

# Learning with differentiable and amortized optimization

Brandon Amos • Meta AI (FAIR) NYC



<http://github.com/bamos/presentations>

# Optimization is crucial technology

*Optimization is a **modeling** and **decision-making** paradigm and **encodes reasoning operations***

Breakthroughs over the past century enabled by optimization advancements include:

1. **controlling systems** (robotic, autonomous, mechanical, and multi-agent)
2. **making operational decisions** based on future predictions
3. efficiently **transporting** or **matching** resources, information, and measures  
for **optimal transport** and **generative modeling**
4. **allocating** budgets and portfolios,
5. **designing** materials, molecules, and other structures,
6. **solving inverse problems**  
infers underlying hidden costs, incentives, geometries, terrains, and other structures
7. **parameter learning** of predictive and statistical models

# Optimal control drives systems

**Setting:** deterministic, discrete-time system with a **continuous state-action space**

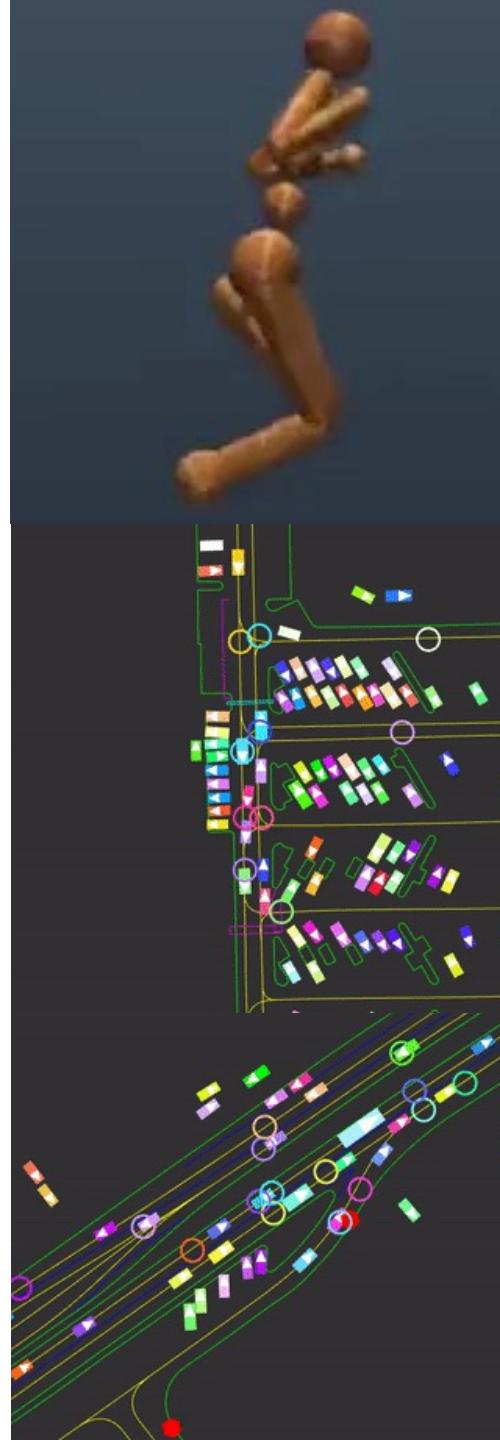
$$x_{1:T}^*, u_{1:T}^* \in \operatorname{argmin}_{x_{1:T}, u_{1:T}} \sum_t \begin{array}{|c|} \text{cost} \\ \hline C(x_t, u_t) \end{array} \text{ s.t. } \begin{array}{|c|} \text{initial state} \\ \hline x_1 = x_{\text{init}} \end{array} \quad \begin{array}{|c|} \text{dynamics} \\ \hline x_{t+1} = f(x_t, u_t) \end{array} \quad \begin{array}{|c|} \text{constraints} \\ \hline u_t \in \mathcal{U} \end{array}$$

Full notation:  $u_{1:T}^*(x_{\text{init}}, f_\theta)$

**Widely deployed** over the past century in aviation, robotics, vehicles, HVAC  
Often for a **Markov decision process** but doesn't have to be

The **real-world is non-convex**, so are our controllers  
**Convex** in some cases, e.g., with quadratic cost/linear dynamics (LQR)

**No learning necessary** if we know the system — just pure optimization



# Optimal transport connects measures

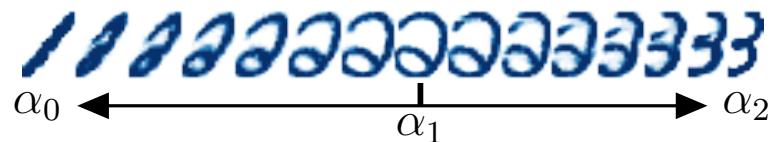
**Setting:** transport between measures  $\alpha$  and  $\beta$

**Monge** problem for Euclidean Wasserstein-2

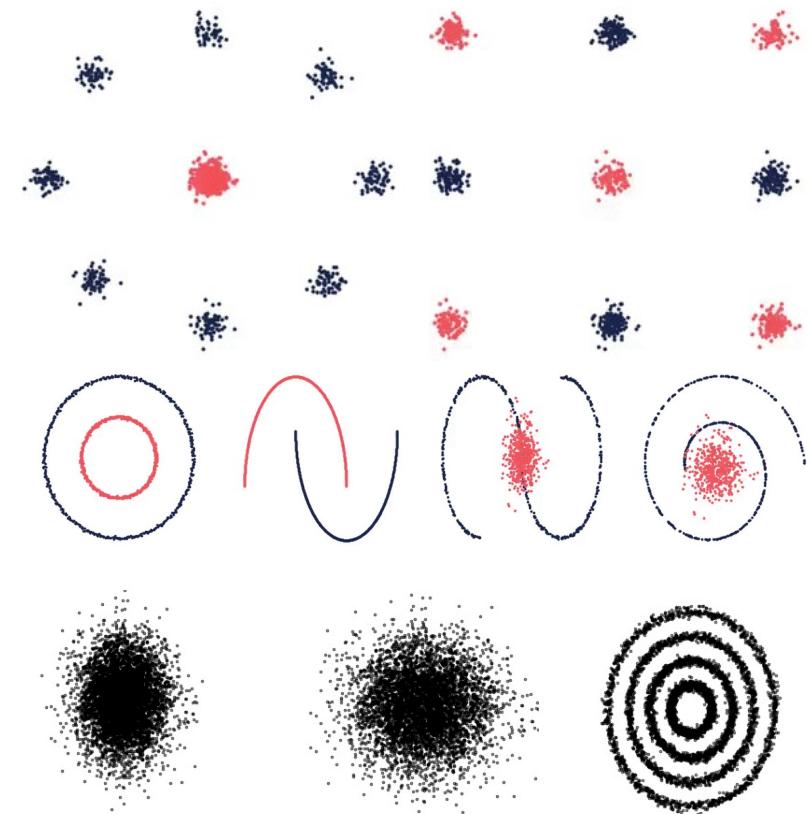
$$T^*(\alpha, \beta) \in \operatorname{argmin}_{T \in \mathcal{C}(\alpha, \beta)} \mathbb{E}_{x \sim \alpha} \|x - T(x)\|_2^2$$

The transport map  $T^*$  **connects**  $\alpha$  to  $\beta$

**No learning necessary**, again just optimization



*Meta Optimal Transport. Amos et al., 2022*

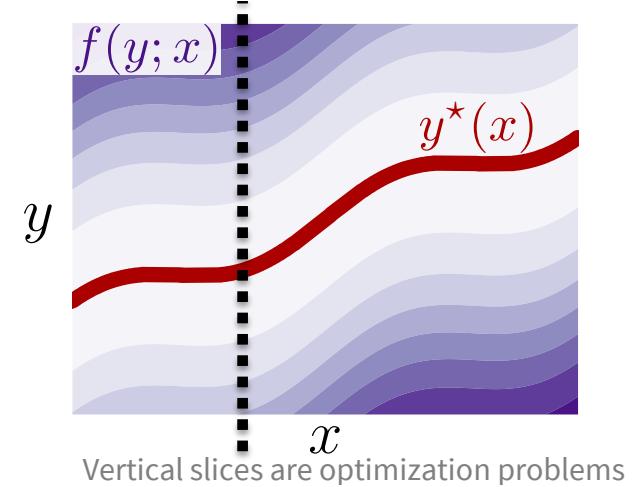


*On amortizing convex conjugates for optimal transport. Amos, 2022*

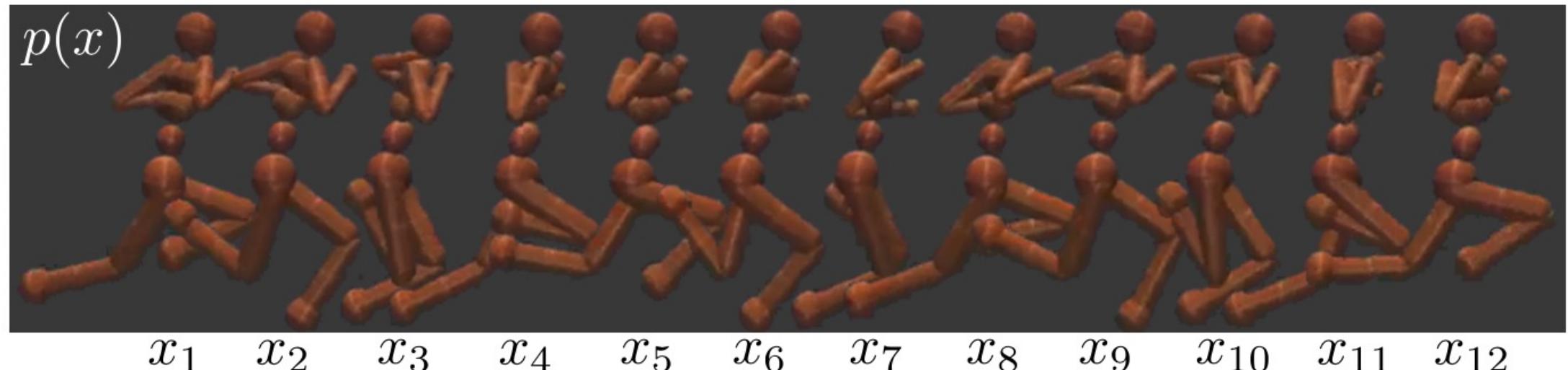
# The magic of optimization

Find the **best way to interact** with a **representation of the world**

$$\begin{array}{c} \text{optimal solution} \\ | \\ y^*(x) \in \operatorname{argmin}_{y \in \mathcal{C}(x)} f(y; x) \\ | \qquad \qquad \qquad | \\ \text{optimization variable} \quad \text{constraints} \\ | \qquad \qquad \qquad | \\ \text{objective} \qquad \text{context (or parameterization)} \end{array}$$



In control,  $x$  is the **system state**,  $y$  is the **action**,  $f(y; x)$  is a **cost**, and  $y^*(x)$  is an **optimal action**



# When optimization fails, machine learning helps

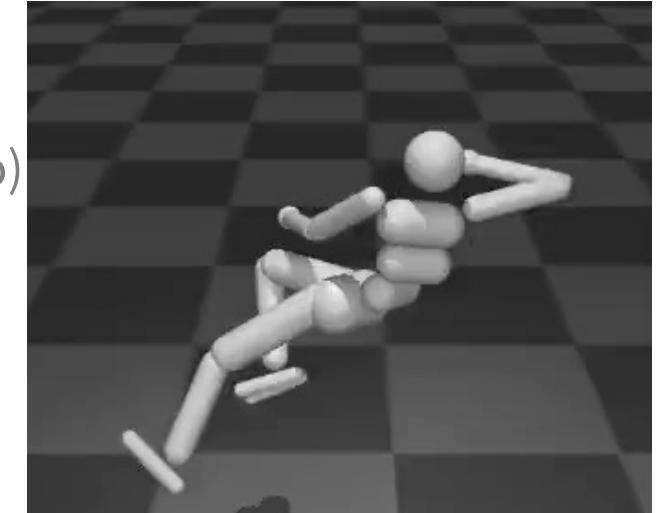
$$y^*(x) \in \operatorname{argmin}_{y \in \mathcal{C}(x)} f(y; x)$$

**Optimization starts failing us when:**

1. Analytically **encoding every detail** into  $f$  and  $\mathcal{C}$  is impossible
  - May be **unknown, mis-specified, or inaccurate**
  - Especially difficult with **high-dimensional data** (text, images, video)
2. **Solving** for  $y^*(x)$  is **intractable** and **difficult to compute**

