Learning with differentiable and amortized optimization

Brandon Amos • Meta AI (FAIR) NYC

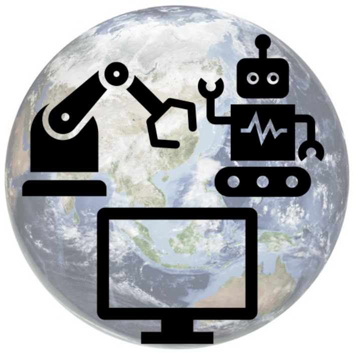
http://github.com/bamos/presentations

Optimization is crucial technology

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

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

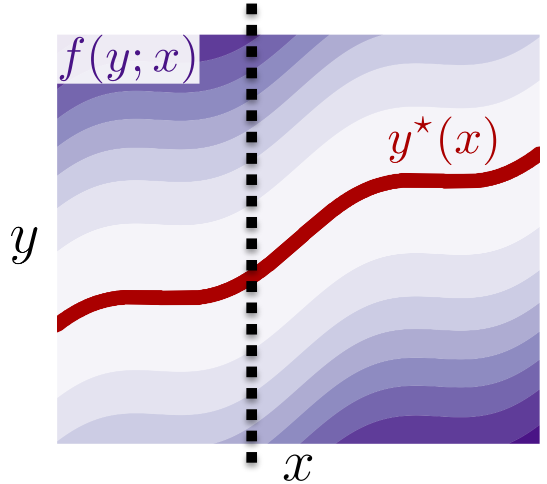
Focus: parametric optimization problems that are **repeatedly solved**



optimal solution — objective — context (or parameterization)

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

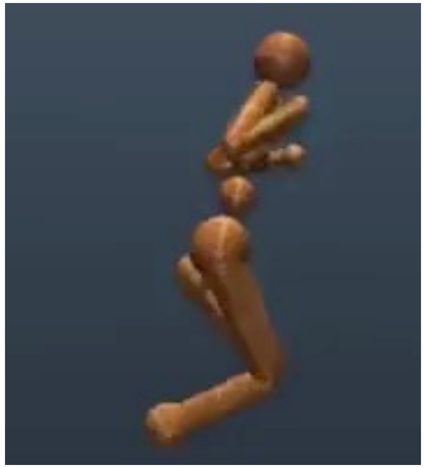
optimization variable — constraints

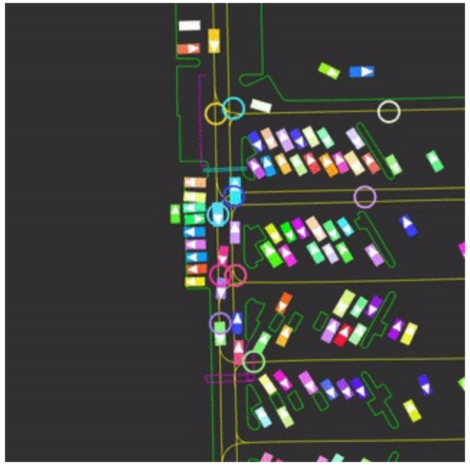


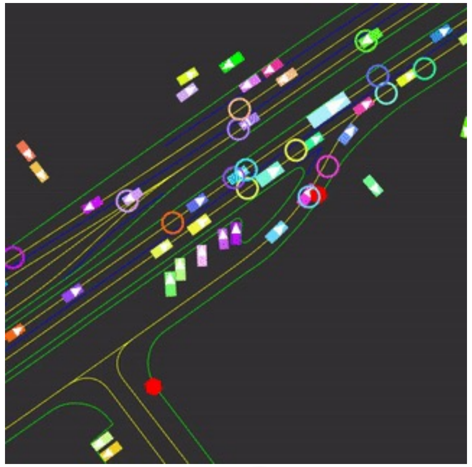
vertical slices are optimization problems

Breakthroughs enabled by optimization include

1. **controlling systems** (robotic, autonomous, mechanical, and multi-agent)



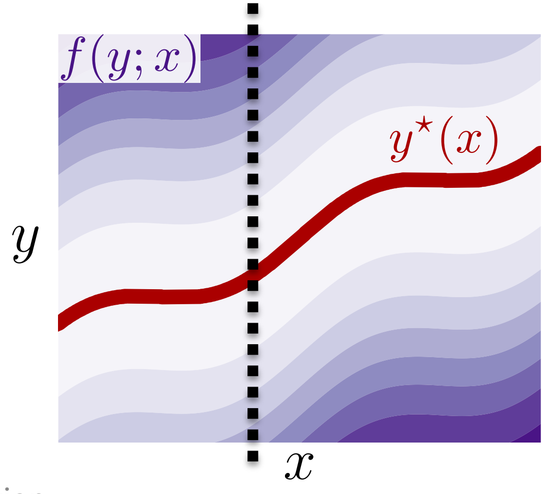




optimal solution — objective — context (or parameterization)

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

optimization variable — constraints



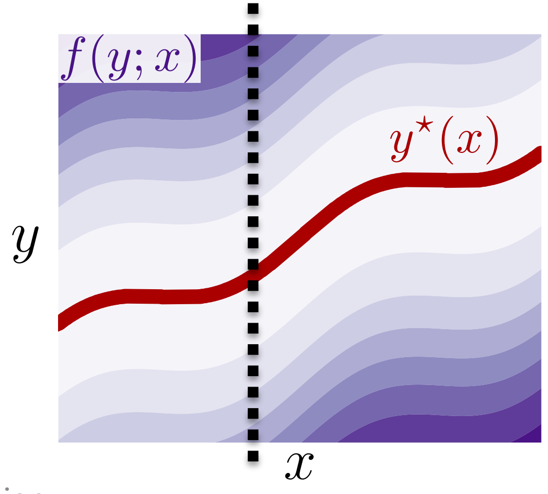
Breakthroughs enabled by optimization include

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
4. **allocating** budgets and portfolios
5. **designing** materials, molecules, and other structures
6. **solving inverse problems** (to infer underlying hidden costs, incentives, geometries, terrains)
7. **parameter learning** of predictive and statistical models

optimal solution   objective   context (or parameterization)

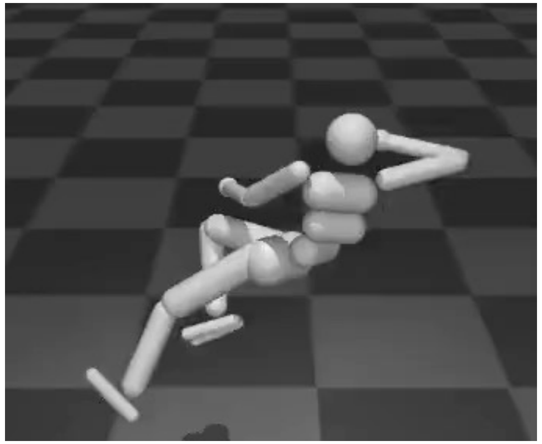
$$y^\star(x) \in \operatorname*{argmin}_{y \in \mathcal{C}(x)} f(y; x)$$

optimization variable   constraints



When optimization fails, machine learning helps

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



Bad representation of the world (unknown, mis-specified, or inaccurate)
Solving is computationally difficult

When machine learning fails, optimization helps

Optimization provides an **internal reasoning operation**

A lot of data → Model → Predictions → Loss

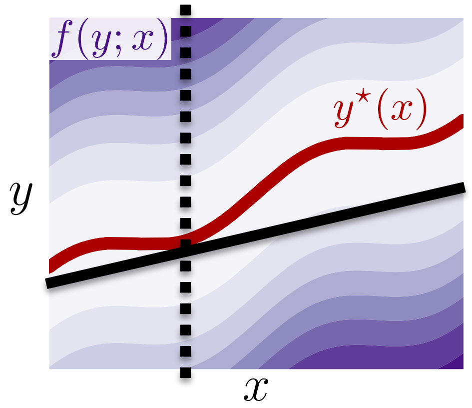
Domain knowledge: matrix operations, convolutions, activation functions, transformers, attention mechanisms
This talk: optimization-based domain knowledge

This talk: integrating optimization and learning

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

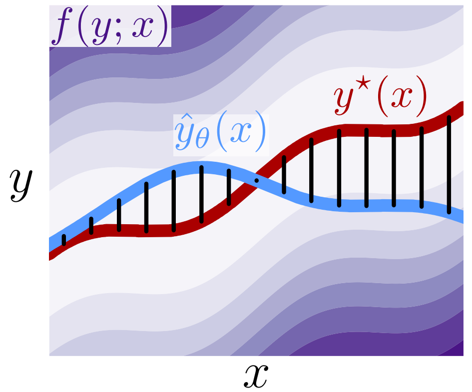
Differentiable optimization — $\frac{\partial}{\partial x} y^\star(x)$

Task-based optimization
Foundations: convex quadratic and cone programs
Applications



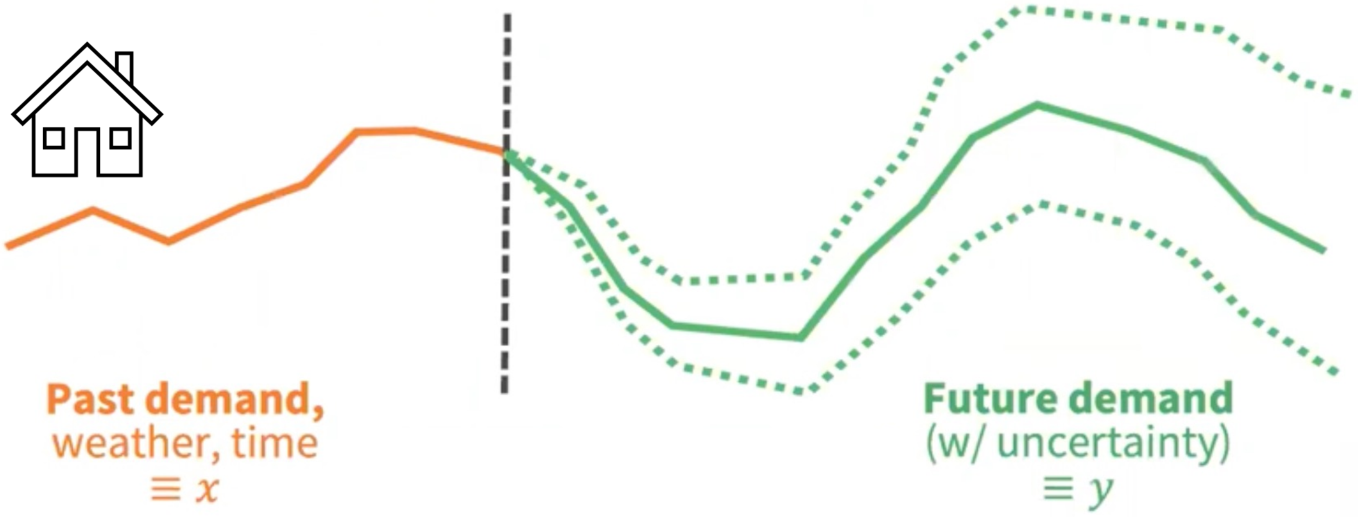
Amortized optimization — $\hat{y}_\theta(x) \approx y^\star(x)$

RL as amortized optimization
Foundations: modeling and loss choices
Applications
Amortization via learning latent subspaces



Demand prediction and scheduling

Predicted electricity demands → Electricity generation schedule

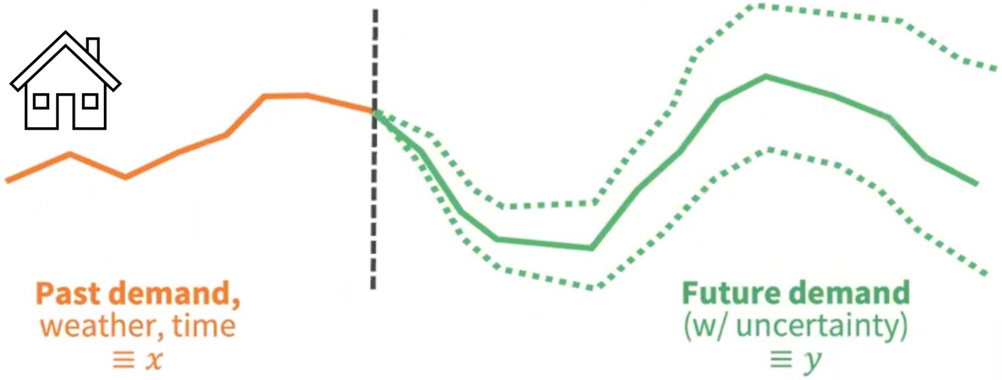


$$z^\star(x, y) \in \underset{z \in \mathcal{C}(x,y)}{\operatorname{argmin}} f(z; x, y)$$

Using predictions for scheduling

Stage 1: maximum likelihood training

A lot of data → Model → Predictions → Loss (log-likelihood)



Stage 2: deploy within a larger system

Query point → Model → Prediction → Downstream task (an optimization problem)

$$z^\star(x, y) \in \underset{z \in \mathcal{C}(x,y)}{\operatorname{argmin}} f(z; x, y)$$

Using predictions for scheduling

Stage 1: maximum likelihood training

A lot of data → Model → Predictions → Loss (log-likelihood)

max-likelihood model ≠ best model for the task

Why? Modeling errors impact tasks in different ways

Task-based end-to-end model learning in stochastic optimization. Donti, Amos, and Kolter, NeurIPS 2017.
Objective mismatch in model-based reinforcement learning. Lambert, Amos, Yadan, and Calandra, L4DC 2020.

