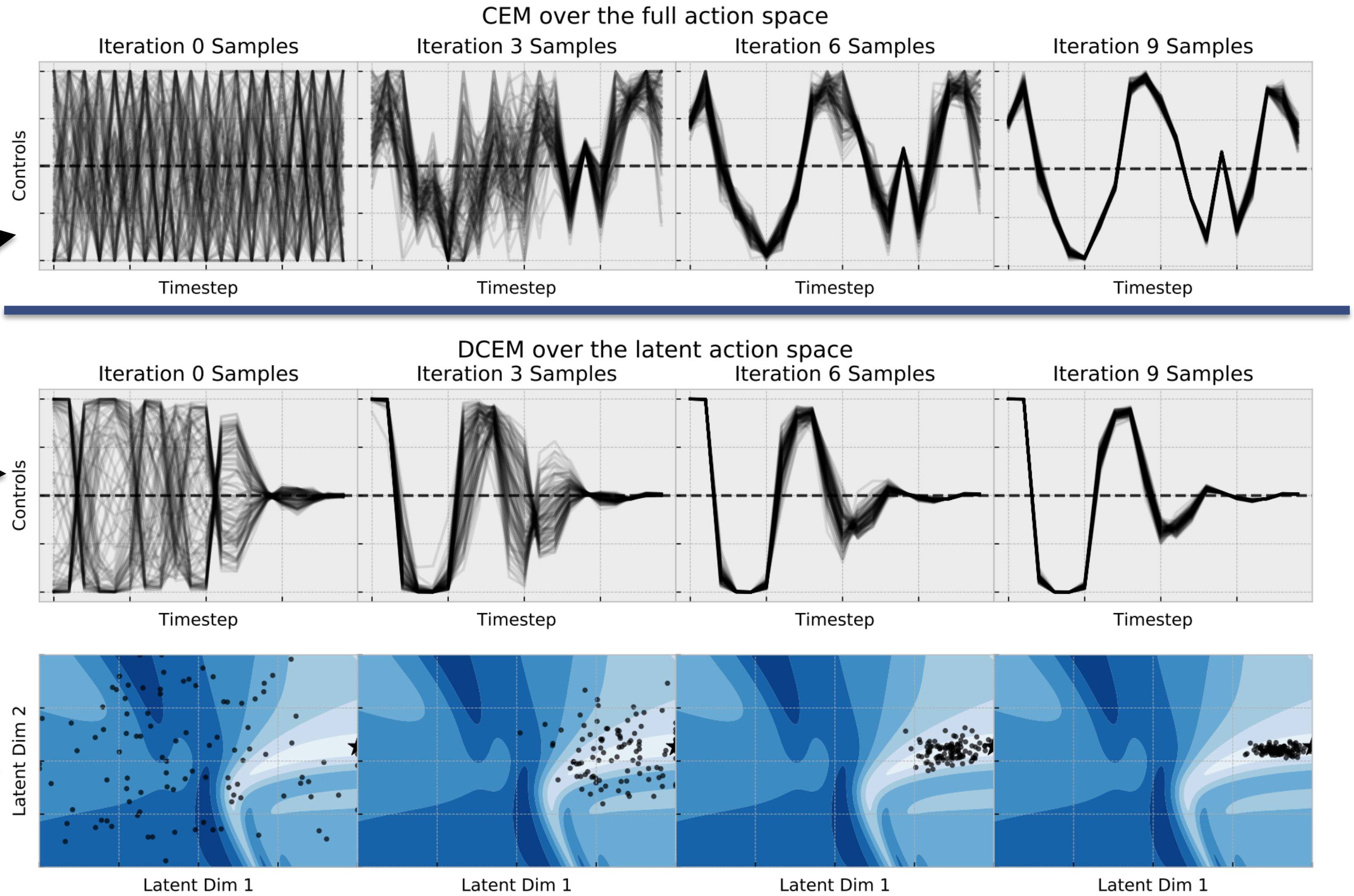
Applications

- Learning deep energy-based models
- Learning embedded optimizers
- Control

DCEM can exploit the solution space structure

$$x^\star = \underset{x \in [0,1]^N}{\operatorname{argmin}}\ f(x)$$



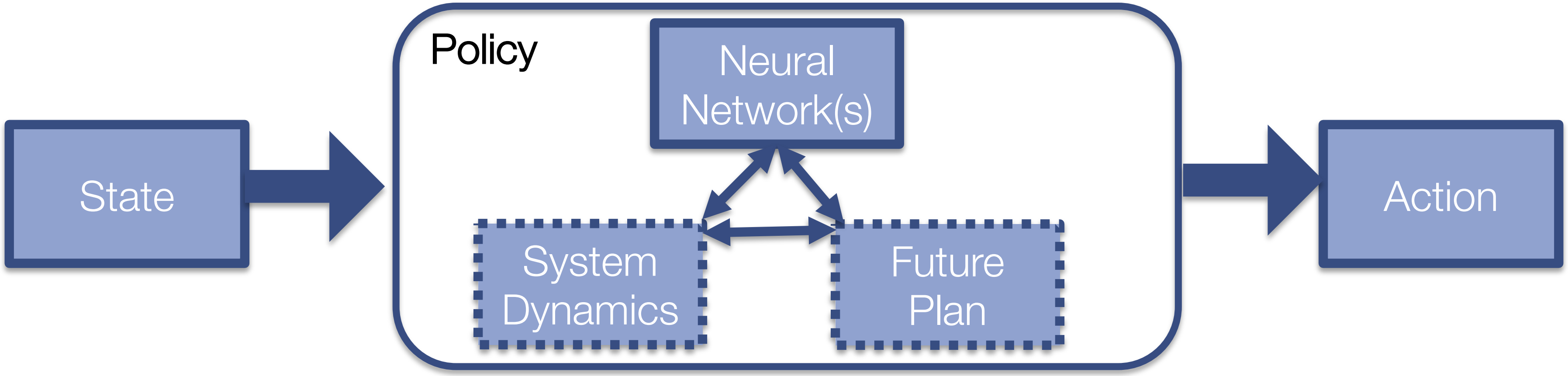


This Talk