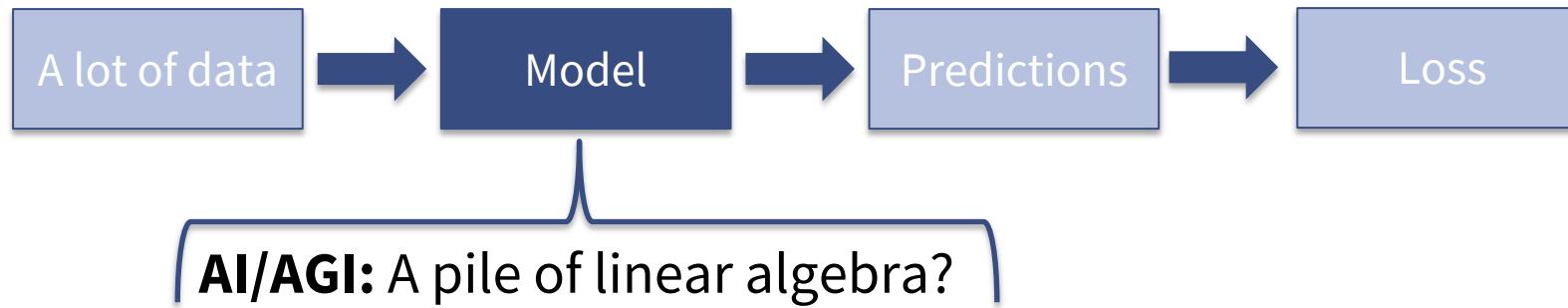
Machine learning systems are **part of the solution** and **excel** at:

1. **Modeling systems** we can extract a lot of data from
2. Extracting **meaningful latent representations**
3. Making **fast and accurate** predictions



# When optimization fails, machine learning helps

# When machine learning fails, optimization helps

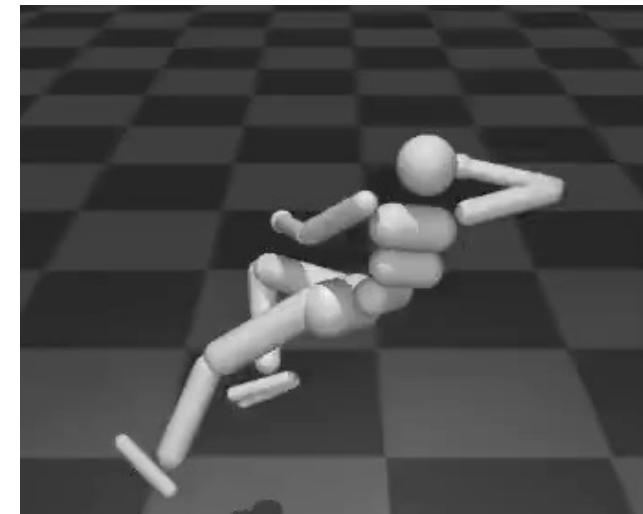


Is a **black-box of linear algebra** the **solution to AI/AGI?** *Maybe...*

Without enough data, learning systems struggle to:

1. **Encode hard constraints** and **rules** of the world crucial for safety
2. Perform **reasoning operations**
3. Isolate **interpretable** and **modular subcomponents**
4. **Generalize** to new settings beyond the training distribution

Optimization often **excels in these** and **can be brought into the model/loss**



# Integrating optimization and machine learning

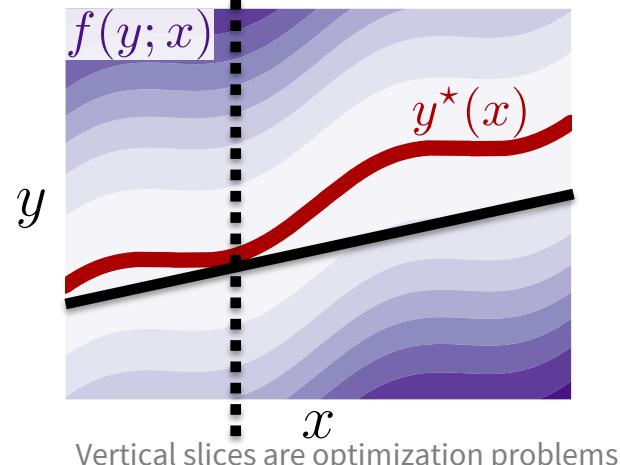
**Key:** view **optimization as a function** from the context  $x$  to the solution  $y^*(x) \in \operatorname{argmin}_{y \in \mathcal{C}(x)} f(y; x)$

**Differentiable optimization** —  $\frac{\partial}{\partial x} y^*(x)$

Optimization is a function that can be **differentiated**

Tells how **perturbing the context changes the solution**

Enables **optimization as a layer** or loss for ML

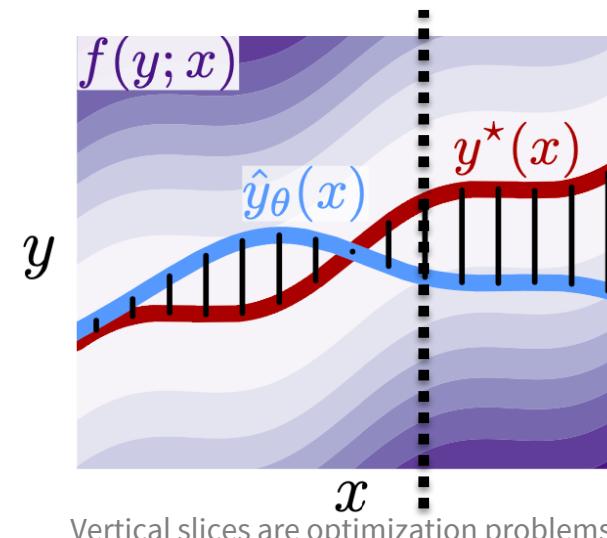


**Amortized optimization** —  $\hat{y}_\theta(x) \approx y^*(x)$

Optimization is a function that can be **approximated**

**Approximate the solution map** with  $\hat{y}_\theta$

**Power of optimization, fast speed of an ML model**



# This talk

**Differentiable optimization** —  $\frac{\partial}{\partial x} y^*(x)$

Foundations

Differentiable convex optimization layers

Differentiable control

Optimization-based task losses

**Amortized optimization** —  $\hat{y}_\theta(x) \approx y^*(x)$

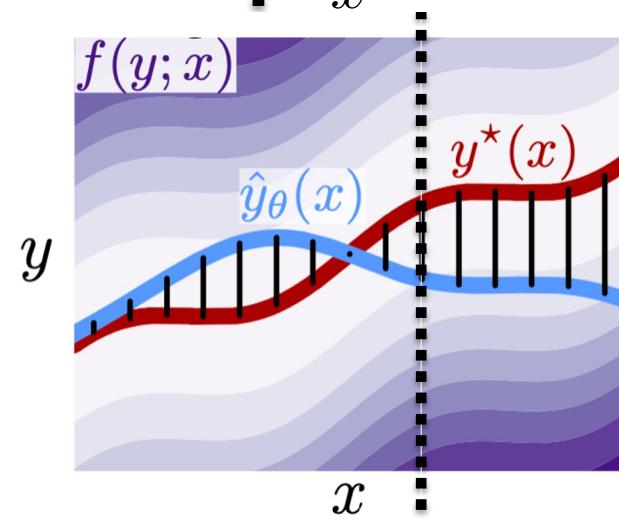
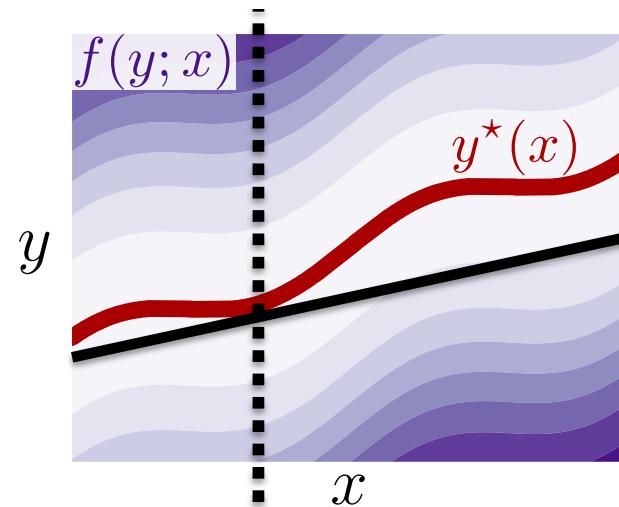
Foundations

RL as amortized optimization

Amortization via learning latent subspaces

Meta Optimal Transport

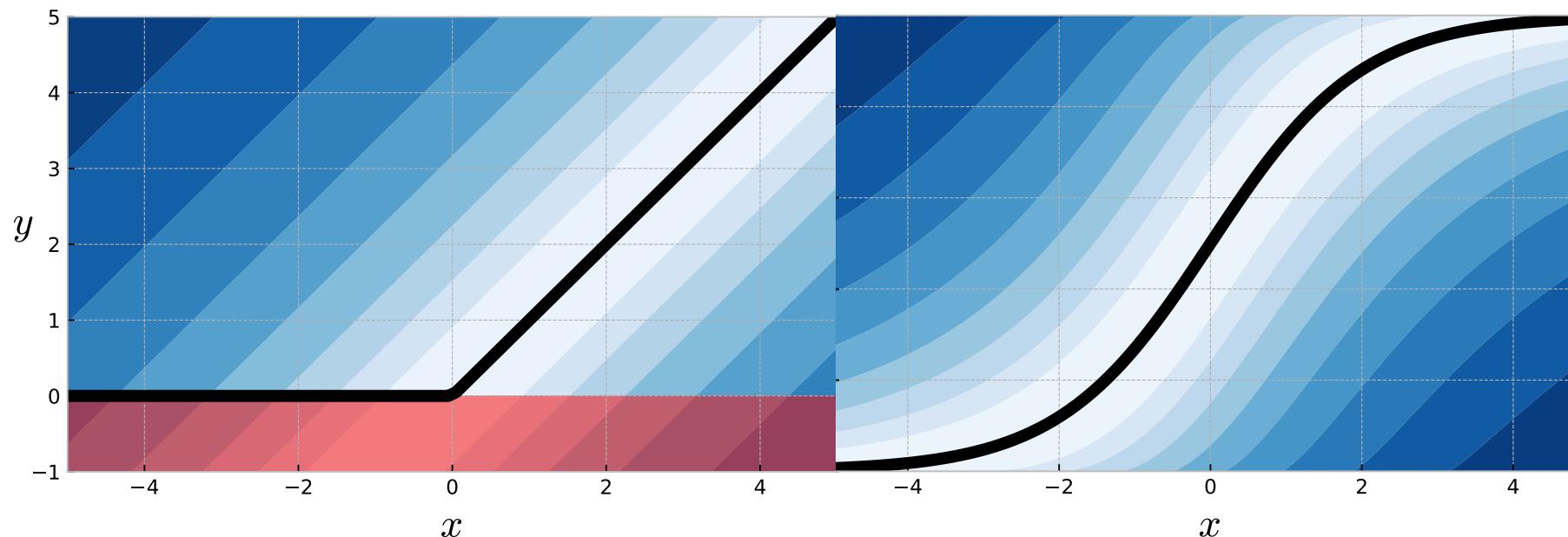
Amortizing convex conjugates



# Convex optimization is expressive

The **argmin** of a **convex** optimization problem is **non-convex** and **expressive**  
Standard non-linearities to be seen as **solutions** to convex optimization problems  
We'll start simple for **intuition** and **motivation to generalize beyond these**

$$y^*(x) = \underset{y}{\operatorname{argmin}} f(y; x) \text{ subject to } y \in C(x)$$

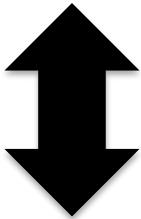


# The ReLU is a convex optimization layer

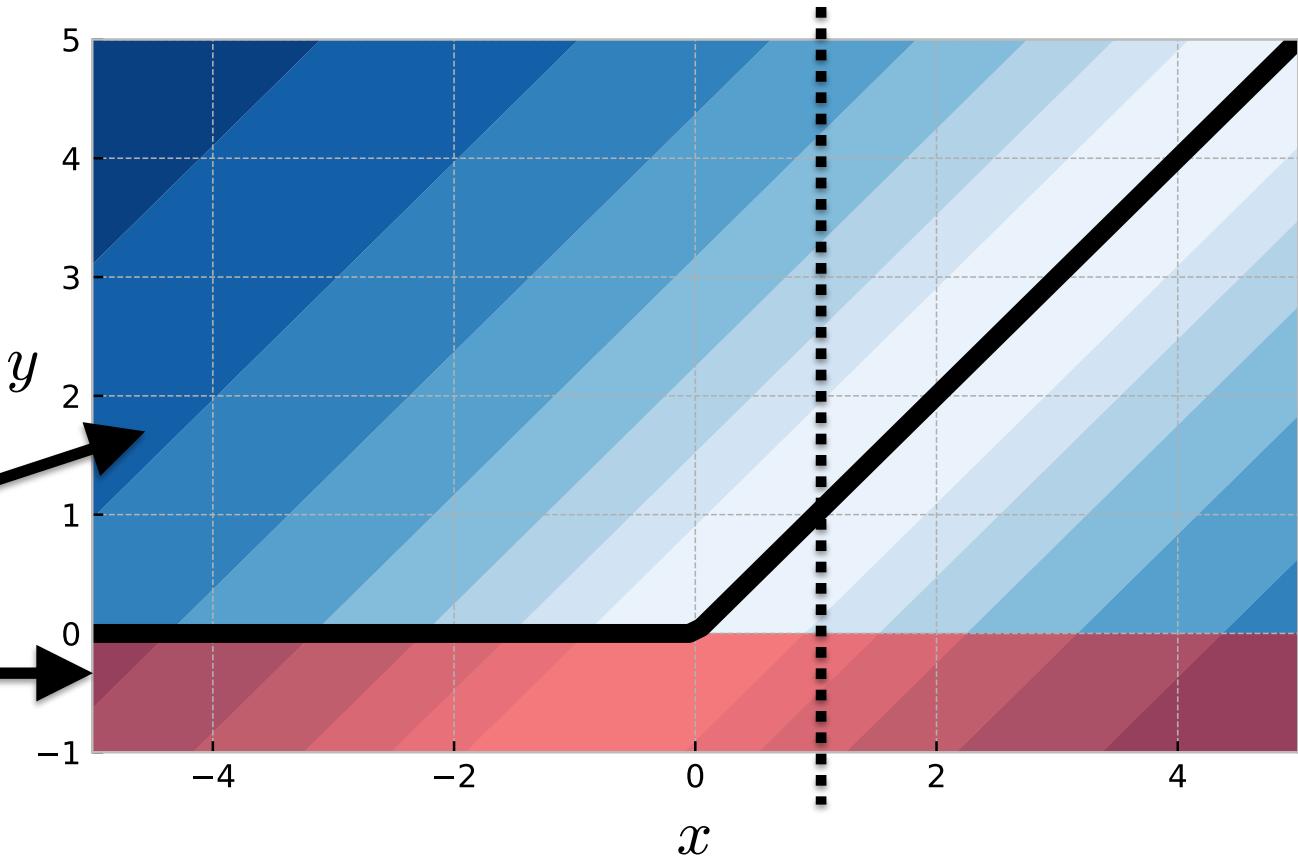
Section 2 of Differentiable optimization-based modeling for machine learning. Amos, PhD Thesis 2019