Stage 2: deploy within a larger system

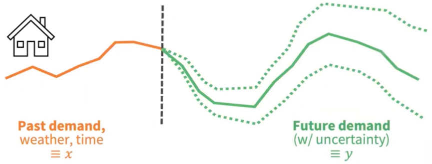
Query point → Model → Prediction → Downstream task (an optimization problem)

$$z^\star(x,y) \in \underset{z \in \mathcal{C}(x,y)}{\operatorname{argmin}} f(z; x, y)$$

Idea: improve the model with the task loss

Stage 1: maximum likelihood training

A lot of data → Model → Predictions → Loss (log-likelihood)



$\nabla_\theta \ell_{\mathrm{NLL}}$: standard backpropagation

Stage 2: deploy within a larger system. Improve the model with the task information

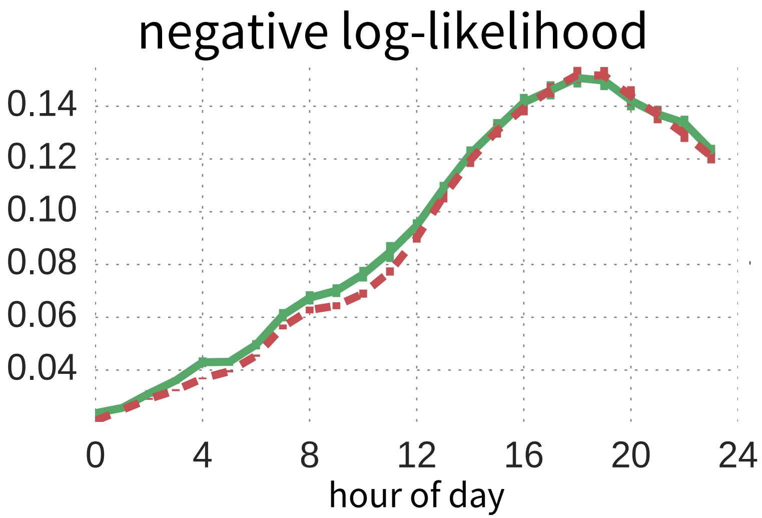
Query point → Model → Prediction → Downstream task (an optimization problem)

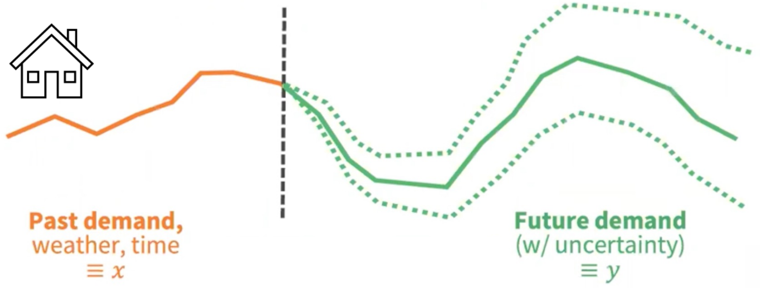
$$z^\star(x, y) \in \operatorname*{argmin}_{z \in \mathcal{C}(x,y)} f(z; x, y)$$

$\nabla_\theta \ell_{\mathrm{task}}$: differentiates through an optimization problem

Incorporating the task loss is crucial

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





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

How to differentiate an optimization problem?

Stage 1: maximum likelihood training

A lot of data → Model → Predictions → Loss (log-likelihood)

$\nabla_\theta \ell_{\mathrm{NLL}}$: standard backpropagation

Stage 2: deploy within a larger system. Improve the model with the task information

Query point → Model → Prediction → Downstream task (an optimization problem)

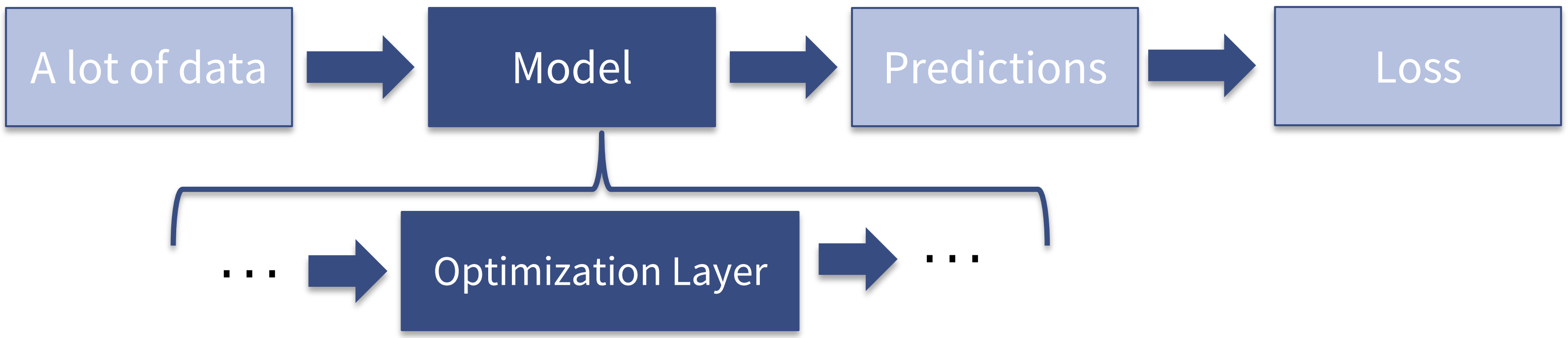
$$z^\star(x,y) \in \underset{z \in \mathcal{C}(x,y)}{\operatorname{argmin}}\, f(z; x, y)$$

$\nabla_\theta \ell_{\mathrm{task}}$: differentiates through an optimization problem

???

Differentiable optimization layers

Definition. A **differentiable optimization layer** for a machine learning model internally solves an optimization problem and is learned with backpropagation



$$
\begin{aligned}
z_{i+1} = \quad & \underset{z}{\operatorname{argmin}} \; f_\theta(z, z_i) \\
& \text{subject to } \; z \in C_\theta(z, z_i)
\end{aligned}
$$

Differentiable convex quadratic programs

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

$$x^\star = \underset{x}{\operatorname{argmin}} \ \frac{1}{2} x^\top Q x + p^\top x \quad \text{subject to} \ Ax = b \quad Gx \leq h$$

$$\Downarrow$$

KKT Optimality

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

$$\Downarrow$$

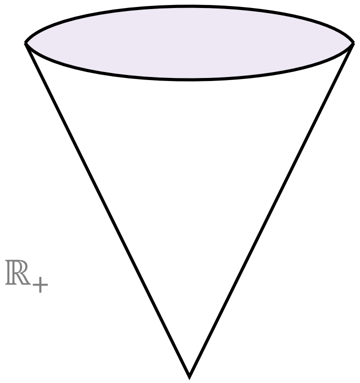
Implicitly differentiating $\mathcal{R}$ gives $D_\theta(z^\star) = -\left(D_z \mathcal{R}(z^\star)\right)^{-1} D_\theta \mathcal{R}(z^\star)$

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.

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

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: $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_p$



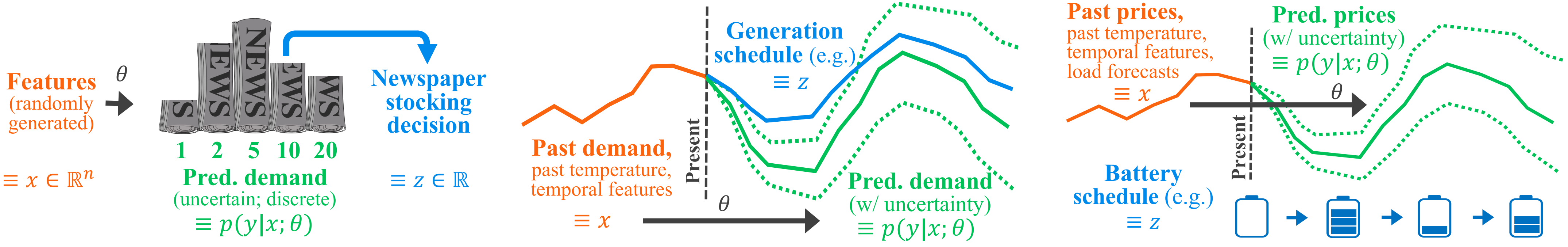
Conic Optimality
Find $z^\star$ s.t. $\mathcal{R}(z^\star, \theta) = 0$ where $z^\star = [x^\star, \ldots]$ and $\theta = \{A, b, c\}$

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

Applications of differentiable optimization

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

Task-based learning (task-aware predictions, decision-focused learning)



(a) Inventory stock problem (b) Load forecasting problem (c) Price forecasting problem

Applications of differentiable optimization

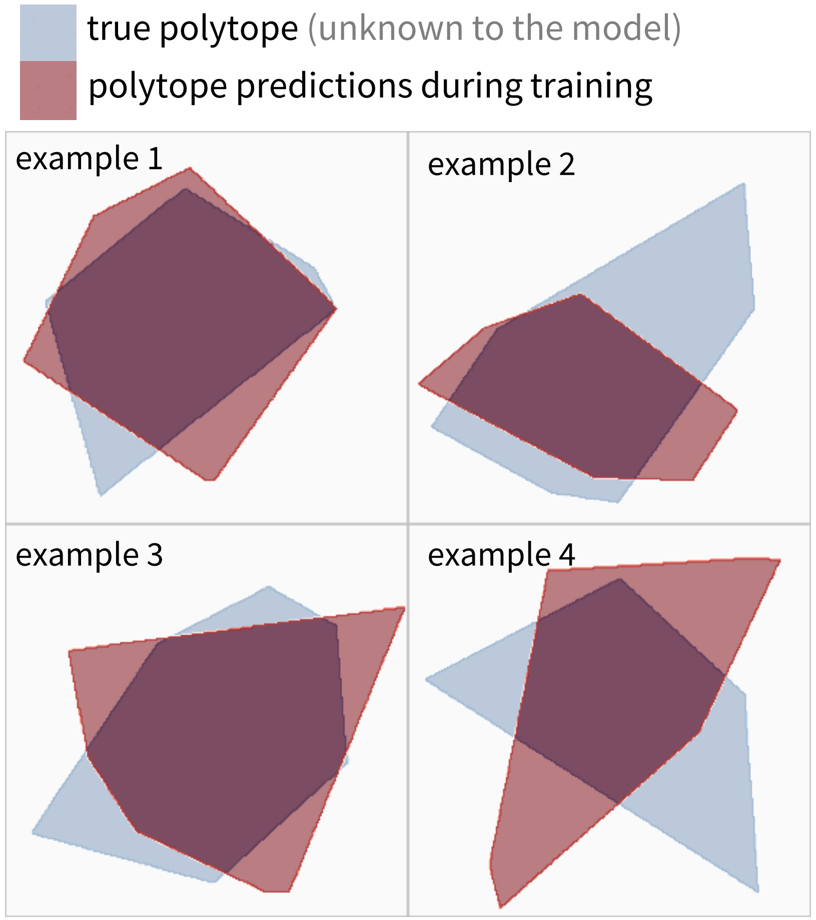
OptNet: Differentiable Optimization as a Layer in Neural Networks. Amos and Kolter, ICML 2017.