Method: The differentiable-cross entropy method

Applications
- Learning deep energy-based models
- Learning embedded optimizers
- Control

Should RL policies have a system dynamics model or not?



Model-free RL

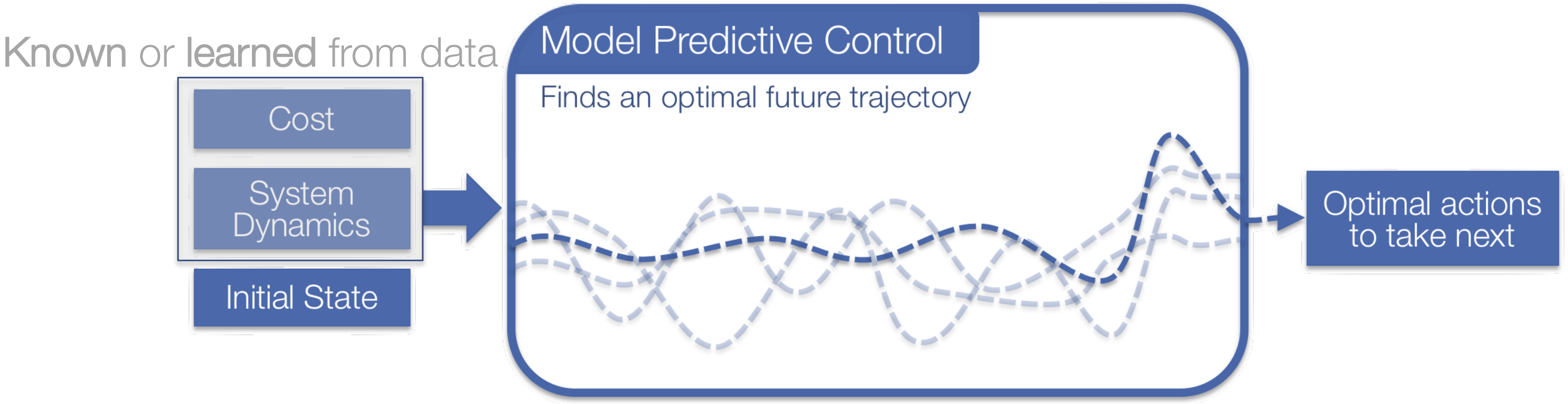
More general, doesn't make as many assumptions about the world