**Proof:** Comes from first-order optimality

$$\text{ReLU}(x) = \max\{0, x\}$$



$$\begin{aligned} \text{ReLU}(x) &= \operatorname{argmin}_y \|y - x\|_2^2 \\ \text{s.t. } y &\geq 0 \end{aligned}$$

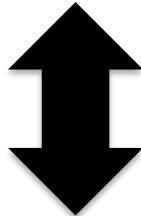


# The sigmoid is a convex optimization layer

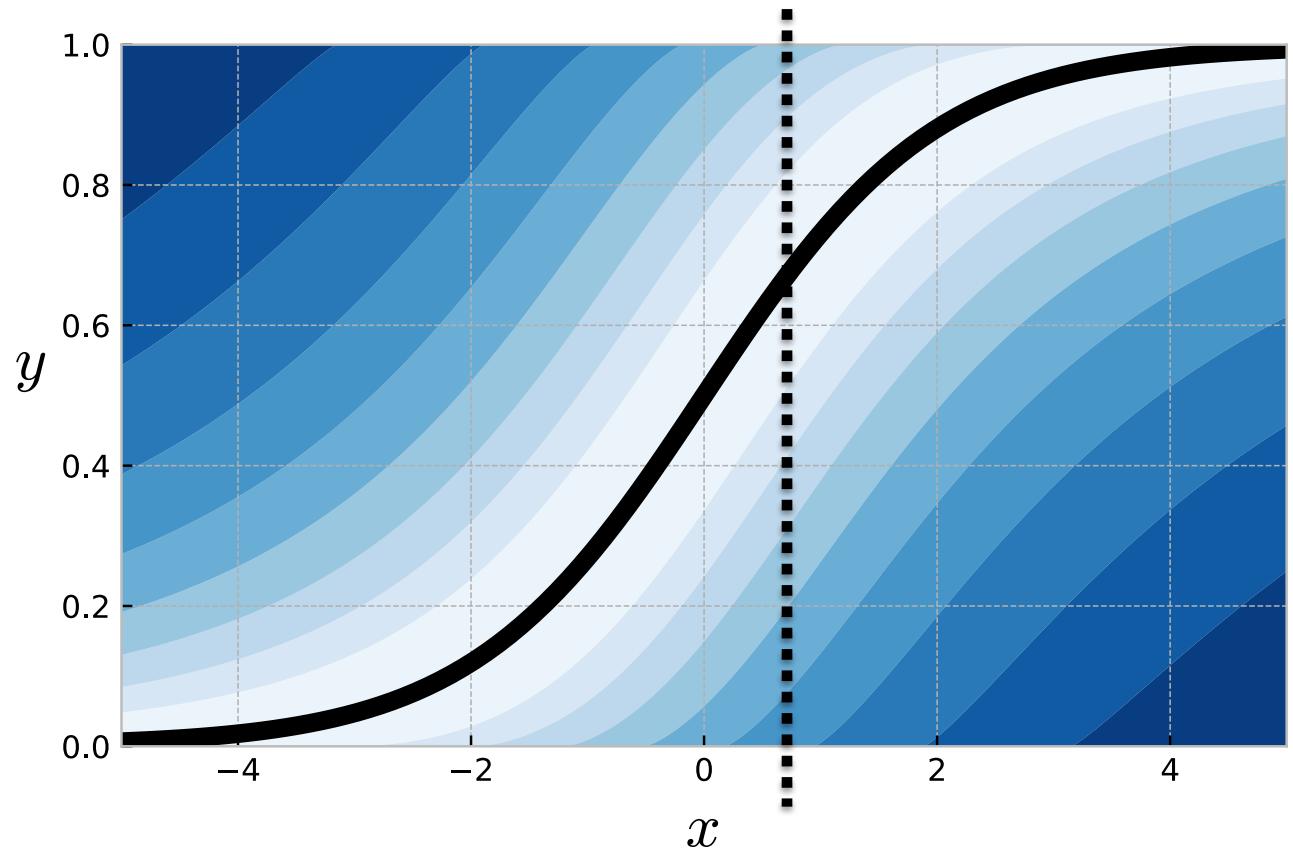
Section 2 of Differentiable optimization-based modeling for machine learning. Amos, PhD Thesis 2019

**Proof:** Comes from first-order optimality

$$\sigma(x) = \frac{1}{1 + \exp \{-x\}}$$



$$\begin{aligned} \sigma(x) = \operatorname{argmin}_y & -y^T x - H_b(y) \\ \text{s.t. } & 0 \leq y \leq 1 \end{aligned}$$



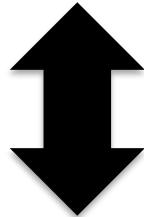
$H_b(y) := -\sum_i(y_i \log y_i + (1 - y_i) \log(1 - y_i))$  is the binary cross-entropy function

# The softargmax is a convex optimization layer

Section 2 of *Differentiable optimization-based modeling for machine learning*. Amos, PhD Thesis 2019

**Proof:** Comes from first-order optimality

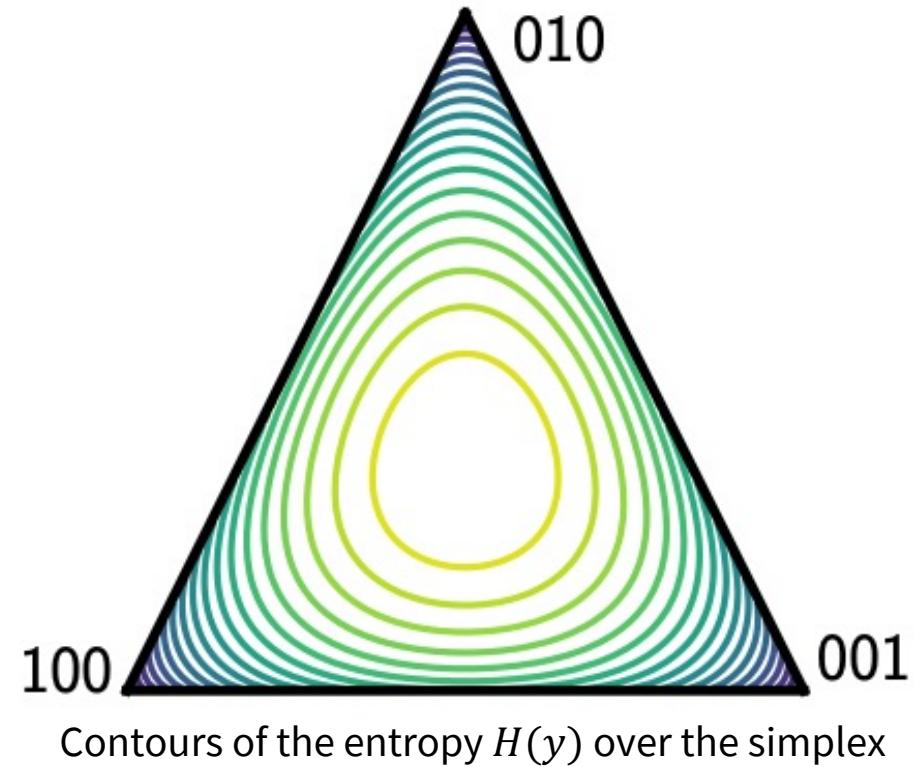
$$\pi_{\Delta}(x) = \frac{\exp x}{\sum_i \exp x_i}$$



$$\pi_{\Delta}(x) = \operatorname{argmin}_y -y^T x - H(y)$$

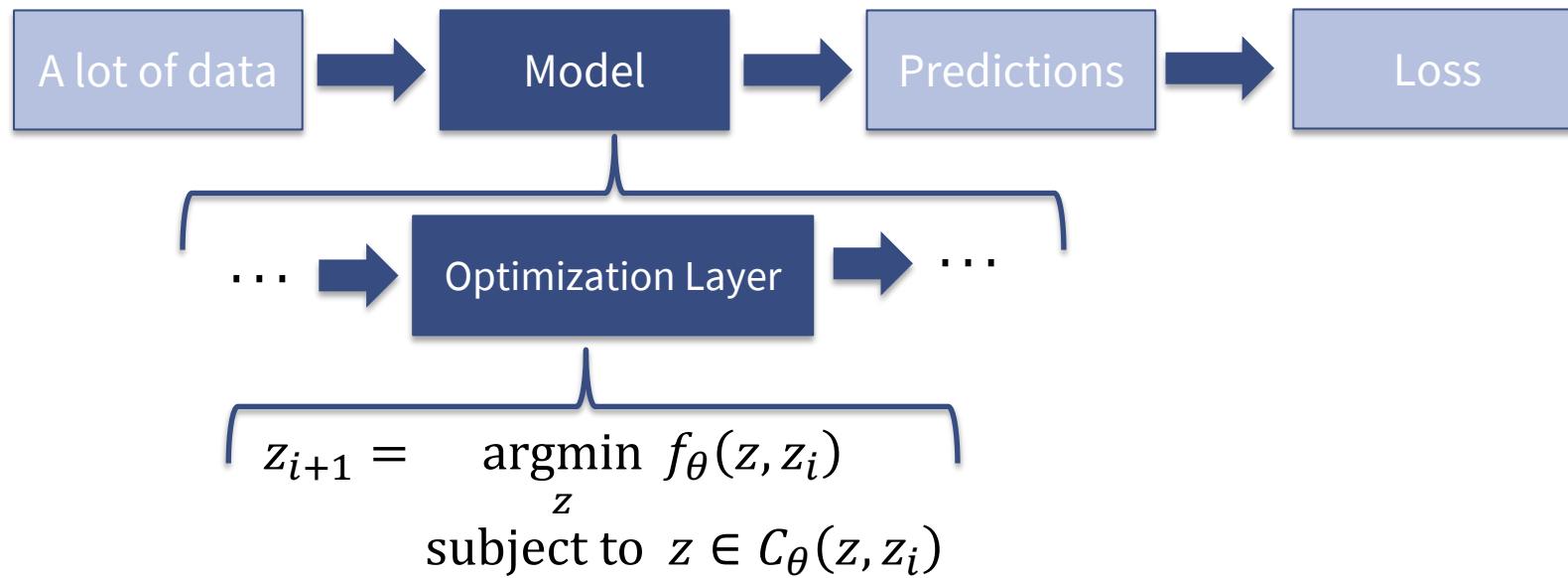
$$\begin{aligned} \text{s.t. } & 0 \leq y \leq 1 \\ & 1^T y = 1 \end{aligned}$$

$H(y) := -\sum_i y_i \log y_i$  is the entropy function



Contours of the entropy  $H(y)$  over the simplex

# How can we generalize this?



## Derivatives and backpropagation

For learning, we **differentiate** or backpropagate through these layers — **differentiable optimization**

Easy if the optimization problem has an **explicit, closed-form solution** (often standard differentiation)

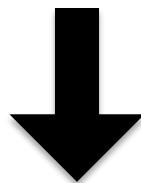
Otherwise, need to use **implicit differentiation**, which is also used for sensitivity analysis

# Differentiable convex quadratic programs

*OptNet: Differentiable Optimization as a Layer in Neural Networks.* Amos and Kolter, ICML 2017.