Task-based learning (task-aware predictions, decision-focused learning)

Learning **hard constraints** (Sudoku from data)

$$y^\star(x) = \underset{y}{\operatorname{argmin}} \ \operatorname{dist}(x, y) \quad \text{subject to} \quad Gy \le h$$

parameters $\theta = \{G, h\}$



Applications of differentiable optimization

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

Task-based learning (task-aware predictions, decision-focused learning)

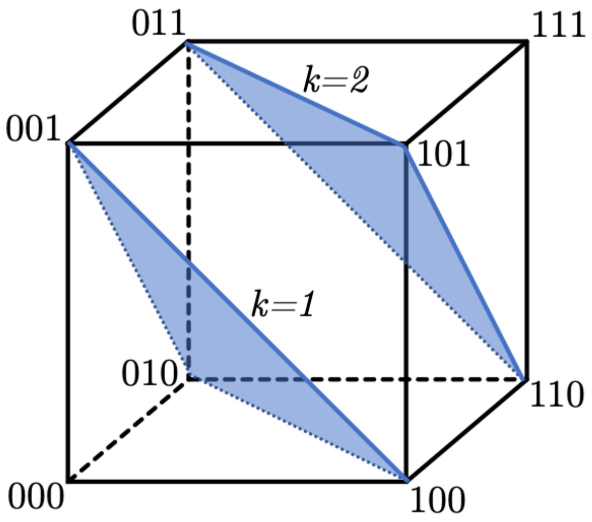
Learning **hard constraints** (Sudoku from data)

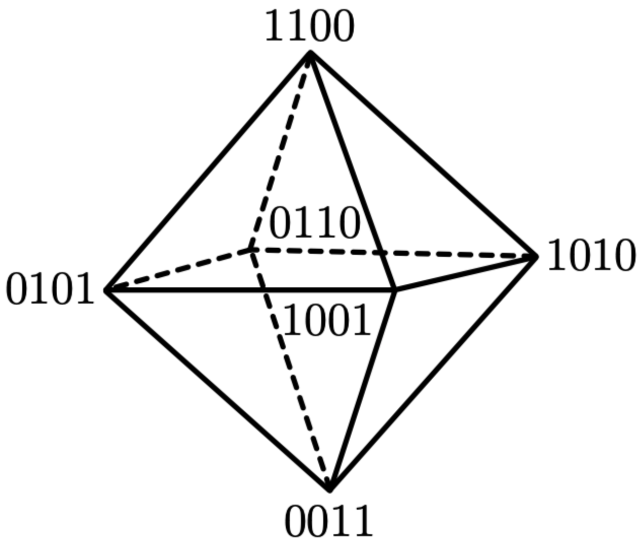
Modeling **projections** (ReLU, sigmoid, softmax; differentiable top-k, and sorting)

$$\operatorname{argtopk}_\tau(x) = \underset{y}{\operatorname{argmin}} \; -y^\top x - \tau H_b(y) \quad \text{subject to} \quad 0 \le y \le 1, \; 1^\top y = k$$

$$H_b(y) := -\sum_i (y_i \log y_i + (1-y_i)\log(1-y_i))$$

is the binary cross-entropy function





Applications of differentiable optimization

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

Task-based learning (task-aware predictions, decision-focused learning)

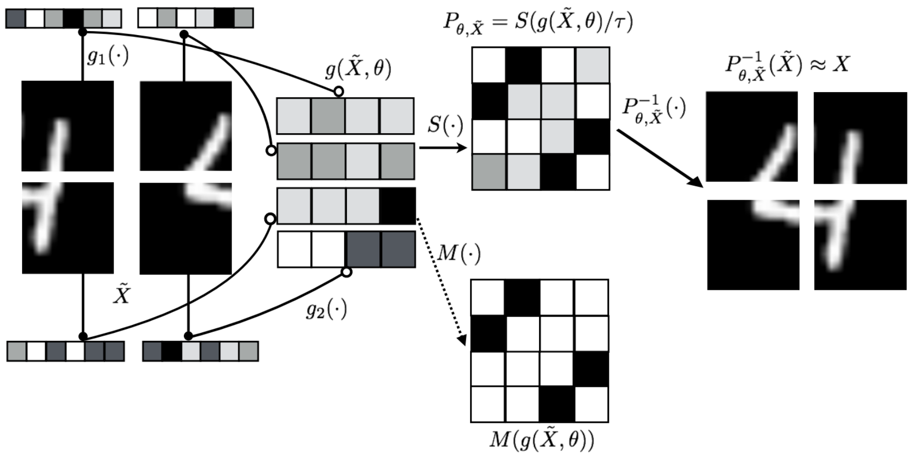
Learning **hard constraints** (Sudoku from data)

Modeling **projections** (ReLU, sigmoid, softmax; differentiable top-k, and sorting)

Gumbel-Sinkhorn: projection onto the **Birkhoff polytope** $\mathcal{B}_N$:

$$\pi_{\mathcal{B}_N,\tau}(X) = \operatorname*{argmax}_{P\in\mathcal{B}_N} \langle P, X\rangle_F + \tau H(P)$$

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



Applications of differentiable optimization

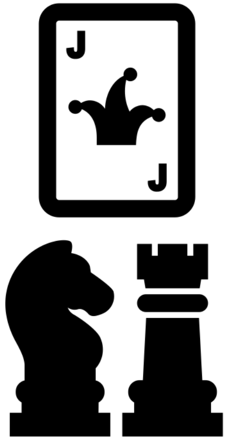
What Game Are We Playing? End-to-end Learning in Normal and Extensive Form Games. Ling et al., IJCAI 2018.

Task-based learning (task-aware predictions, decision-focused learning)

Learning **hard constraints** (Sudoku from data)

Modeling **projections** (ReLU, sigmoid, softmax; differentiable top-k, and sorting)

Game theory (differentiable equilibrium finding)



$$\min_u \max_v u^\top P v \quad \text{subject to} \quad 1^\top u = 1 \quad 1^\top v = 1 \quad u, v \geq 0$$

Parameterize and learn payoff P

Applications of differentiable optimization

Differentiable MPC for end-to-end planning and control. Amos et al., NeurIPS 2018.
The differentiable cross-entropy method. Amos and Yarats, ICML 2020.

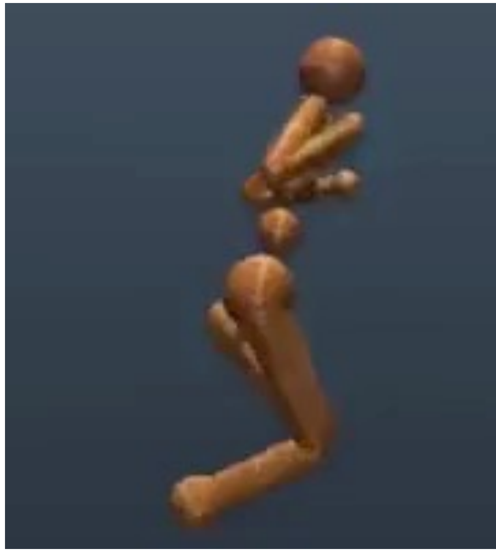
Task-based learning (task-aware predictions, decision-focused learning)

Learning **hard constraints** (Sudoku from data)

Modeling **projections** (ReLU, sigmoid, softmax; differentiable top-k, and sorting)

Game theory (differentiable equilibrium finding)

RL and control (differentiable control-based policies, enforcing safety constraints)



$$x^\star_{1:T}, u^\star_{1:T} \in \operatorname*{argmin}_{x_{1:T}, u_{1:T}} \sum_t \underbrace{\boxed{C(x_t, u_t)}}_{\text{cost}} \ \text{s.t.} \ \underbrace{\boxed{x_1 = x_{\text{init}}}}_{\text{initial state}} \ \underbrace{\boxed{x_{t+1} = f(x_t, u_t)}}_{\text{dynamics}} \ \underbrace{\boxed{u_t \in \mathcal{U}}}_{\text{constraints}}$$

Parameterize and learn cost and dynamics

Applications of differentiable optimization

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

Task-based learning (task-aware predictions, decision-focused learning)

Learning **hard constraints** (Sudoku from data)

Modeling **projections** (ReLU, sigmoid, softmax; differentiable top-k, and sorting)

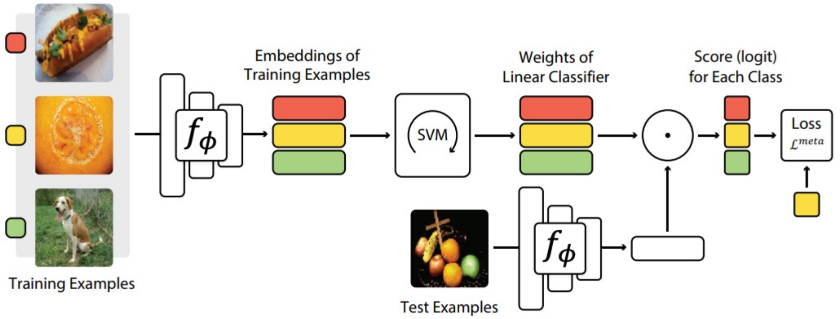
Game theory (differentiable equilibrium finding)

RL and control (differentiable control-based policies, enforcing safety constraints)

Meta-learning (differentiable SVMs and optimizers, implicit MAML)

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

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



Applications of differentiable optimization

Input-convex neural networks. Amos, Xu, Kolter, ICML 2017.

Task-based learning (task-aware predictions, decision-focused learning)

Learning **hard constraints** (Sudoku from data)

Modeling **projections** (ReLU, sigmoid, softmax; differentiable top-k, and sorting)

Game theory (differentiable equilibrium finding)

RL and control (differentiable control-based policies, enforcing safety constraints)

Meta-learning (differentiable SVMs and optimizers, implicit MAML)

Energy-based learning and **structured prediction** (differentiable inference with, e.g., ICNNs)

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

Applications of differentiable optimization

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

Task-based learning (task-aware predictions, decision-focused learning)

Learning **hard constraints** (Sudoku from data)

Modeling **projections** (ReLU, sigmoid, softmax; differentiable top-k, and sorting)

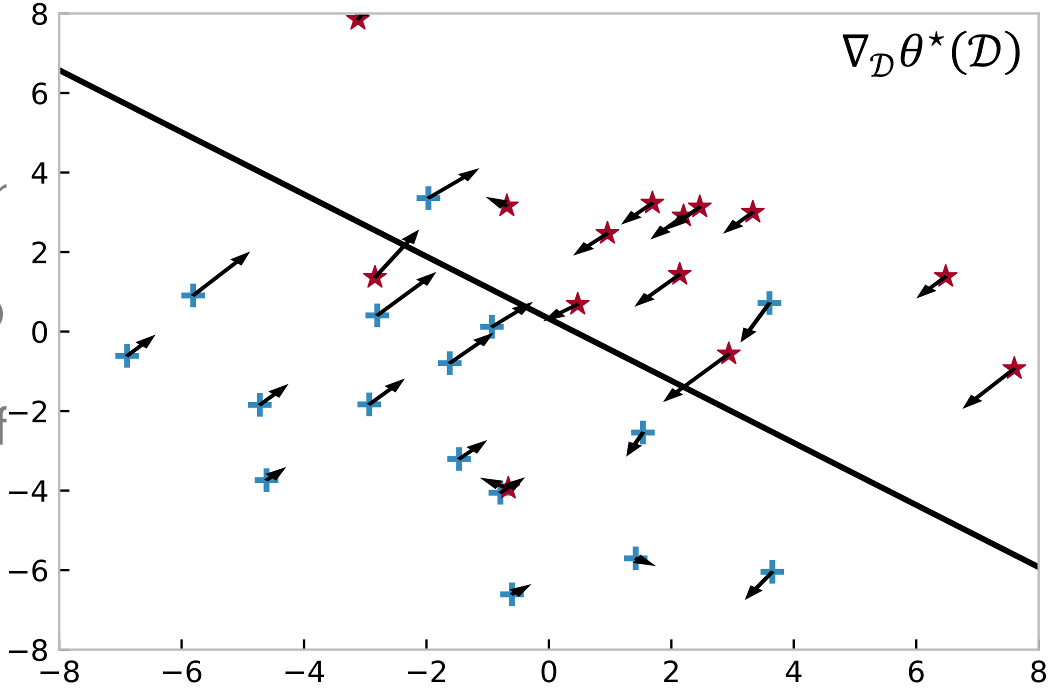
Game theory (differentiable equilibrium finding)