Rife with poor data efficiency and learning stability issues

Model-based RL (or control)

A useful prior on the world if it lies within your set of assumptions

Model Predictive Control



Differentiable Control via DCEM

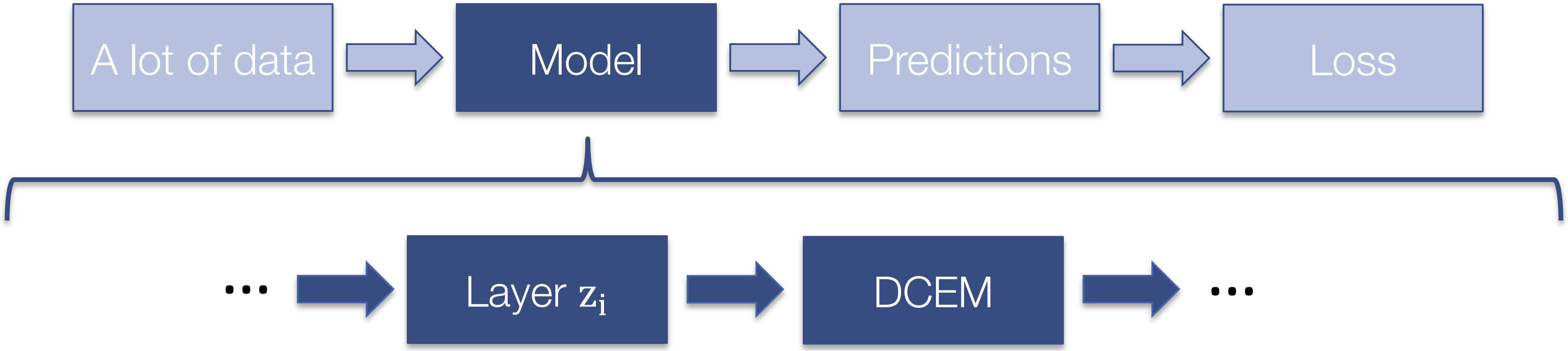
A pure **planning problem** given (potentially non-convex) **cost** and **dynamics**:

$$
\begin{aligned}
\tau_{1:T}^{\star} = \operatorname*{argmin}_{\tau_{1:T}} \; & \sum_t \boxed{C_\theta(\tau_t)} \text{ Cost} \\
\text{subject to } \; & x_1 = x_{\text{init}} \\
& x_{t+1} = \boxed{f_\theta(\tau_t)} \text{ Dynamics} \\
& \underline{u} \leq u \leq \overline{u}
\end{aligned}
$$