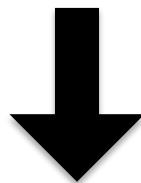
$$x^* = \underset{x}{\operatorname{argmin}} \frac{1}{2} x^\top Q x + p^\top x$$

subject to  $Ax = b$     $Gx \leq h$



## KKT Optimality

Find  $z^*$  s.t.  $\mathcal{R}(z^*, \theta) = 0$  where  $z^* = [x^*, \dots]$  and  $\theta = \{Q, p, A, b, G, h\}$



**Implicitly differentiating**  $\mathcal{R}$  gives  $D_\theta(z^*) = -\left(D_z \mathcal{R}(z^*)\right)^{-1} D_\theta \mathcal{R}(z^*)$

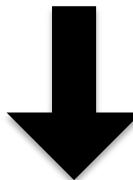
# Differentiable convex conic programs

Section 7 of Differentiable optimization-based modeling for machine learning. Amos, PhD Thesis 2019

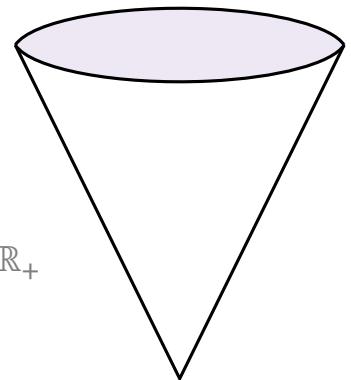
Differentiating through a cone program. Agrawal et al., 2019

Differentiable convex optimization layers. Agrawal\*, Amos\*, Barratt\*, Boyd\*, Diamond\*, Kolter\*, NeurIPS 2019.

$$x^* = \underset{x}{\operatorname{argmin}} \quad c^T x \\ \text{subject to } b - Ax \in \mathcal{K}$$

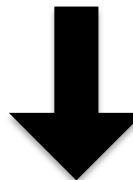


- Zero:  $\{0\}$
- Free:  $\mathbb{R}^n$
- Non-negative:  $\mathbb{R}_+^n$
- Second-order (Lorentz):  $\{(t, x) \in \mathbb{R}_+ \times \mathbb{R}^n \mid \|x\|_2 \leq t\}$
- Semidefinite:  $\mathbb{S}_+^n$
- Exponential:  $\{(x, y, z) \in \mathbb{R}^3 \mid ye^{x/y} \leq z, y > 0\} \cup \mathbb{R}_- \times \{0\} \times \mathbb{R}_+$
- Cartesian Products:  $\mathcal{K} = \mathcal{K}_1 \times \dots \times \mathcal{K}_p$



## Conic Optimality

Find  $z^*$  s.t.  $\mathcal{R}(z^*, \theta) = 0$  where  $z^* = [x^*, \dots]$  and  $\theta = \{A, b, c\}$



**Implicitly differentiating**  $\mathcal{R}$  gives  $D_\theta(z^*) = -(D_z \mathcal{R}(z^*))^{-1} D_\theta \mathcal{R}(z^*)$

# From the softmax to soft and differentiable top-k

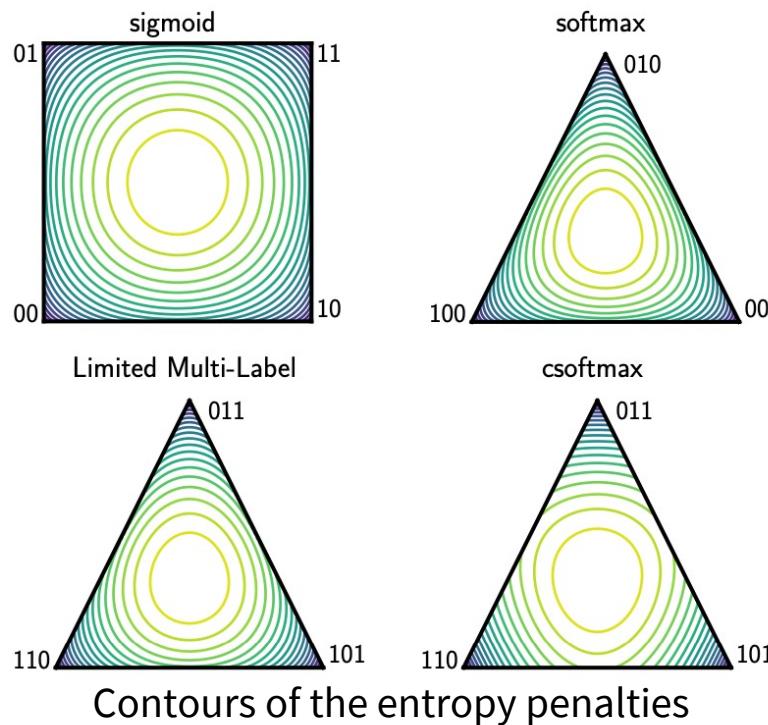
*Limited multi-label projection layer.* Amos et al., 2019.

$$y^* = \underset{y}{\operatorname{argmin}} -y^\top x - H(y)$$

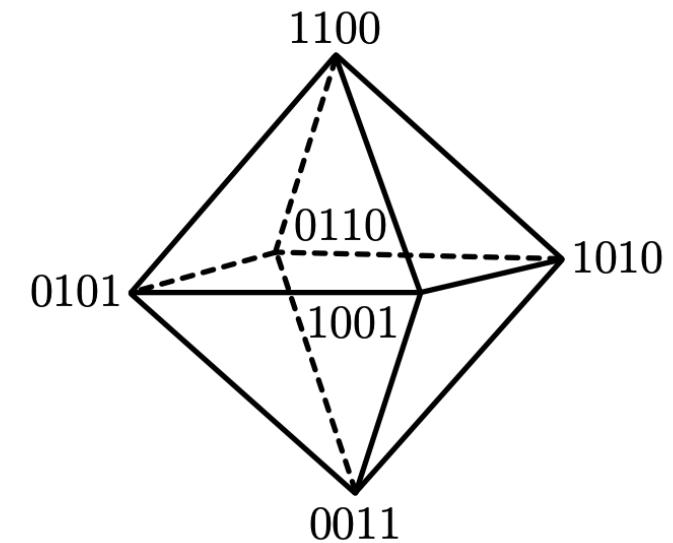
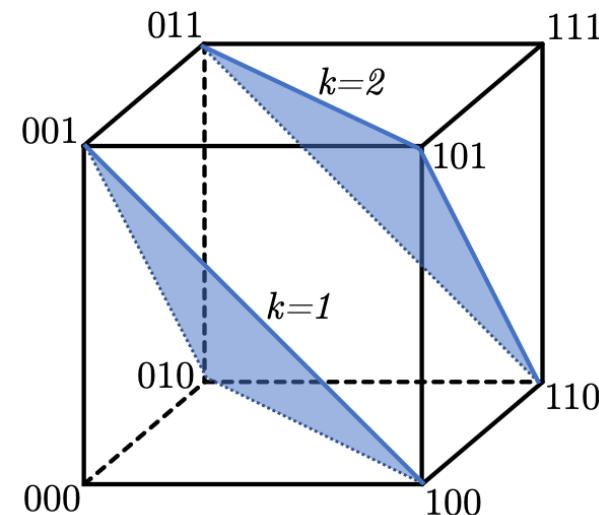
subject to  $0 \leq y \leq 1$

$$1^\top y = 1$$

$$\begin{aligned} y^* = \operatorname*{argmin}_y \quad & -y^\top x - H_b(y) \\ \text{subject to} \quad & 0 \leq y \leq 1 \\ & 1^\top y = k \end{aligned}$$



$H_b(y) \coloneqq -\sum_i(y_i \log y_i + (1 - y_i) \log(1 - y_i))$  is the binary cross-entropy function



# Differentiable permutations, sorting and SVMs

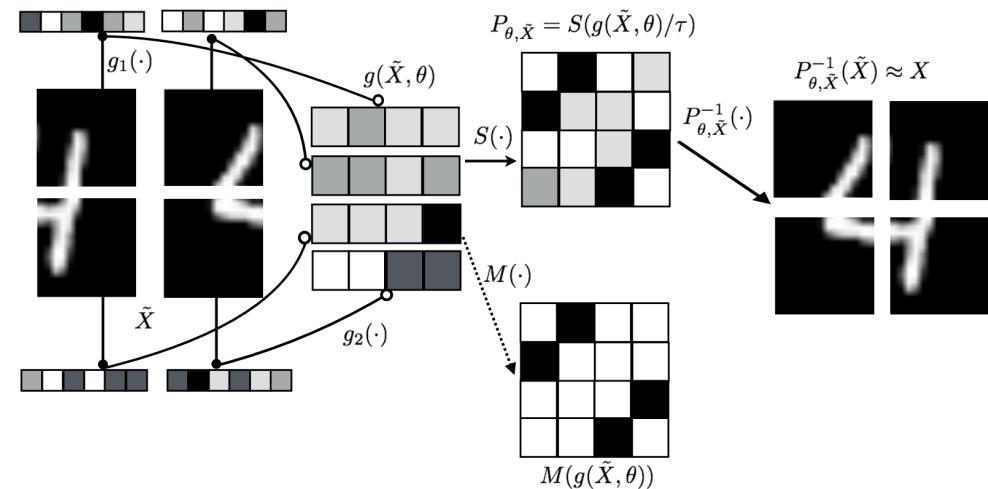
## Differentiable permutations and sorting (Gumbel-Sinkhorn)

Projection onto the **Birkhoff polytope**  $\mathcal{B}_N$ :

$$S\left(\frac{X}{\tau}\right) = \underset{P \in \mathcal{B}_N}{\operatorname{argmax}} \langle P, X \rangle_F + \tau H(P)$$

$$\mathcal{B}_N = \{X: X \geq 0, \sum_i X_{ij} = \sum_j X_{ij} = 1\}$$

*Learning latent permutations with Gumbel-Sinkhorn networks.* Mena et al., ICLR 2018.

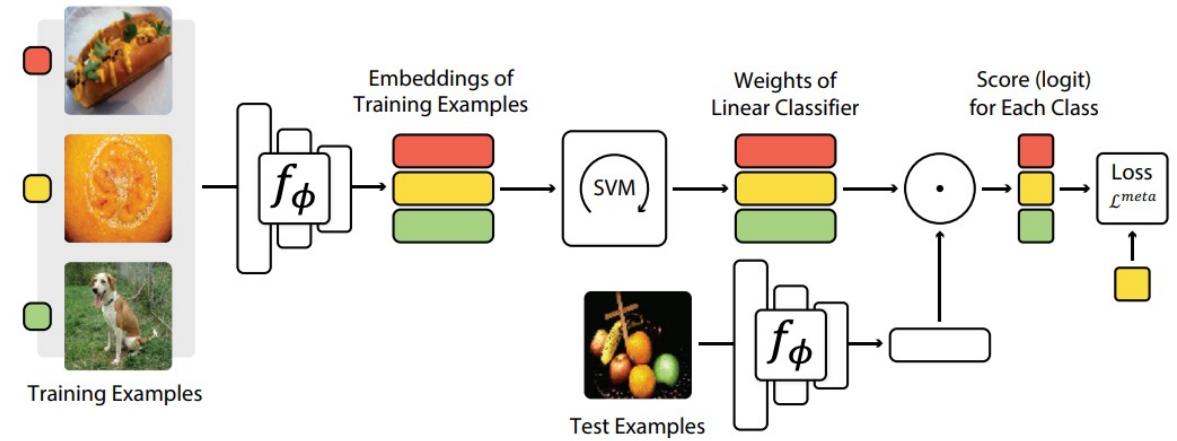


## Differentiable SVMs (MetaOptNet)

Differentiate the decision boundary w.r.t. the dataset

$$w^*(\mathcal{D}) = \underset{w}{\operatorname{argmin}} \|w\|^2 + C \sum_i \max\{0, 1 - y_i f(x_i)\}$$

*Meta-learning with differentiable convex optimization.* Lee et al., CVPR 2019.



# Sensitivity and perturbation analysis

*Differentiable convex optimization layers.* Agrawal\*, Amos\*, Barratt\*, Boyd\*, Diamond\*, Kolter\*, NeurIPS 2019.