RL and control (differentiable control-based policies, er

Meta-learning (differentiable SVMs and optimizers, imp

Energy-based learning and **structured prediction** (diff

Sensitivity analysis (differentiable logistic regression)



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

Applications of differentiable optimization

Task-based learning (task-aware predictions, decision-focused learning)

Learning **hard constraints** (Sudoku from data)

Modeling **projections** (ReLU, sigmoid, softmax; differentiable top-k, and sorting)

Game theory (differentiable equilibrium finding)

RL and control (differentiable control-based policies, enforcing safety constraints)

Meta-learning (differentiable SVMs and optimizers, implicit MAML)

Energy-based learning and **structured prediction** (differentiable inference with, e.g., ICNNs)

Sensitivity analysis (differentiable logistic regression)

Differentiable CVXPY layers

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

$$
\begin{aligned}
x^\star(\theta) = \ & \underset{x}{\operatorname{argmin}} \ f(x;\theta) \\
& \text{subject to } \ g(x;\theta) \le 0 \\
& \qquad\qquad\quad h(x;\theta) = 0
\end{aligned}
$$

(Officially part of CVXPY!)

$$
\begin{aligned}
z_{i+1} = \ & \underset{z}{\operatorname{argmin}} \ \frac{1}{2} z^\top Q(z_i) z + q(z_i)^\top z \\
& \text{subject to } \ A(z_i) z = b(z_i) \\
& \qquad\qquad\quad G(z_i) z \le h(z_i)
\end{aligned}
$$

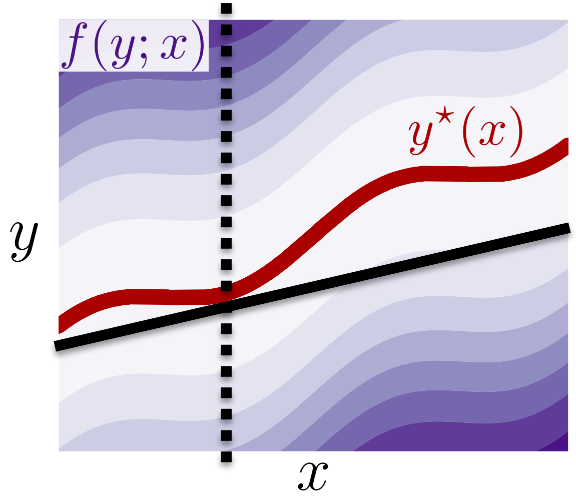
Parameters/Submodules : Q, q, A, b, G, h

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

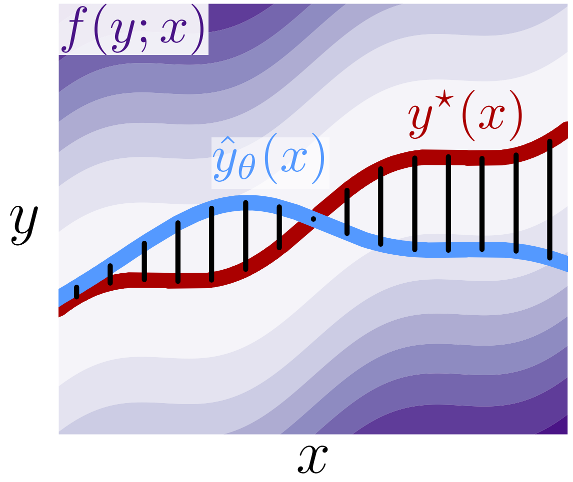
```
import cvxpy as cp
from cvxpyth import CvxpyLayer
obj = cp.Minimize(0.5*cp.quad_form(x, Q) + p.T * x)
cons = [A*x == b, G*x <= h]
prob = cp.Problem(obj, cons)
layer = CvxpyLayer(prob, params=[Q, p, A, b, G, h], out=[x])
```

This talk

Differentiable optimization — $\frac{\partial}{\partial x} y^\star(x)$
Task-based optimization
Foundations: convex quadratic and cone programs
Applications



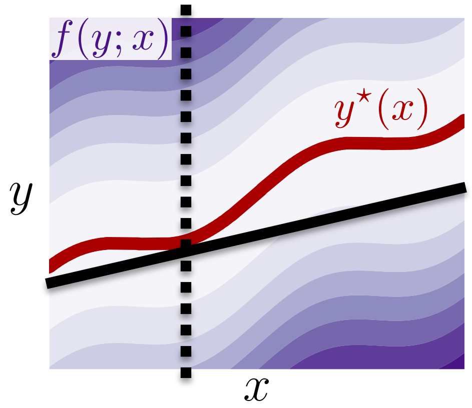
Amortized optimization — $\hat{y}_\theta(x) \approx y^\star(x)$
RL as amortized optimization
Foundations: modeling and loss choices
Applications
Amortization via learning latent subspaces



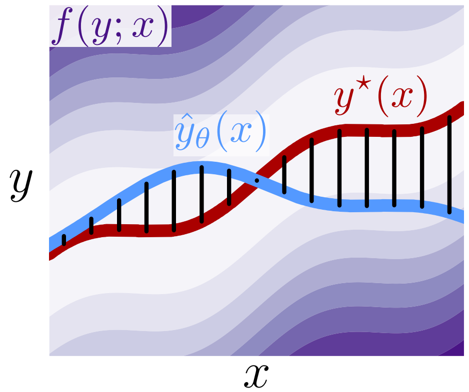
This talk: integrating optimization and learning

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

Differentiable optimization — $\frac{\partial}{\partial x} y^\star(x)$
Task-based optimization
Foundations: convex quadratic and cone programs
Applications

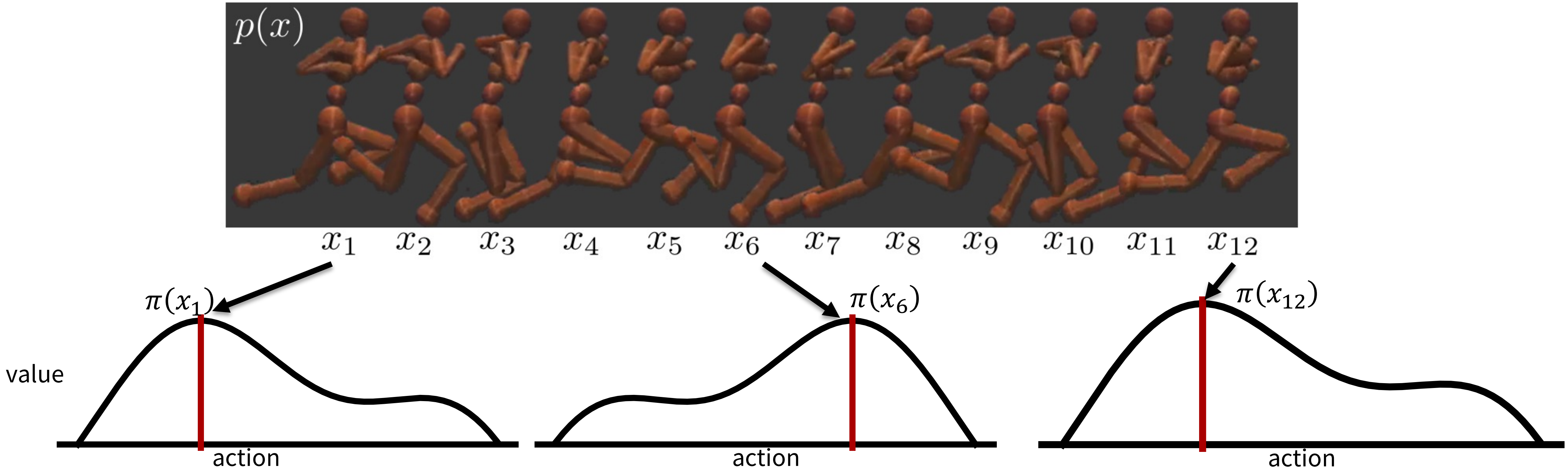


Amortized optimization — $\hat{y}_\theta(x) \approx y^\star(x)$
RL as amortized optimization
Foundations: modeling and loss choices
Applications
Amortization via learning latent subspaces



Deploying optimization and repeated solves

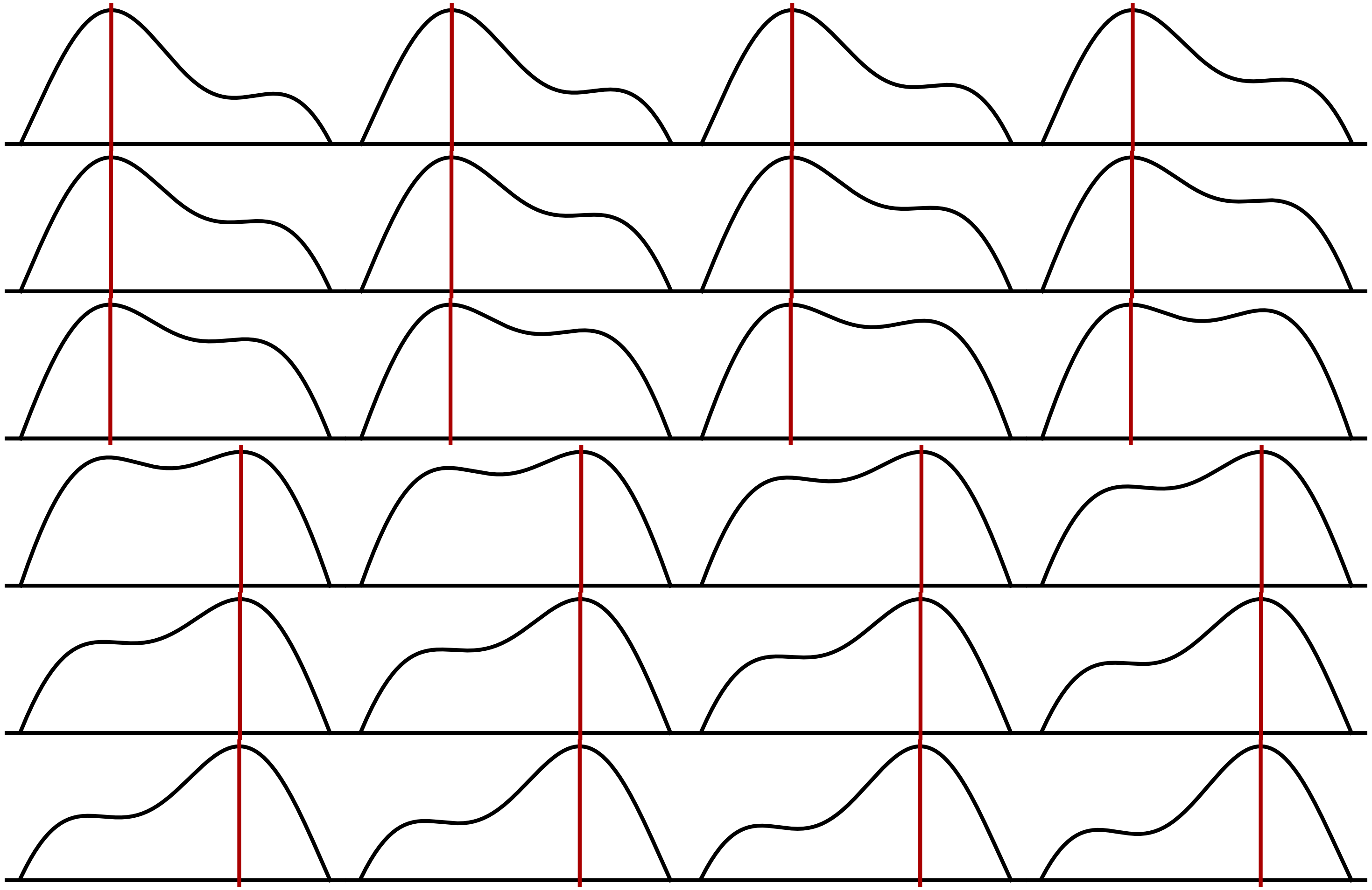
Tutorial on amortized optimization for learning to optimize over continuous domains. Amos, Foundations and Trends in Machine Learning 2023.
On the model-based stochastic value gradient for continuous reinforcement learning. Amos et al., L4DC 2021.