where $\tau_t = \{x_t, u_t\}$

Idea: Solve this optimization problem with DCEM and differentiate through it

Differentiable Control via DCEM

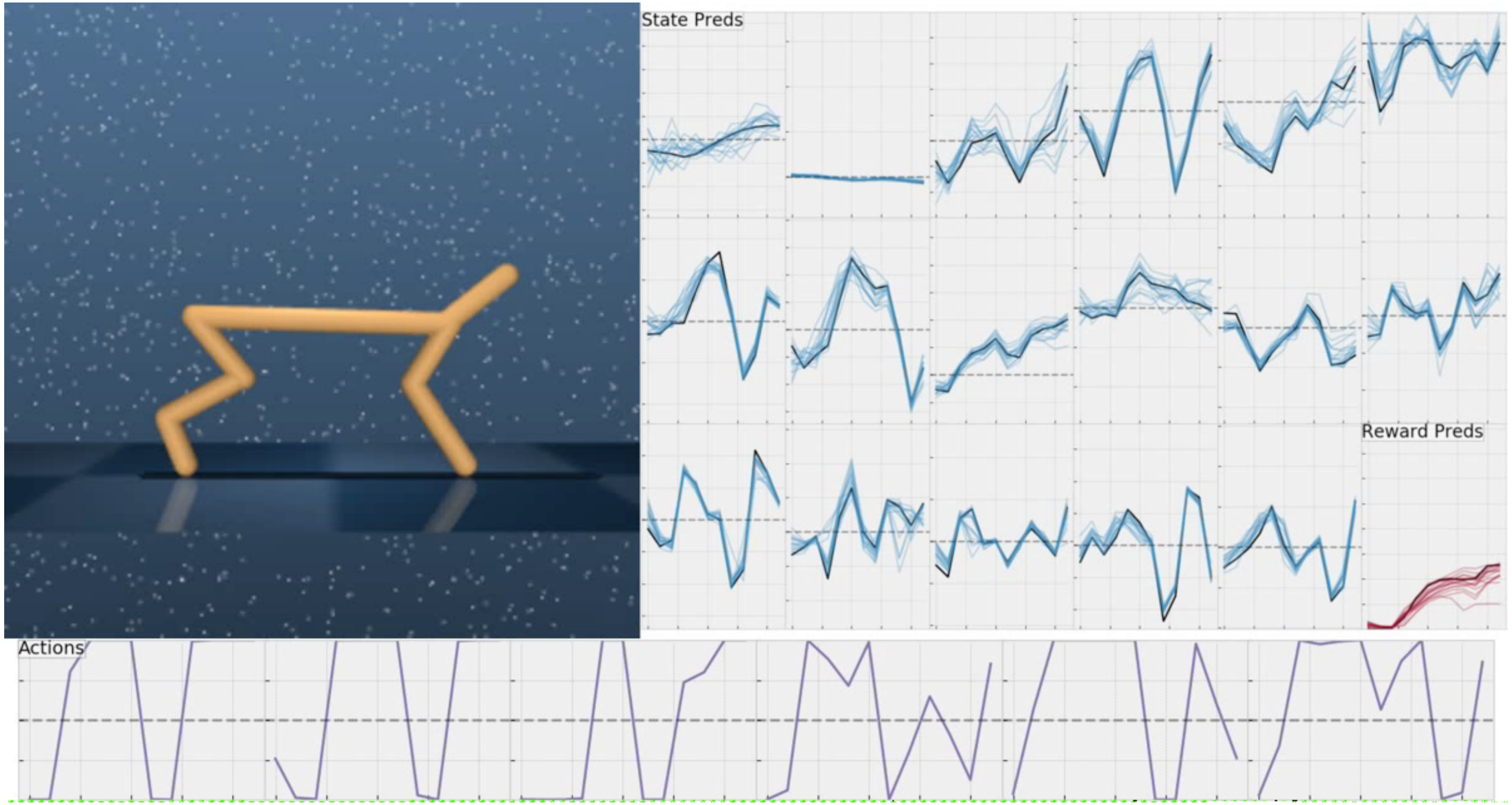


What can we do with this now?

Augment neural network policies in model-free algorithms with MPC policies
Fight objective mismatch by end-to-end learning dynamics
The cost can also be end-to-end learned! No longer need to hard-code in values

Caveat: Control problems are often intractably high-dimensional, so we use embedded DCEM

DCEM fine-tunes highly non-convex controllers



sites.google.com/view/diff-cross-entropy-method

The Differentiable Cross-Entropy Method

Brandon Amos[1] Denis Yarats[12]

[1]Facebook AI Research [2]New York University

ICML 2020

brandondamos denisyarats

bamos.github.io cs.nyu.edu/~dy1042