**Adjoint derivatives** for optimization problems have been studied for decades  
We have focused on uses for learning, but also widely used for **sensitivity analysis**

## Logistic regression example

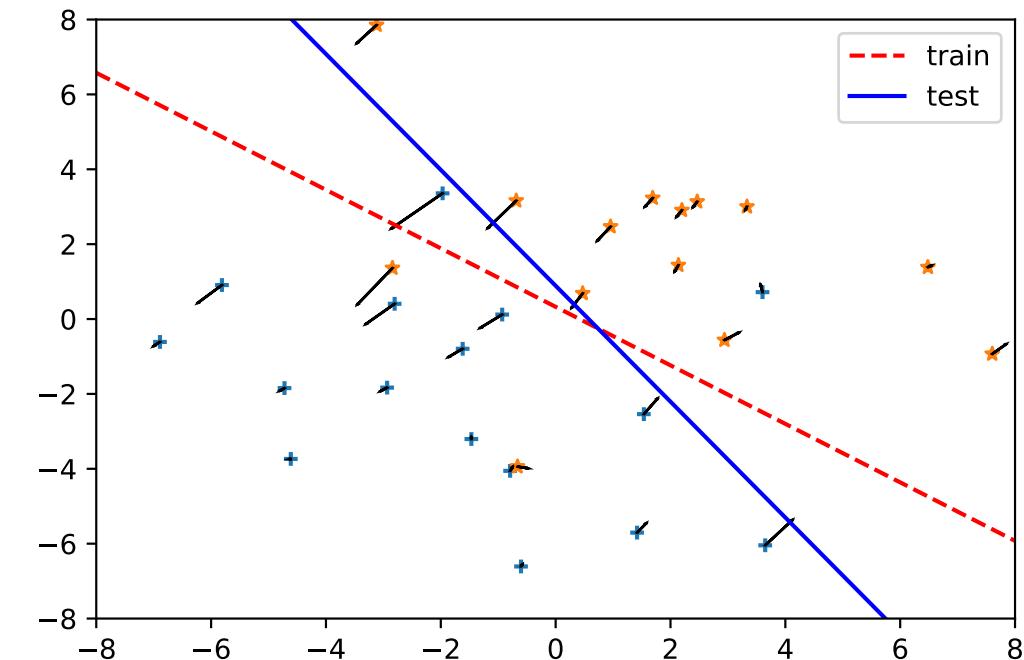
Find optimal decision boundary:

$$\theta^*(\mathcal{D}) \in \operatorname{argmax}_{\theta} \sum_i \log p_{\theta}(y_i | x_i)$$

$\theta^*(\mathcal{D})$  is **just a function** of the data

Derivatives  $\frac{\partial \theta^*}{\partial x_i}$  provide **sensitivity** to the data points

- *Show how much the data impacts the decision boundary*



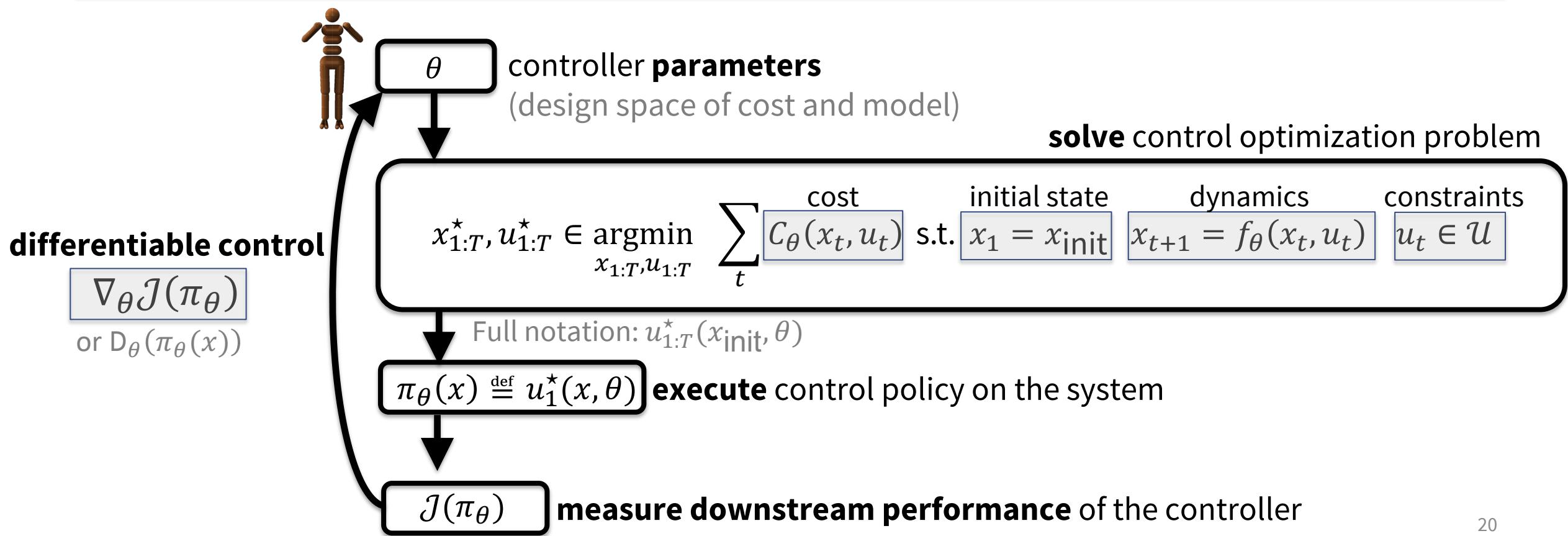
# Differentiable control

Differentiable MPC for end-to-end planning and control. Amos et al., NeurIPS 2018.

The differentiable cross-entropy method. Amos and Yarats, ICML 2020.

We can often measure the **downstream performance** induced by the controller

**Idea:** optimize (i.e., tune/learn) the parameters for a **downstream performance metric** by **differentiating through the control optimization problem**



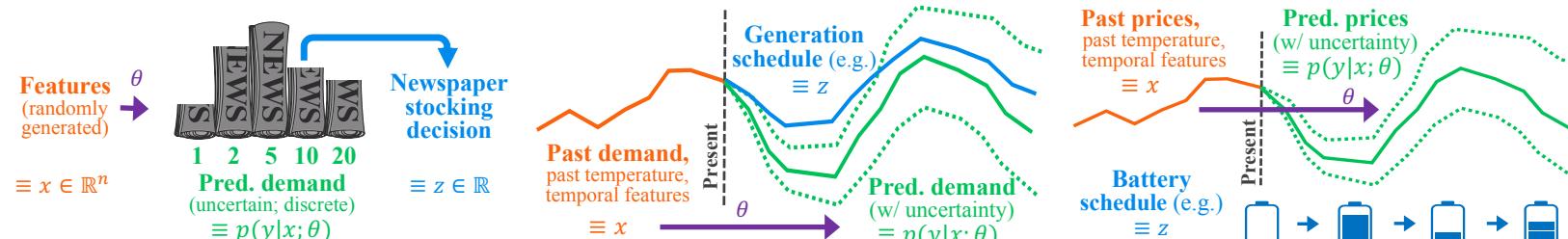
# Optimization as a loss for downstream tasks

Task-based end-to-end model learning in stochastic optimization. Donti, Amos, and Kolter, NeurIPS 2017.

A model's predictions may be used in a **downstream optimization problem** (e.g., scheduling)

**Challenge:** models are **inaccurate** and **not aware of how errors impact the downstream problem**

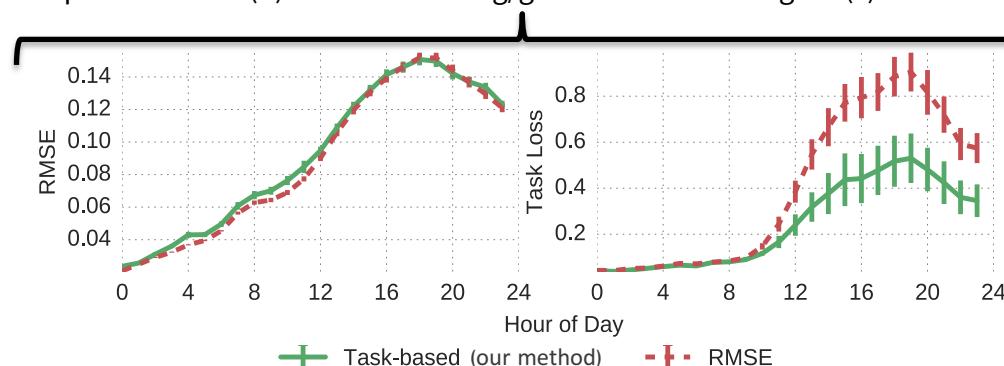
**Idea:** use the **downstream optimizer as a loss** for the model using differentiable optimization



(a) Inventory stock problem

(b) Load forecasting/generator scheduling

(c) Price forecasting/battery arbitrage



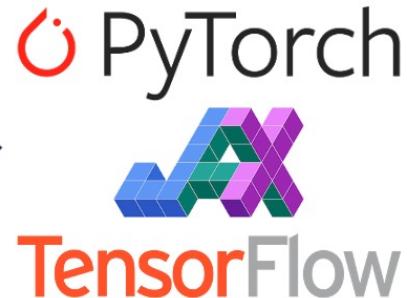
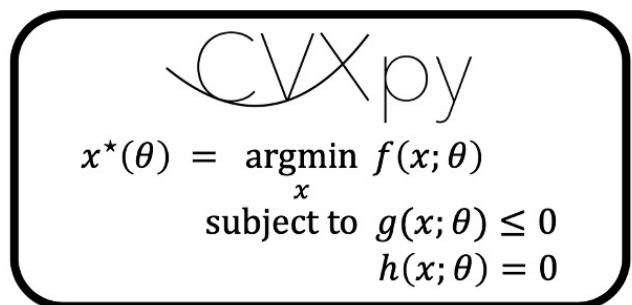
$$\underset{z \in \mathbb{R}^{24}}{\text{minimize}} \mathbf{E}_y \left[ \sum_{i=1}^{24} \gamma_s [y_i - z_i]_+ + \gamma_e [z_i - y_i]_+ + \frac{1}{2} \|z_i - y_i\|^2 \right] \quad \text{subject to } |z_i - z_{i+1}| \leq c_{\text{ramp}}.$$

# Implementing differentiable optimization

**Manually:** unroll or implicitly differentiate the optimality conditions (such as the KKT system)  
My original differentiable QP implementation was **~1k lines of code** to make efficient

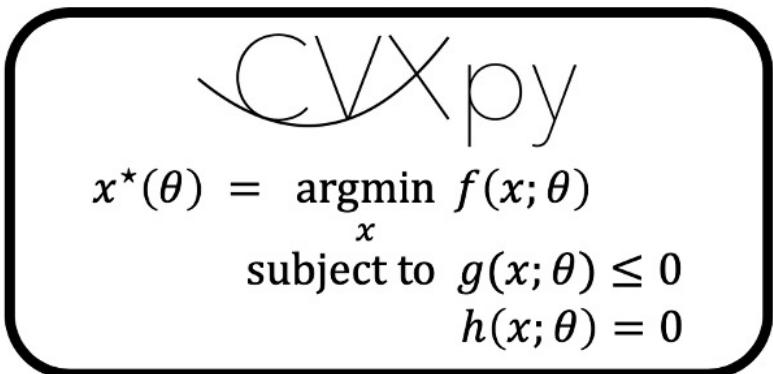
**Use a software package** to prototype such as:

- **cvxpylayers**: differentiable convex optimization in CVXPY [Agrawal\*, Amos\*, et al., NeurIPS 2019]  
**Implements a differentiable QP in ~10 lines of code**
- **Theseus**: differentiable non-linear least squares with a focus on robotics [Pineda et al., NeurIPS 2022]
- **higher**: differentiable unrolled optimizers [Grefenstette et al., 2019]

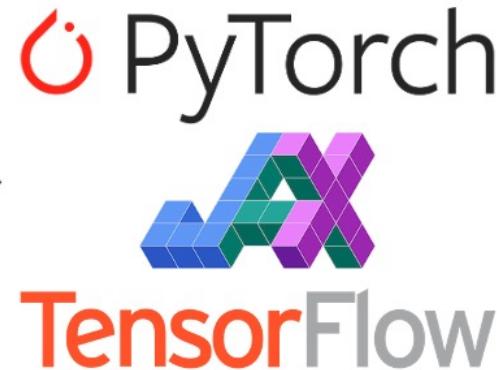


# Differentiable CVXPY

*Differentiable convex optimization layers.* Agrawal\*, Amos\*, Barratt\*, Boyd\*, Diamond\*, Kolter\*, NeurIPS 2019.



(Officially part of CVXPY!)



An equation block containing an optimization problem:

$$z_{i+1} = \underset{z}{\operatorname{argmin}} \frac{1}{2} z^\top Q(z_i)z + q(z_i)^\top z$$

subject to

$$A(z_i)z = b(z_i)$$
$$G(z_i)z \leq h(z_i)$$

**Parameters/Submodules :**  $Q, q, A, b, G, h$

**Before:** 1k lines of code, **now:**

```
import cvxpy as cp
from cvxpyth import CvxpyLayer
obj = cp.Minimize(0.5*cp.quad_form(x, Q) + p.T * x)
cons = [A*x == b, G*x <= h]
prob = cp.Problem(obj, cons)
layer = CvxpyLayer(prob, params=[Q, p, A, b, G, h], out=[x])
```

# This talk

**Differentiable optimization** —  $\frac{\partial}{\partial x} y^*(x)$

Foundations

Differentiable convex optimization layers

Differentiable control

Optimization-based task losses

**Amortized optimization** —  $\hat{y}_\theta(x) \approx y^*(x)$

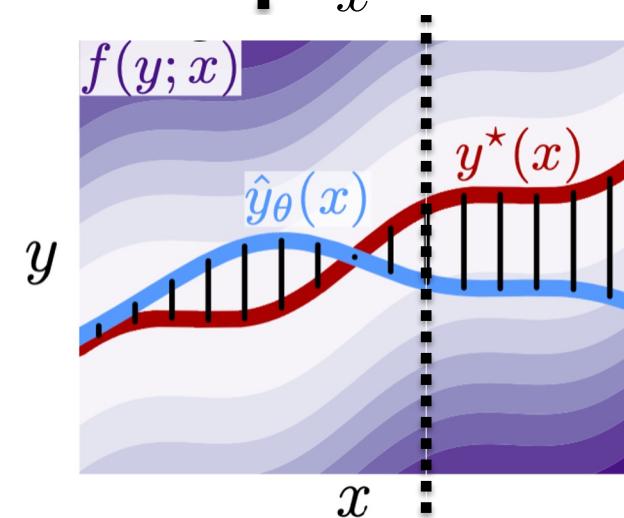
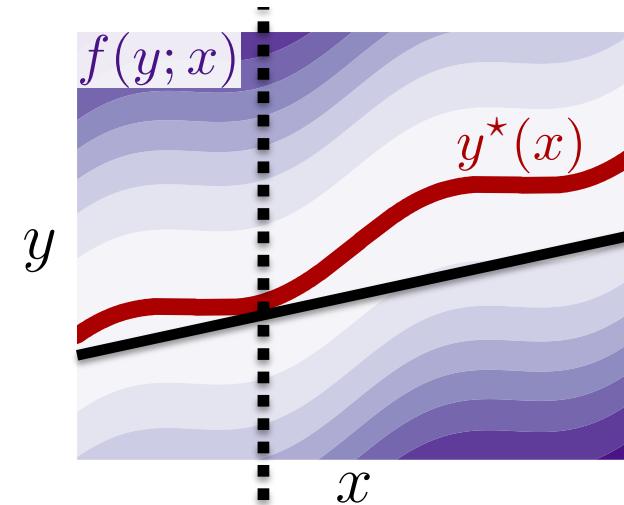
Foundations