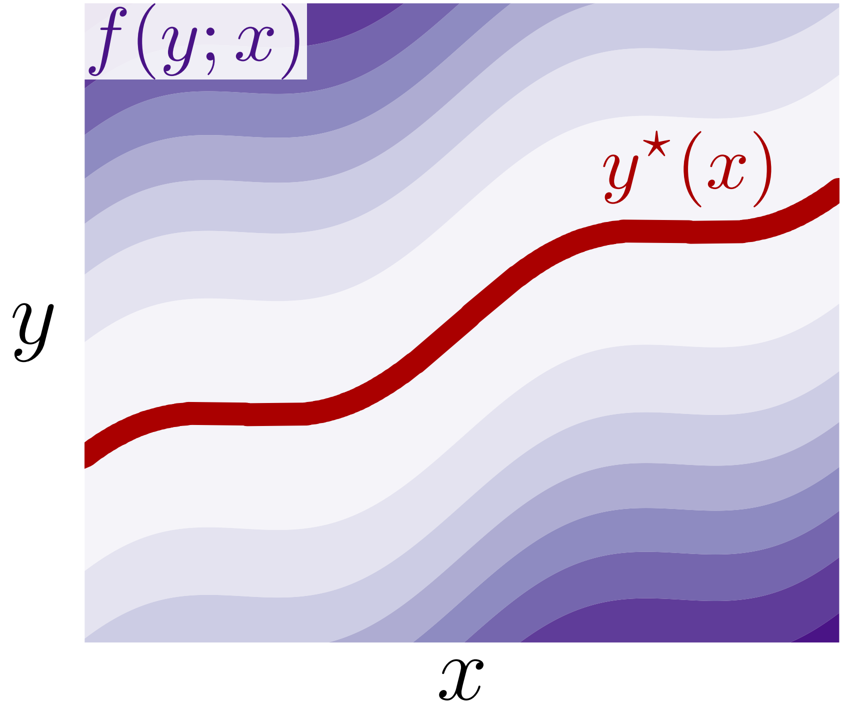


$$\pi(x) = \operatorname*{argmax}_u Q(x, u)$$

Repeatedly solved problems share structure

Tutorial on amortized optimization for learning to optimize over continuous domains. Amos, Foundations and Trends in Machine Learning 2023.





Amortization: approximate the solution map

Tutorial on amortized optimization for learning to optimize over continuous domains. Amos, Foundations and Trends in Machine Learning 2023.

A **fast amortization model** $\hat{y}_\theta$ can be **25,000 times faster** than solving $y^\star$ from scratch for VAEs

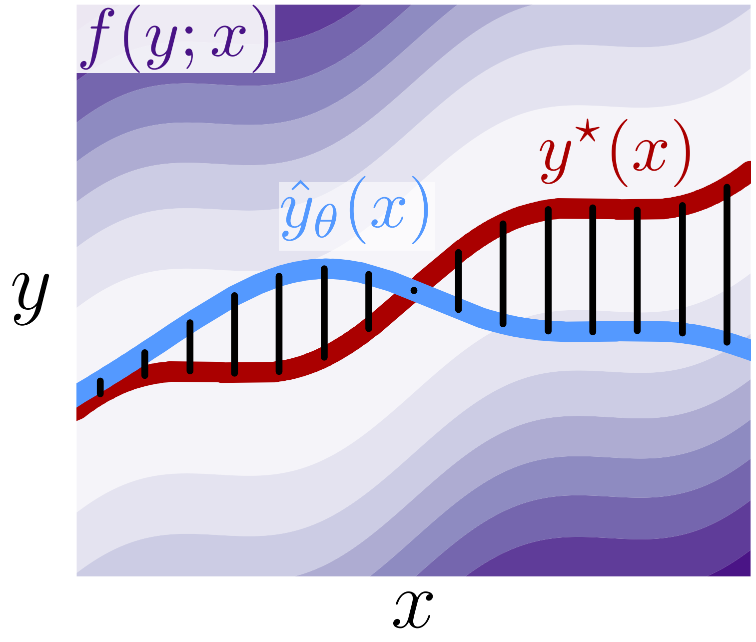
Amortization model $\hat{y}_\theta(x)$ tries to approximate $y^\star(x)$

Example: A neural network mapping from x to the solution

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

Regression: $\mathcal{L}(\hat{y}_\theta) \coloneqq \mathbb{E}_{p(x)} \|\hat{y}_\theta(x) - y^\star(x)\|_2^2$

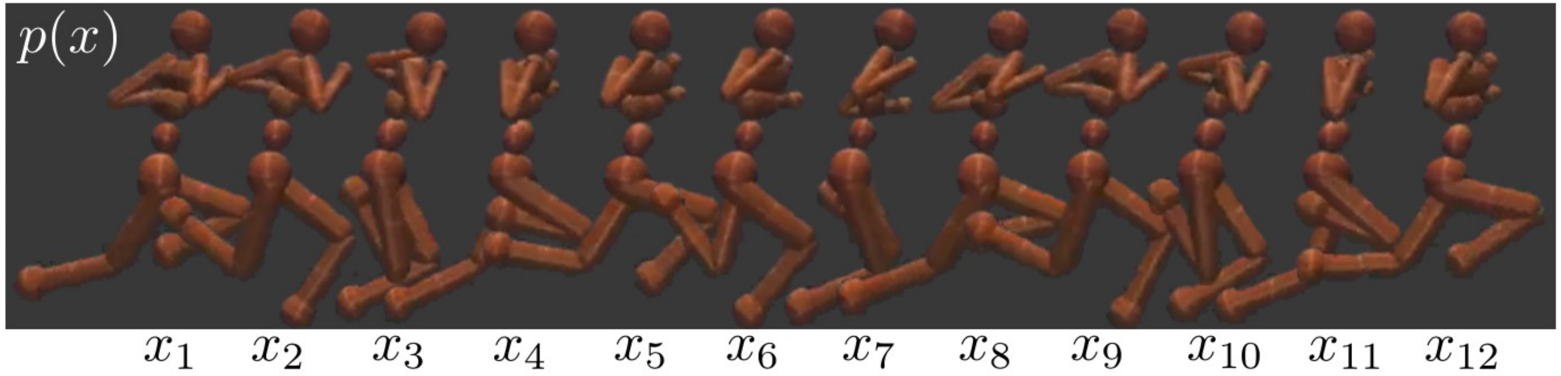
Objective: $\mathcal{L}(\hat{y}_\theta) \coloneqq \mathbb{E}_{p(x)} f(\hat{y}_\theta(x))$



Applications of amortized optimization

Tutorial on amortized optimization for learning to optimize over continuous domains. Amos, Foundations and Trends in Machine Learning 2023.

Reinforcement learning and **control** (actor-critic methods, SAC, DDPG, GPS, BC)



Applications of amortized optimization

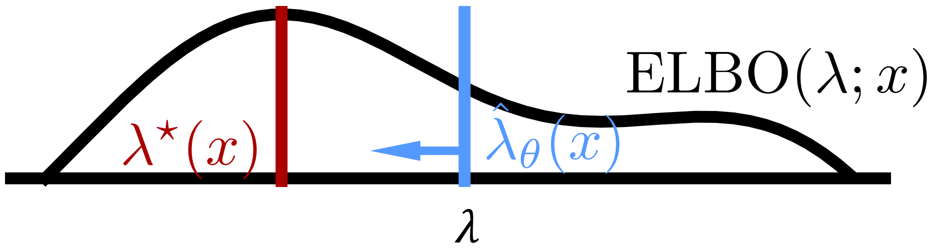
Tutorial on amortized optimization for learning to optimize over continuous domains. Amos, Foundations and Trends in Machine Learning 2023.

Reinforcement learning and **control** (actor-critic methods, SAC, DDPG, GPS, BC)

Variational inference (VAEs, semi-amortized VAEs)

Given a **VAE** model $p(x) = \log \int_z p(x|z)p(x)$, **encoding** amortizes the optimization problem

$$\lambda^\star(x) = \operatorname*{argmax}_\lambda \mathrm{ELBO}(\lambda; x) \quad \text{where} \quad \mathrm{ELBO}(\lambda; x) \coloneqq \mathbb{E}_{q(z;\lambda)}[\log p(x|z)] - \mathrm{D}_{\mathrm{KL}}(q(x;\lambda)|p(z)).$$



Applications of amortized optimization

Tutorial on amortized optimization for learning to optimize over continuous domains. Amos, Foundations and Trends in Machine Learning 2023.

Reinforcement learning and **control** (actor-critic methods, SAC, DDPG, GPS, BC)

Variational inference (VAEs, semi-amortized VAEs)

Meta-learning (HyperNets, MAML)

Given a **task** $\mathcal{T}$, amortize the **computation of the optimal parameters** of a model

$$\theta^\star(\mathcal{T}) = \operatorname*{argmax}_\theta \ell(\mathcal{T}, \theta)$$

Applications of amortized optimization

Tutorial on amortized optimization for learning to optimize over continuous domains. Amos, Foundations and Trends in Machine Learning 2023.

Reinforcement learning and **control** (actor-critic methods, SAC, DDPG, GPS, BC)

Variational inference (VAEs, semi-amortized VAEs)

Meta-learning (HyperNets, MAML)

Sparse coding (PSD, LISTA)

Given a **dictionary** W_d of **basis vectors** and **input** x, a **sparse code** is recovered with

$$y^\star(x) \in \operatorname*{argmin}_y \|x - W_d y\|_2^2 + \alpha \|y\|_1$$

Predictive sparse decomposition (PSD) and Learned ISTA (LISTA) **amortize this problem**

Kavukcuoglu, Ranzato, and LeCun, 2010. Gregor and LeCun, 2010.

Applications of amortized optimization

Tutorial on amortized optimization for learning to optimize over continuous domains. Amos, Foundations and Trends in Machine Learning 2023.

Reinforcement learning and **control** (actor-critic methods, SAC, DDPG, GPS, BC)

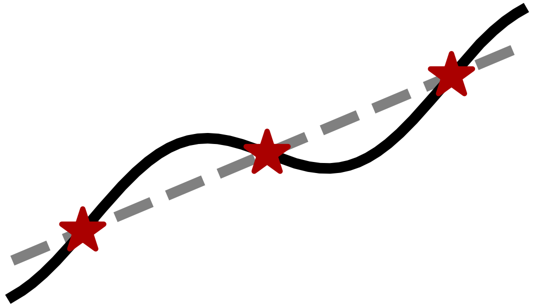
Variational inference (VAEs, semi-amortized VAEs)

Meta-learning (HyperNets, MAML)

Sparse coding (PSD, LISTA)

Roots, fixed points, and convex optimization (NeuralDEQs, RLQP, NeuralSCS)

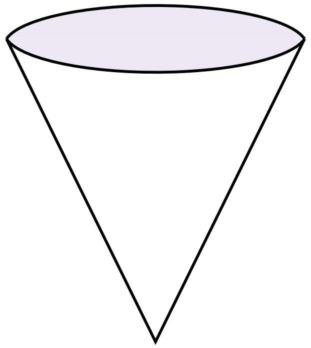
Finding fixed points $y = g(y)$



$$x^\star = \begin{array}{ll} \underset{x}{\operatorname{argmin}} & \frac{1}{2} x^\top Q x + p^\top x \\ \text{subject to} & b - Ax \in \mathcal{K} \end{array}$$

KKT conditions

Find $z^\star$ s.t. $\mathcal{R}(z^\star, \theta) = 0$



Applications of amortized optimization

Tutorial on amortized optimization for learning to optimize over continuous domains. Amos, Foundations and Trends in Machine Learning 2023.