RL as amortized optimization

Amortization via learning latent subspaces

Meta Optimal Transport

Amortizing convex conjugates



# Difficulty: optimization is computationally expensive

Sometimes solving just **once** may be difficult

Exasperated when **repeatedly solving**, e.g., as part of a differentiable optimization layer

# Insight: optimal solutions share structure

Optimization problems **share structure** and don't live in isolation

# Solution: amortized optimization

Use **machine learning** to uncover the shared structure

Create **learning-augmented** versions of classical optimization solvers

**Far surpasses** average or worst-case convergence rates

Also referred to as **learning to optimize**

# Amortized optimization

Tutorial on amortized optimization for learning to optimize over continuous domains. Amos, Foundations and Trends in Machine Learning (to appear)

## Two key components:

1. **Amortization model**  $\hat{y}_\theta(x)$  tries to approximate  $y^*(x)$

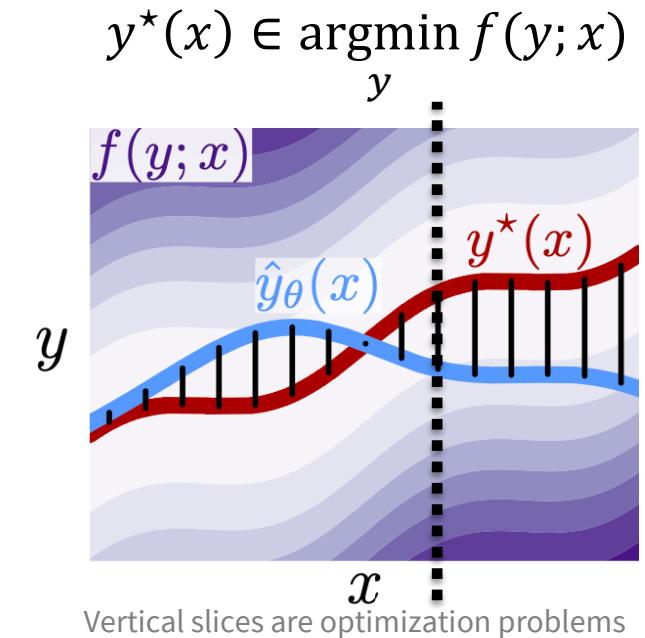
**Fully-amortized:** A neural network mapping from  $x$  to the solution

**Semi-amortized:** Allowed to query the objective  $f$

2. **Loss**  $\mathcal{L}$  measures how well  $\hat{y}$  fits  $y^*$  and optimized with  $\min_{\theta} \mathcal{L}(\hat{y}_\theta)$

**Regression:**  $\mathcal{L}_{\text{reg}}(\hat{y}_\theta) := \mathbb{E}_{p(x)} \|\hat{y}_\theta(x) - y^*(x)\|_2^2$

**Objective-based:**  $\mathcal{L}_{\text{obj}}(\hat{y}_\theta) := \mathbb{E}_{p(x)} f(\hat{y}_\theta(x); x)$



**Amortization is widely deployed** (although often not connected as amortization)

Amortized variational inference (VAEs)

Meta-learning (hypernetworks, MAML)

Reinforcement learning (policy learning for actor-critic methods, SAC)

# RL policy learning is amortized optimization

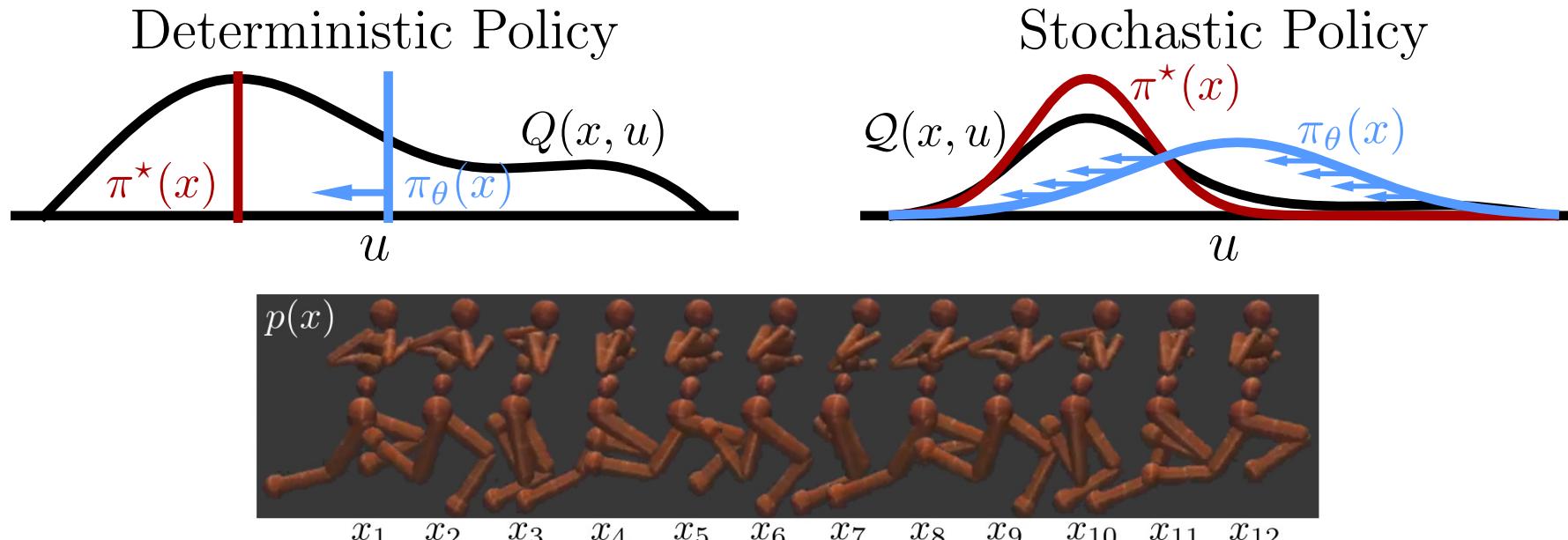
Tutorial on amortized optimization for learning to optimize over continuous domains. Amos, Foundations and Trends in Machine Learning (to appear)  
On the model-based stochastic value gradient for continuous reinforcement learning. Amos et al., L4DC 2021.

**Setup:** controlling a **continuous MDP** with a **model-free policy**  $\pi_\theta(x)$

**Review:** Learning a policy with a **value gradient** amortizes over the  $Q$ -value:

$$\operatorname{argmax}_\theta \mathbb{E}_{p(x)} Q(x, \pi_\theta(x))$$

The **amortization perspective** easily enables **expanding beyond this fully-amortized setting**



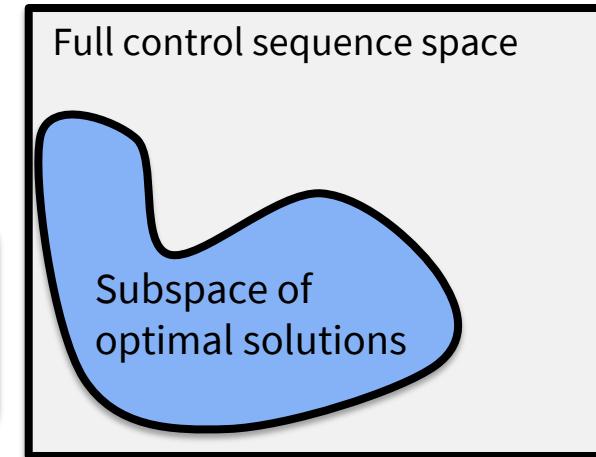
# Amortization via learning latent subspaces

*The differentiable cross-entropy method.* Amos and Yarats, ICML 2020.

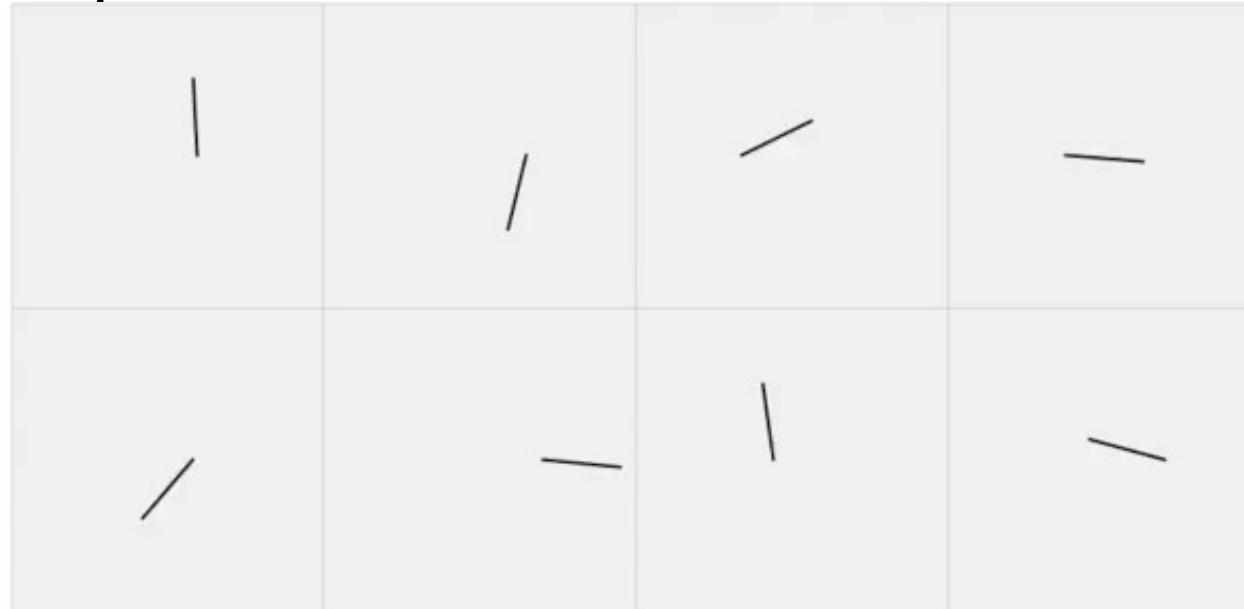
**Amortize the problem by learning a latent subspace of optimal solutions**

**Only search over optimal solutions** rather than the entire space

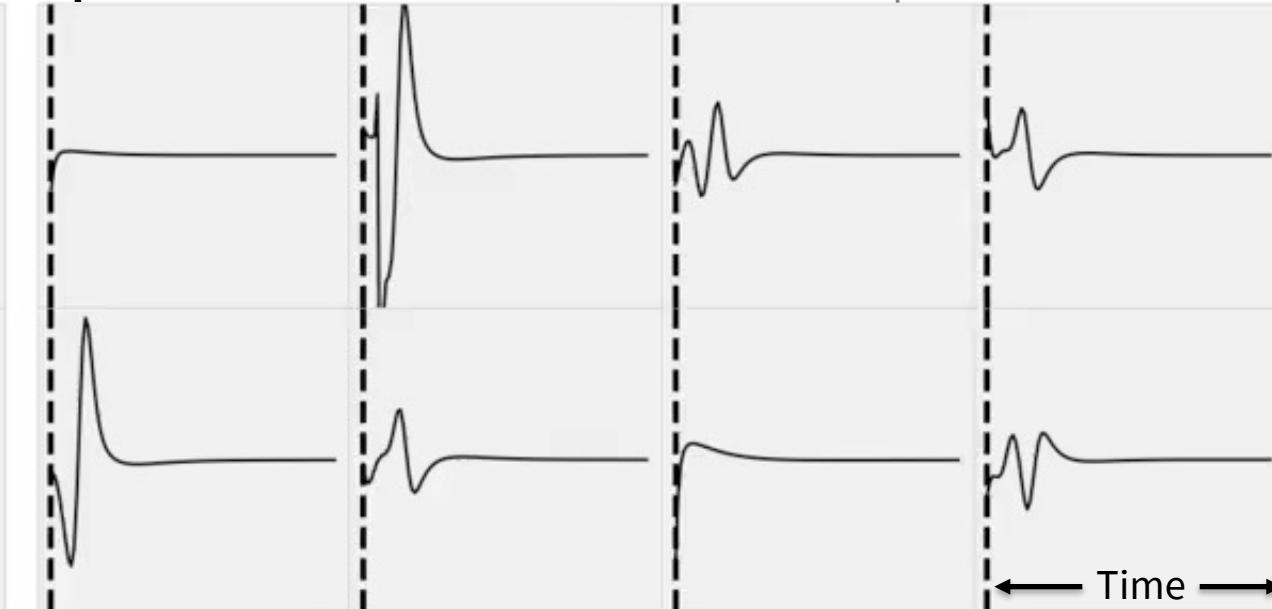
$$x_{1:T}^*, u_{1:T}^* \in \operatorname{argmin}_{x_{1:T}, u_{1:T}} \sum_t \text{cost } C_\theta(x_t, u_t) \text{ s.t. } \begin{array}{l} \text{initial state } x_1 = x_{\text{init}} \\ \text{dynamics } x_{t+1} = f_\theta(x_t, u_t) \\ \text{constraints } u_t \in \mathcal{U} \end{array}$$