Reinforcement learning and **control** (actor-critic methods, SAC, DDPG, GPS, BC)

Variational inference (VAEs, semi-amortized VAEs)

Meta-learning (HyperNets, MAML)

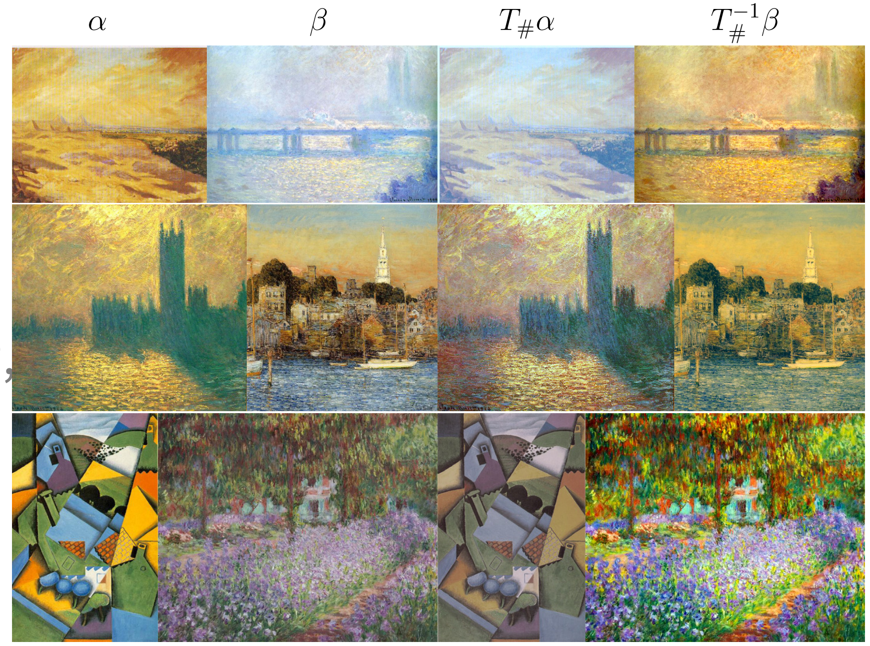
Sparse coding (PSD, LISTA)

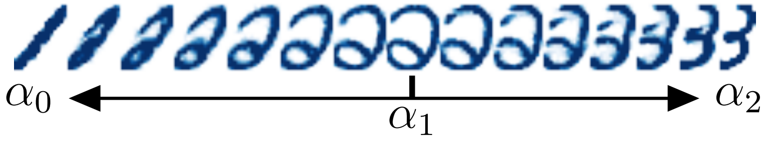
Roots, fixed points, and convex optimization (NeuralDEQs, RLQP,

Optimal transport (slicing, conjugation, Meta Optimal Transport)

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

Meta Optimal Transport. Amos et al., 2022





Applications of amortized optimization

Tutorial on amortized optimization for learning to optimize over continuous domains. Amos, Foundations and Trends in Machine Learning 2023.

Reinforcement learning and **control** (actor-critic methods, SAC, DDPG, GPS, BC)

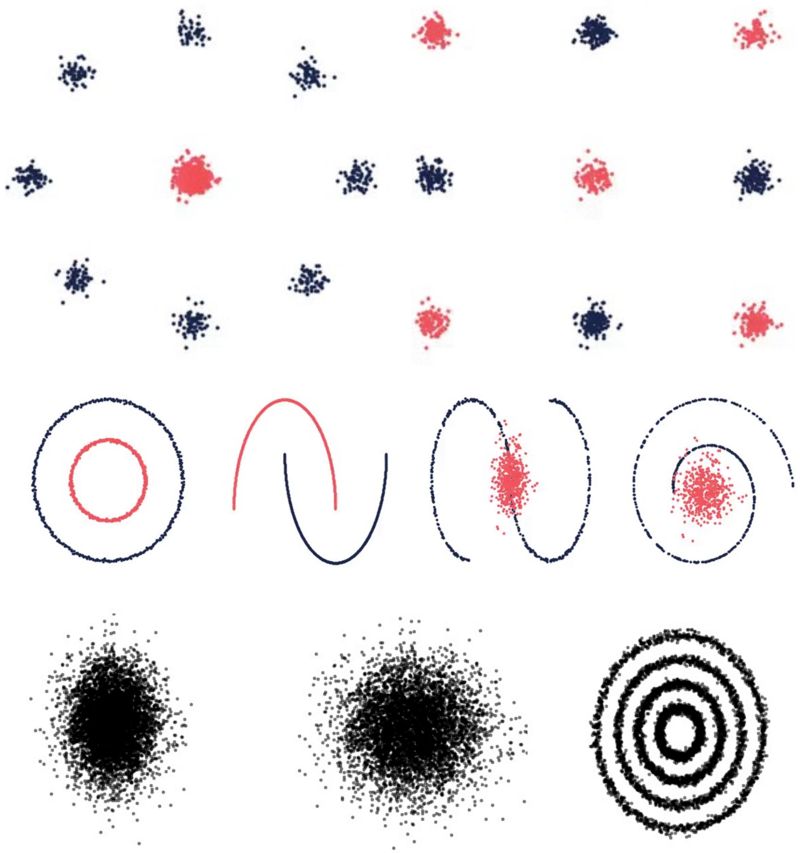
Variational inference (VAEs, semi-amortized VAEs)

Meta-learning (HyperNets, MAML)

Sparse coding (PSD, LISTA)

Roots, fixed points, and convex optimization (NeuralDEQs, RLQP, N

Optimal transport (slicing, conjugation, Meta Optimal Transport)



$$f^c(y) = -\inf_x f(x) - x^\top y$$

On amortizing convex conjugates for optimal transport. Amos, ICLR 2023

Applications of amortized optimization

Reinforcement learning and **control** (actor-critic methods, SAC, DDPG, GPS, BC)

Variational inference (VAEs, semi-amortized VAEs)

Meta-learning (HyperNets, MAML)

Sparse coding (PSD, LISTA)

Roots, fixed points, and convex optimization (NeuralDEQs, RLQP, NeuralSCS)

Optimal transport (slicing, conjugation, Meta Optimal Transport)

Foundations and Trends® in Machine Learning

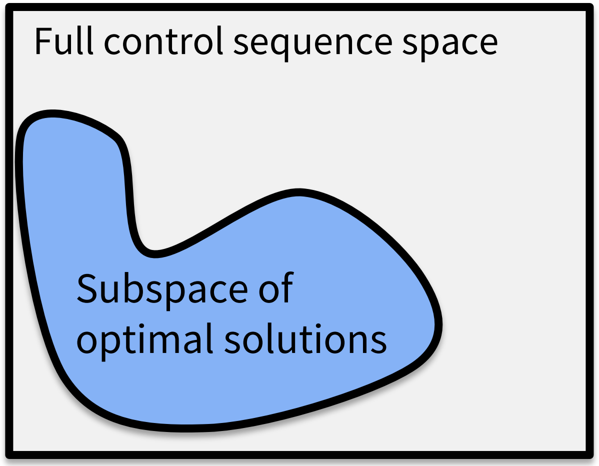
Tutorial on amortized optimization for learning to optimize over continuous domains

Brandon Amos BDA@FB.COM
Facebook AI Research, Meta

Amortization via learning latent subspaces

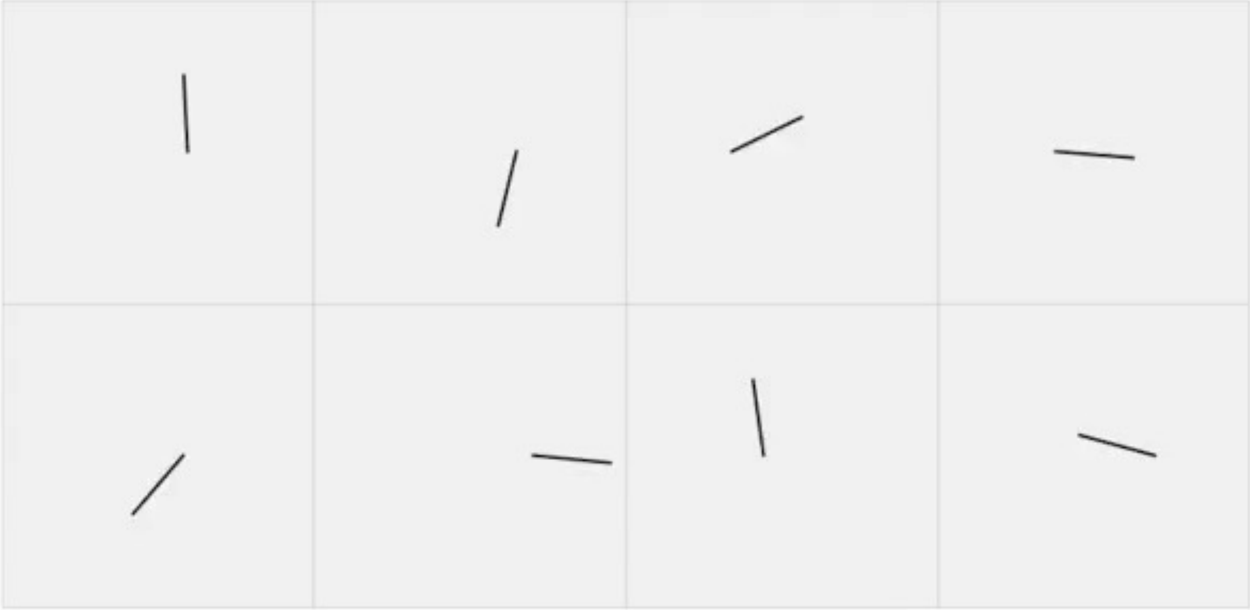
The differentiable cross-entropy method. Amos and Yarats, ICML 2020.

Amortize the problem by **learning a latent subspace** of optimal solutions
Only search over optimal solutions rather than the entire space



$$x^\star_{1:T}, u^\star_{1:T} \in \operatorname*{argmin}_{x_{1:T},u_{1:T}} \sum_t \overset{\text{cost}}{\boxed{C_\theta(x_t, u_t)}} \;\text{s.t.}\; \overset{\text{initial state}}{\boxed{x_1 = x_{\text{init}}}} \; \overset{\text{dynamics}}{\boxed{x_{t+1} = f_\theta(x_t, u_t)}} \; \overset{\text{constraints}}{\boxed{u_t \in \mathcal{U}}}$$

Cartpole videos

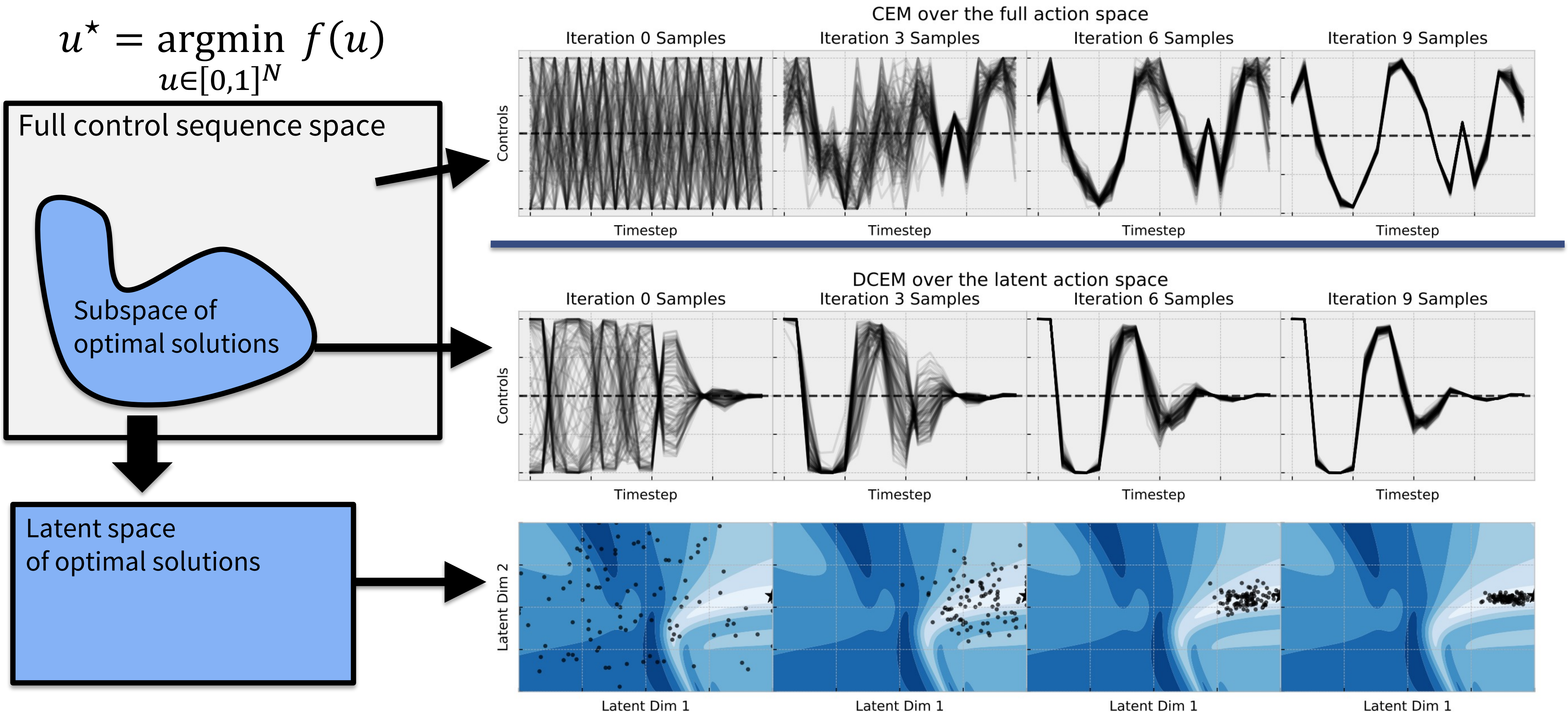


Optimal controls over time — force on the cartpole

Amortization via learning latent subspaces

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

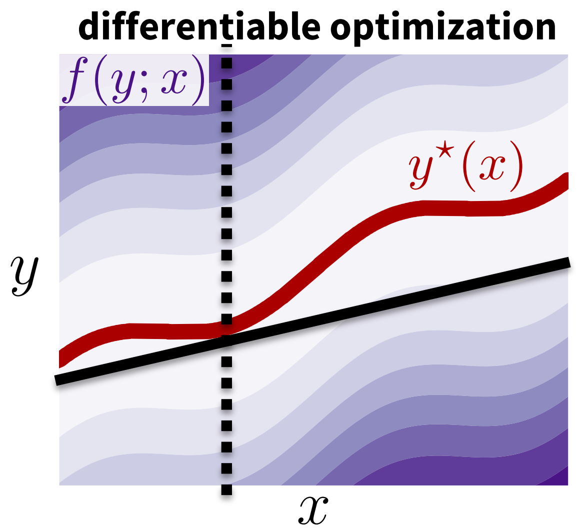
$$u^\star = \underset{u \in [0,1]^N}{\operatorname{argmin}} \; f(u)$$

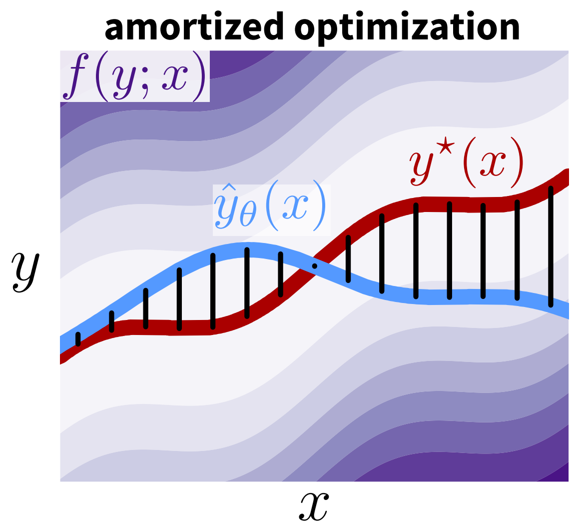


Future directions and open questions

Goal: build intelligent systems that **understand and interact** with the world
Why? To advance scientific and engineering discoveries

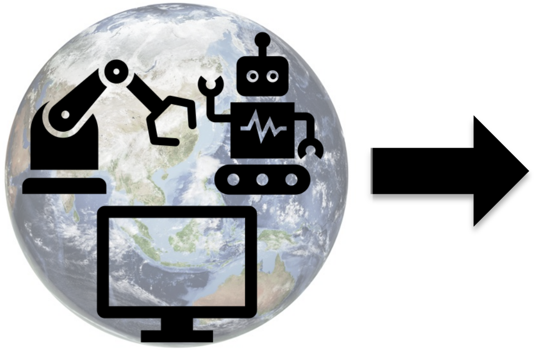
Advancing **optimization** and **machine learning foundations** is crucial





How to handle discrete spaces?

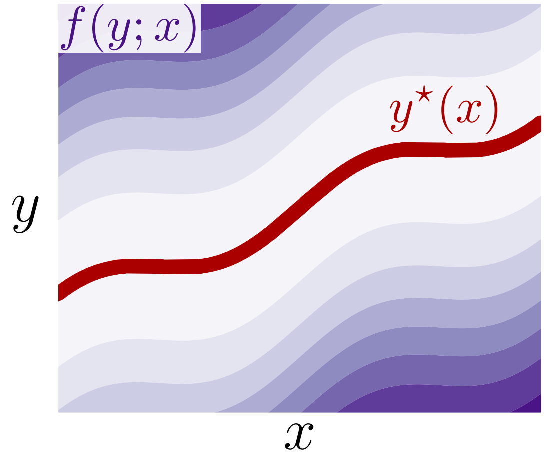
CombOptNet. Paulus, Rolínek, Musil, Amos, and Martius, ICML 2021.

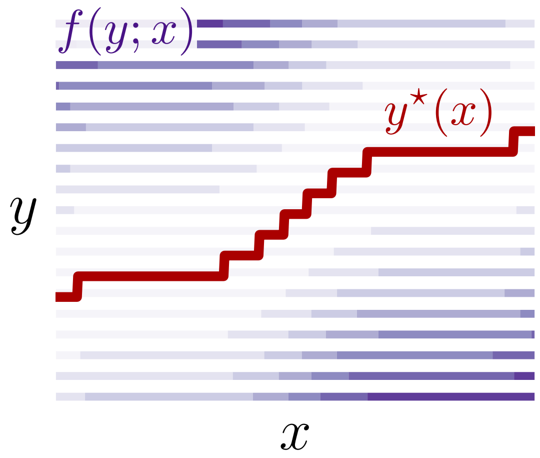


optimal solution / objective / context (or parameterization)

$$\underbrace{y^\star(x)}_{\text{optimal solution}} \in \underset{y \in \mathcal{C}(x)}{\operatorname{argmin}}\, f(y; x)$$

optimization variable / constraints





scheduling/assignment problems

knapsack problems

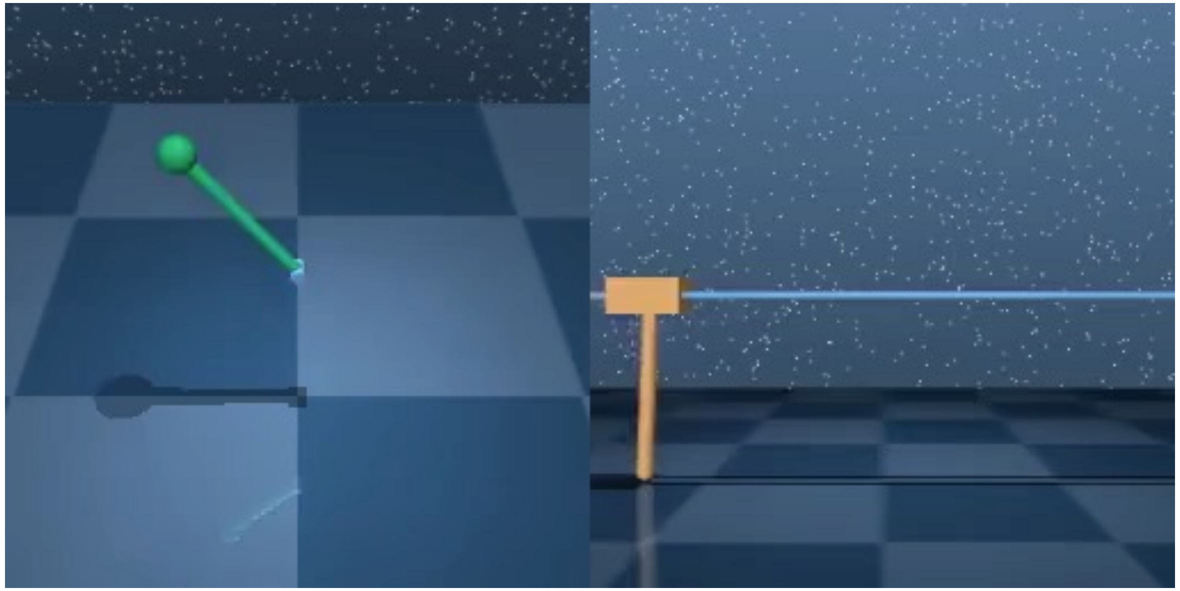
keypoint matching

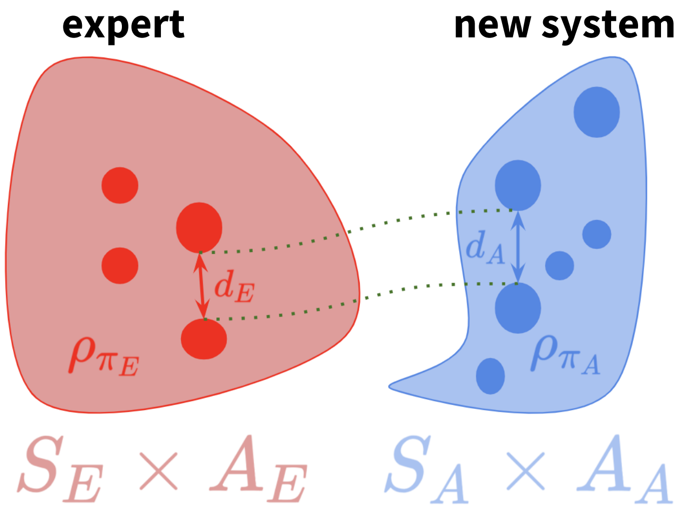
How to transfer knowledge between structures?

Cross-domain imitation learning via optimal transport. Fickinger, Cohen, Russell, Amos, ICLR 2022.

Optimization (optimal transport) **connects disparate spaces** to enable **knowledge transfer**

expert (3 dimensions) → **new system (5 dimensions)**



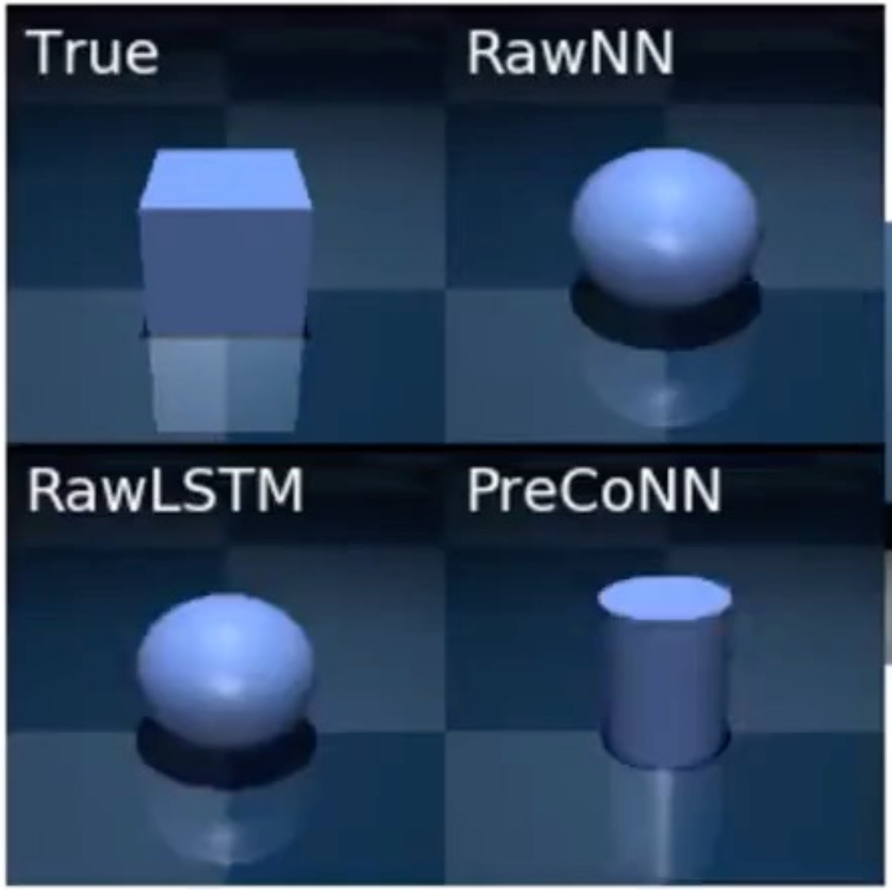


How can latent representations gain an awareness of unobserved concepts?