Cartpole videos



Optimal controls over time — force on the cartpole



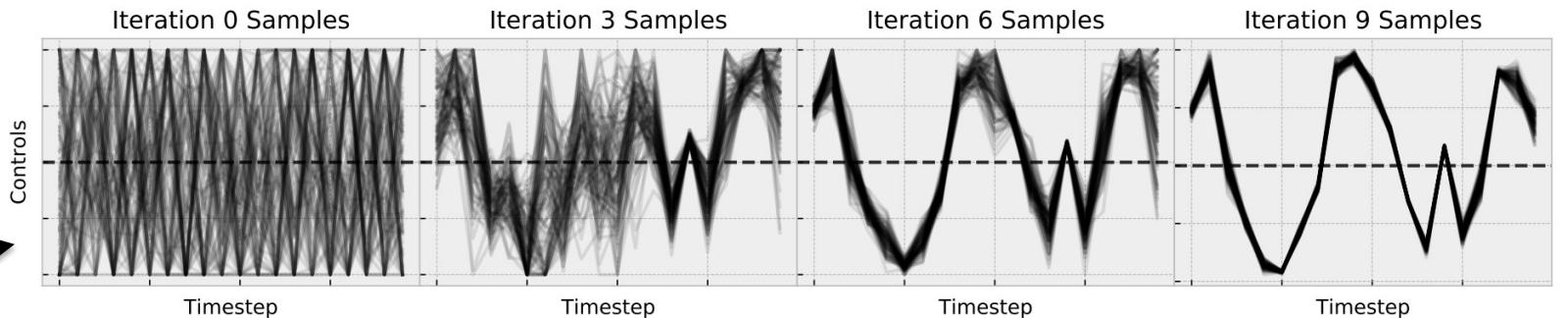
# Amortization via learning latent subspaces

The differentiable cross-entropy method. Amos and Yarats, ICML 2020.

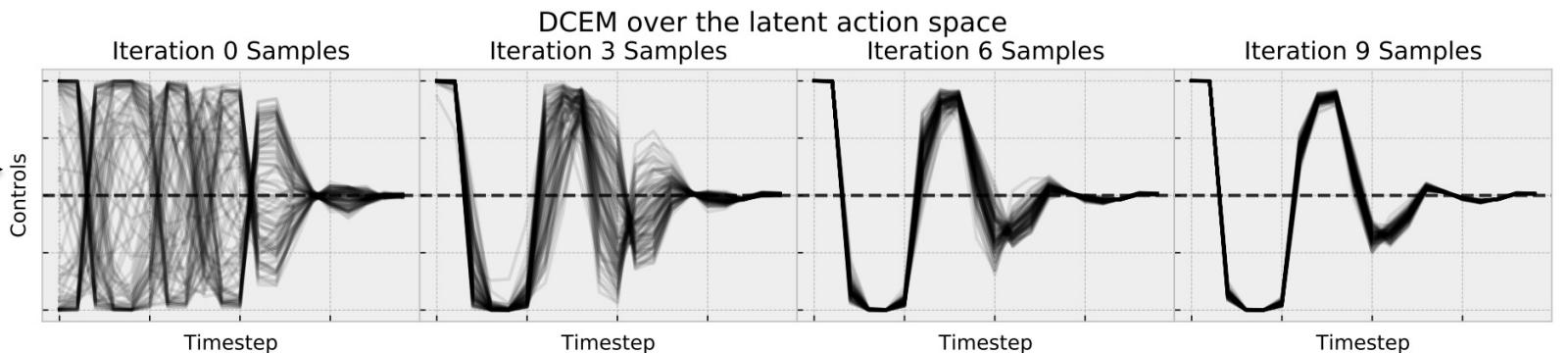
CEM over the full action space

$$u^* = \operatorname{argmin}_{u \in [0,1]^N} f(u)$$

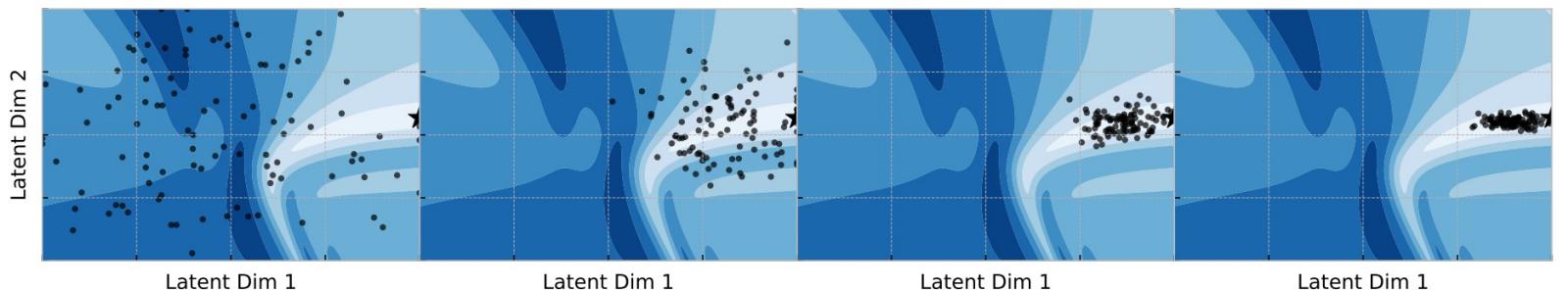
Full control sequence space



Subspace of optimal solutions



Latent space of optimal solutions



# Meta Optimal Transport

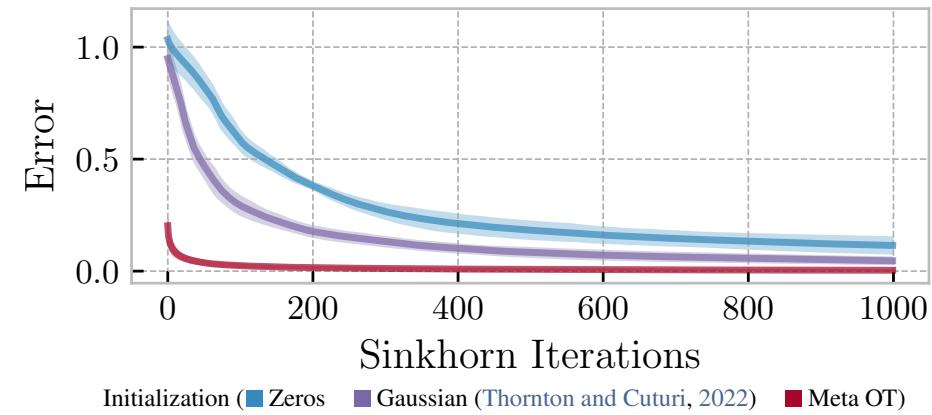
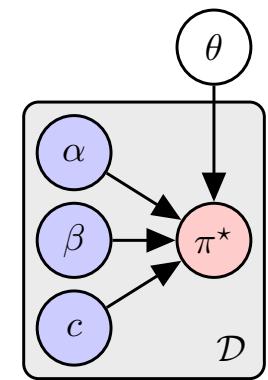
[Amos et al., 2022]

**Goal: optimally transport mass** between measures  $\alpha$  to  $\beta$

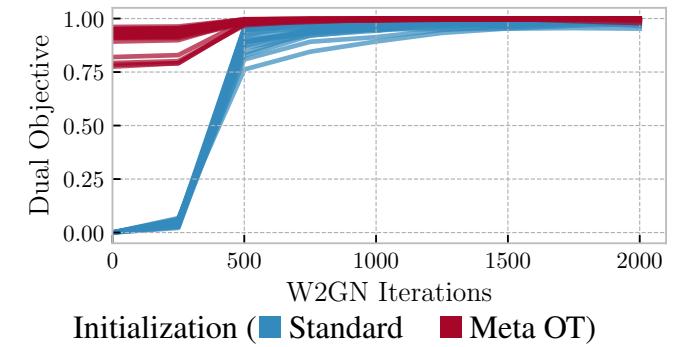
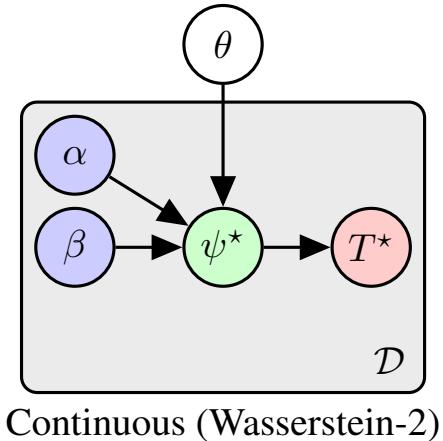
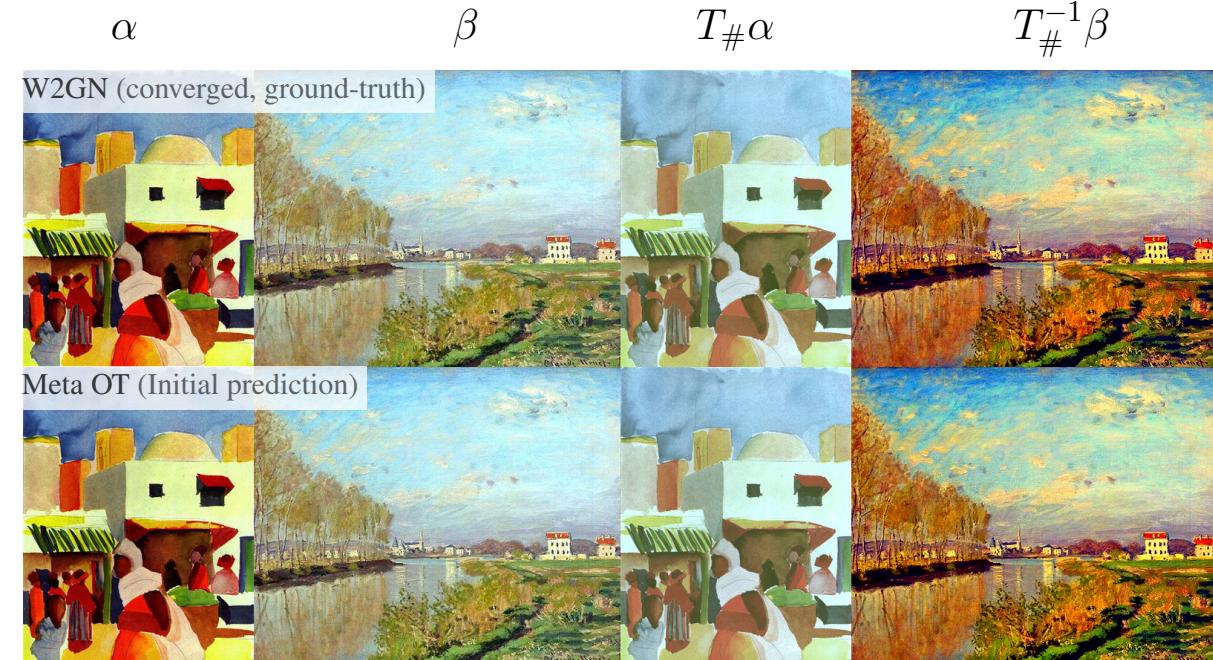
$$\pi^*(\alpha, \beta, c) \in \operatorname{argmin}_{\pi \in \mathcal{U}(\alpha, \beta)} \int_{X \times Y} c(x, y) d\pi(x, y)$$

Use **amortization** when **repeatedly coupling measures**  
e.g., between pairs of images or physical transport

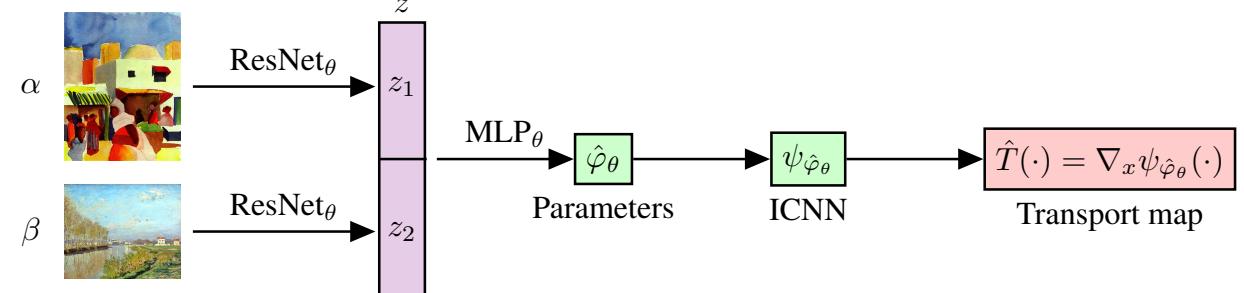
Often amortize over **unconstrained dual potentials**



# Meta OT for continuous OT with Meta ICNNs



**Meta ICNN:** Predict parameters of ICNN coupling  $\alpha$  and  $\beta$



# More Meta OT color transfer predictions

$\alpha$

$\beta$

$T_{\#}\alpha$

$T_{\#}^{-1}\beta$

$\alpha$

$\beta$

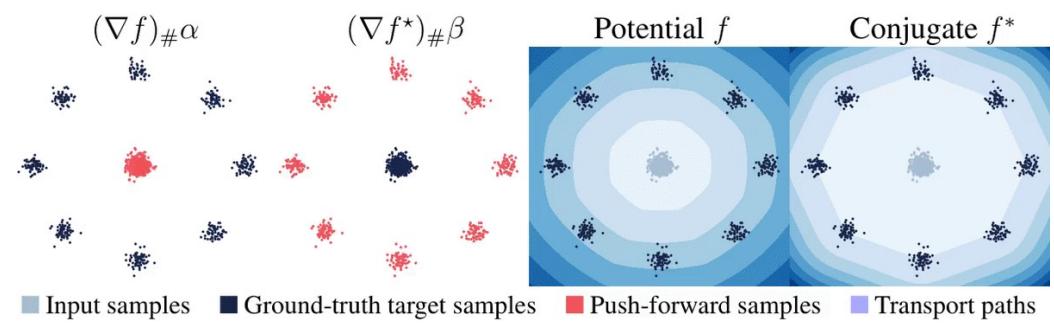
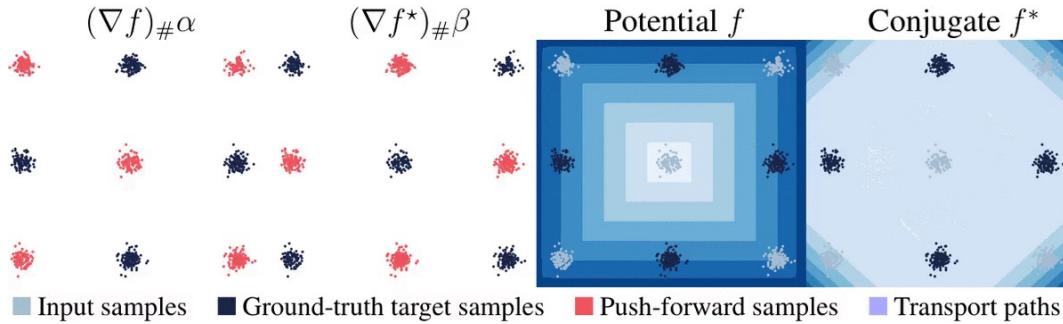
$T_{\#}\alpha$

$T_{\#}^{-1}\beta$



# On amortizing convex conjugates for OT

[Amos, 2022]



**Setting:** transport between measures  $\alpha$  and  $\beta$

**Monge problem** for Euclidean Wasserstein-2

$$T^*(\alpha, \beta) \in \operatorname{argmin}_{T \in \mathcal{C}(\alpha, \beta)} \mathbb{E}_{x \sim \alpha} \|x - T(x)\|_2^2$$

$$T^* = \nabla \hat{f} \text{ from Brenier's theorem}$$

$f^*(y) := - \inf_x f(x) - x^\top y$  is the convex conjugate

**Kantorovich dual**

$$\hat{f} \in \operatorname{argmax}_{f \in \mathcal{L}^1(\alpha)} - \mathbb{E}_{x \sim \alpha}[f(x)] - \mathbb{E}_{y \sim \beta}[f^*(y)]$$

**Challenge:** The **convex conjugate can be computationally expensive** to numerically estimate

**Idea:** Amortize and fine-tune the conjugate to rapidly estimate it

Improves Wasserstein-2 benchmark results by **~2-4x**

# Closing thoughts

Optimization expresses **non-trivial reasoning operations**

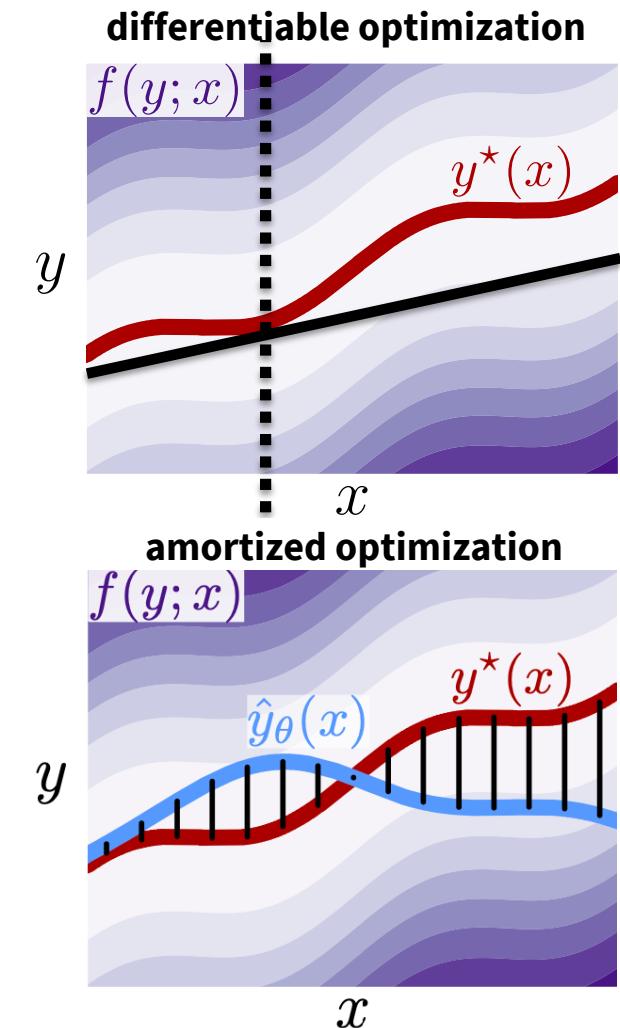
- Integrates nicely with machine learning by **seeing it as a function**

**Differentiable optimization** is a powerful primitive within larger systems

- Theoretical and engineering foundations are here
- Can be propagated through and learned, just like any layer
- Provides a perspective to analyze existing models and layers

**Amortized optimization** enables fast learning-augmented solvers

- Foundations are likewise here
- Provides a perspective on existing methods in RL, VI, OT



# Future directions

**Goal:** build intelligent systems that **understand and interact** with the world

**Why?** To advance scientific and engineering discoveries

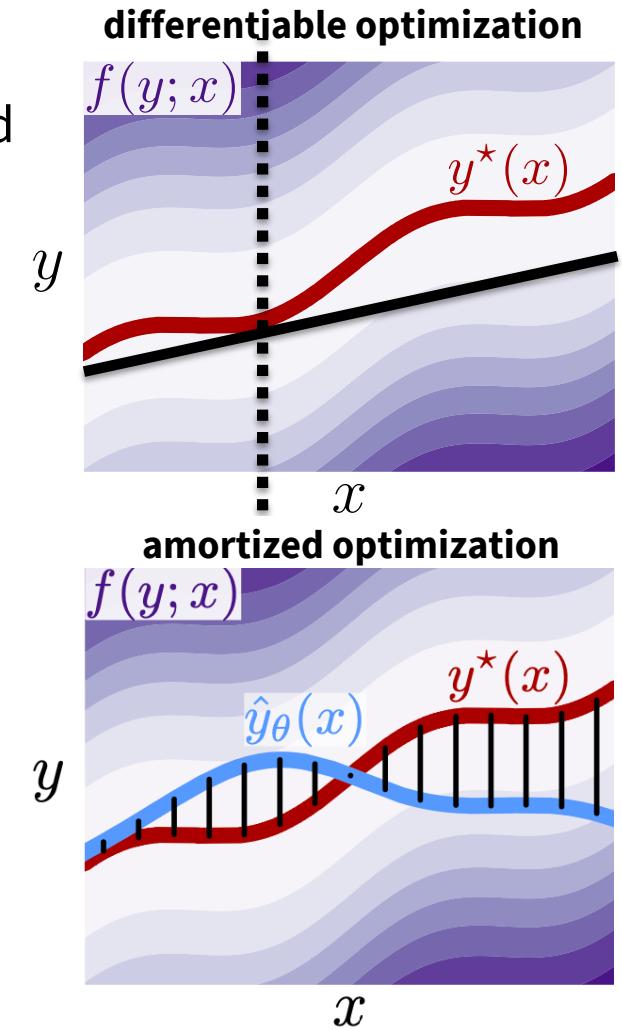
Advancing **optimization** and **machine learning foundations** is crucial

**Differentiable optimization** —  $\frac{\partial}{\partial x} y^*(x)$

- *Differentiable control: learning control-aware representations*
- *Numerics: “differentiation”, ill-behaved functions, approximations*
- *Generalizing to non-convex, non-Euclidean, and/or discrete settings*

**Amortized optimization** —  $\hat{y}_\theta(x) \approx y^*(x)$

- *Numerics: amortizing with distributions and Markov kernels*
- *Building general learning-augmented optimizers*
- *Amortization models that use differentiable optimization*



# Learning foundations beyond optimization

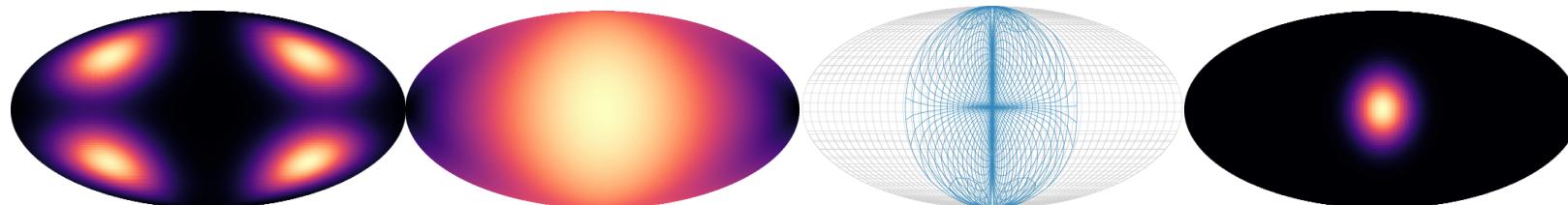
**Goal:** build intelligent systems that **understand and interact** with the world

**Why?** To advance scientific and engineering discoveries

Advancing **optimization** and **machine learning foundations** is crucial

**Next frontier** includes understanding, mapping between, controlling, modeling, and optimizing over:

- **Non-trivial physical systems and dynamics** (e.g. fluid dynamics, open-world robots)
- **High-dimensional** image, video, and text spaces
- **Discrete structures** arising from decisions (e.g. the knapsack) and other natural structures
- **Riemannian geometries** expressing 3D objects and many other mathematical spaces



*Riemannian convex potential maps. Cohen\*, Amos\*, and Lipman, ICML 2021.*

# Learning with differentiable and amortized optimization

Brandon Amos • Meta AI (FAIR) NYC



<http://github.com/bamos/presentations>

[ICML 2017] [Differentiable QPs: OptNet](#)

[NeurIPS 2017] [Differentiable Task-based Model Learning](#)

[NeurIPS 2018] [Differentiable MPC for End-to-end Planning and Control](#)

[NeurIPS 2019] [Differentiable Convex Optimization Layers](#)

[Ph.D. Thesis 2019] [Differentiable Optimization-Based Modeling for ML](#)

[arXiv 2019] [Differentiable Top-k and Multi-Label Projection](#)

[arXiv 2019] [Generalized Inner Loop Meta-Learning](#)

[ICML 2020] [Differentiable Cross-Entropy Method](#)

[ICML 2021] [Differentiable Combinatorial Optimization: CombOptNet](#)

[L4DC 2021] [On the model-based stochastic value gradient](#)

[NeurIPS 2022] [Theseus: Differentiable Nonlinear Optimization](#)

[arXiv 2022] [Meta Optimal Transport](#)

[arXiv 2022] [On amortizing convex conjugates for optimal transport](#)

[Foundations and Trends in ML, to appear] [Tutorial on amortized optimization](#)

Collaborators: Akshay Agrawal, Alexander Rives, Andrew Gordon Wilson, Anselm Paulus, Arnaud Fickinger, Artem Molchanov, Austin Wang, Byron Boots, Caroline Chen, Daniel DeTone, Denis Yarats, Douwe Kiela, Edward Grefenstette, Franziska Meier, Georg Martius, Giulia Luise, Hengyuan Hu, Ievgen Redko, Ivan Jimenez, Jacob Sacks, Jason Liu, Jing Dong, Joseph Ortiz, Joshua Meier, Kyunghyun Cho, Luis Pineda, Maurizio Monge, Michal Rolínek, Mustafa Mukadam, Nathan Lambert, Noam Brown, Omry Yadan, Paloma Sodhi, Phu Mon Htut, Priya Donti, Ricky Chen, Robert Verkuil, Roberto Calandra, Samuel Cohen, Samuel Stanton, Shane Barratt, Shobha Venkataraman, Soumith Chintala, Stephen Boyd, Steven Diamond, Stuart Anderson, Stuart Russell, Taosha Fan, Tom Sercu, Vít Musil, Yann LeCun, Zeming Lin, Zico Kolter