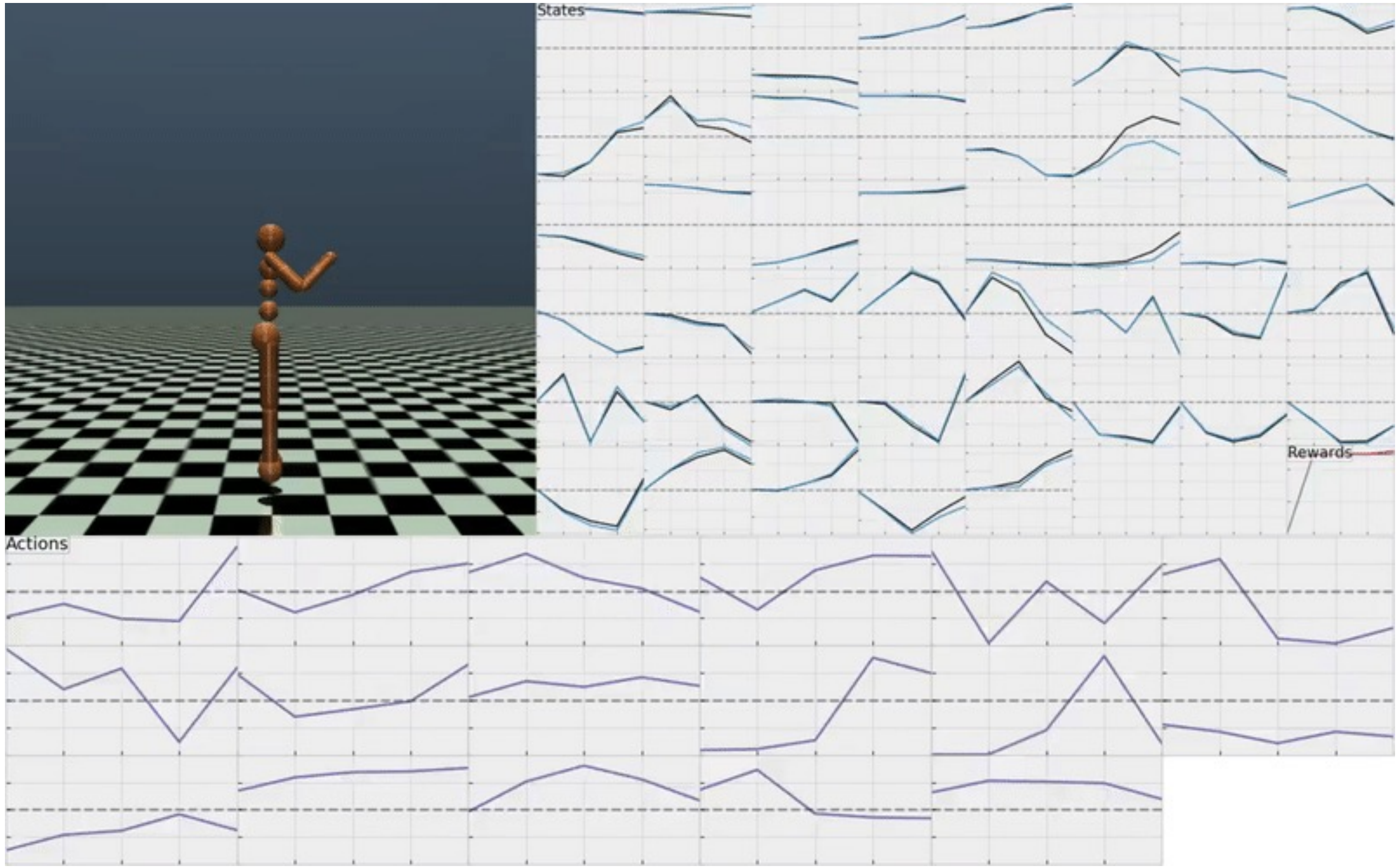
Learning awareness models. Amos et al., ICLR 2018.

***Situation awareness** is the perception of the elements in the environment within a volume of time and space, and the comprehension of their meaning, and the projection of their status in the near future.*

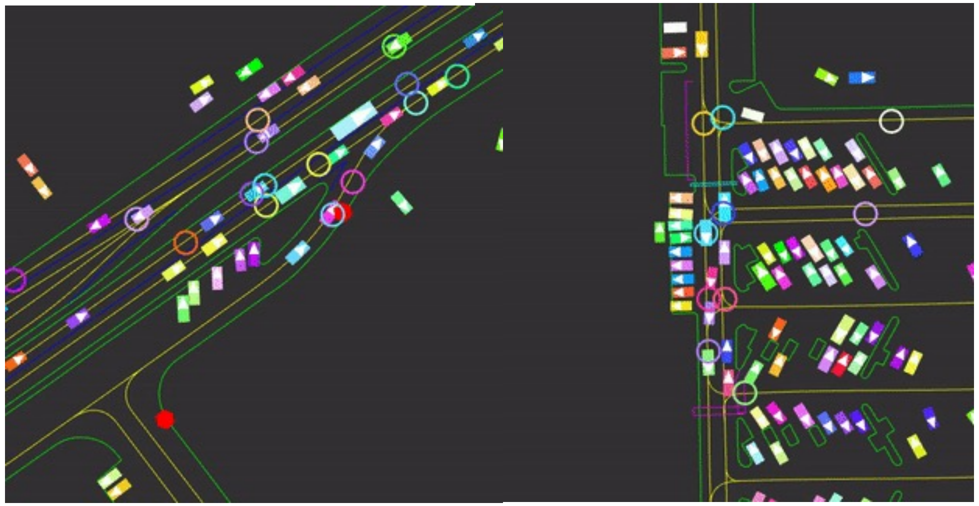
— Mica Endsley (1987)
Former Chief Scientist of the U.S. Air Force



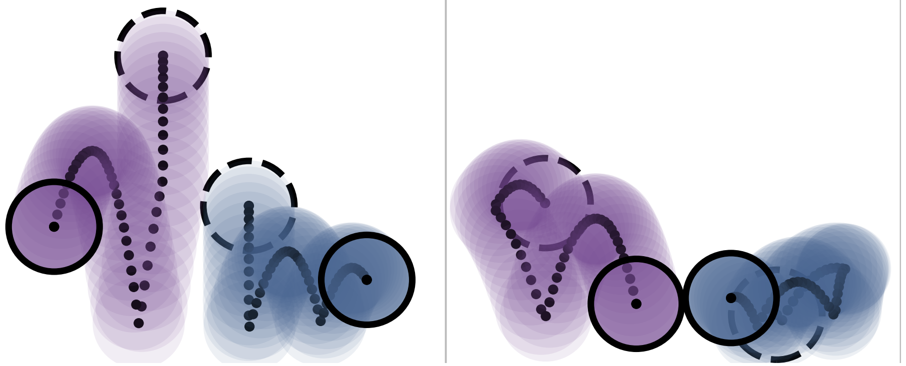
How to model and control non-trivial systems?



On the model-based stochastic value gradient for continuous reinforcement.
B. Amos et al., L4DC 2021.



Nocturne: a driving benchmark for multi-agent learning.
Vinitsky et al., NeurIPS Datasets and Benchmarks 2022

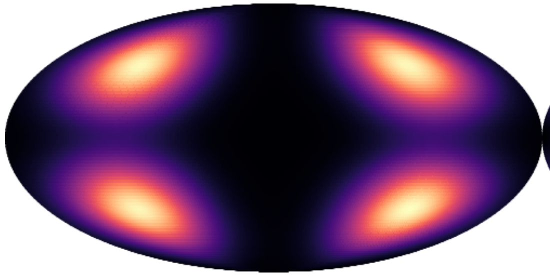


Learning Neural Event Functions for Ordinary Differential Equations.
Chen, Amos, Nickel, ICLR 2021.

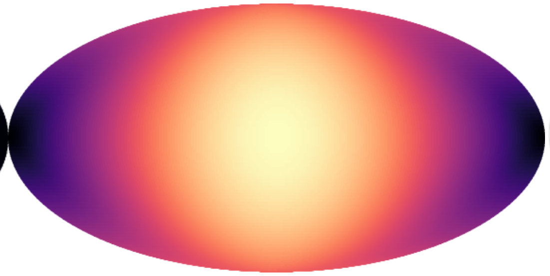
How to perform machine learning and optimization over non-Euclidean spaces?

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

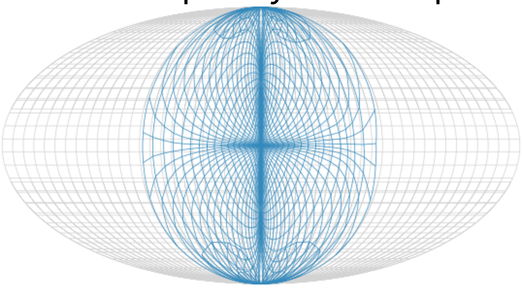
Base distribution



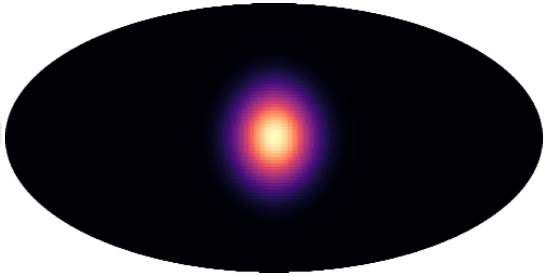
c-convex function



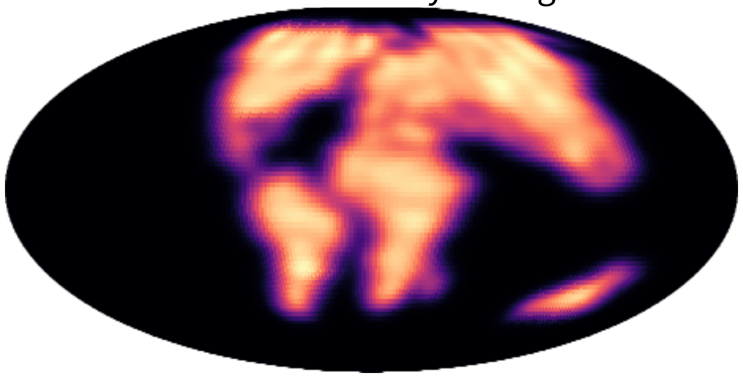
Grid warped by the transport



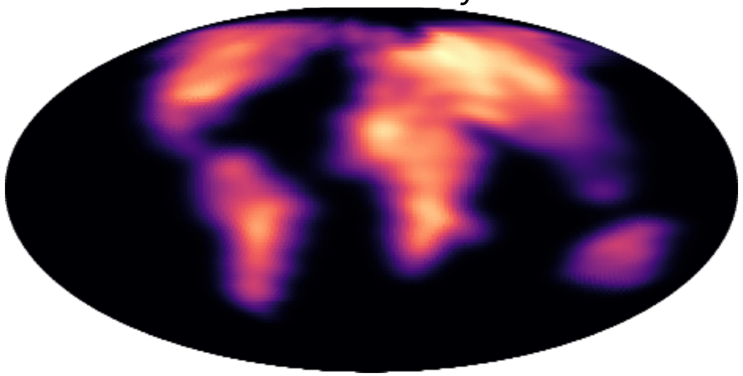
Push-forward distribution



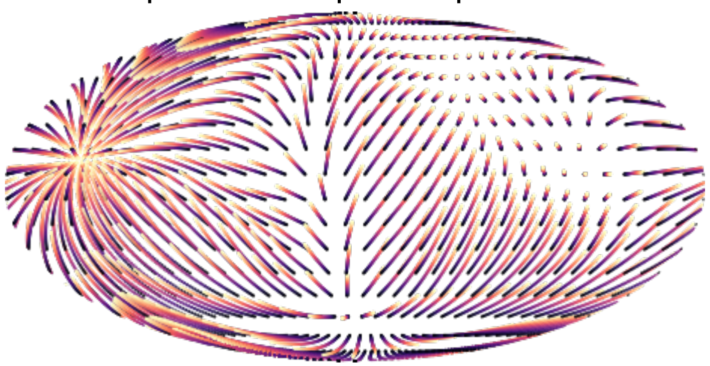
Earth 90 million years ago



Earth today

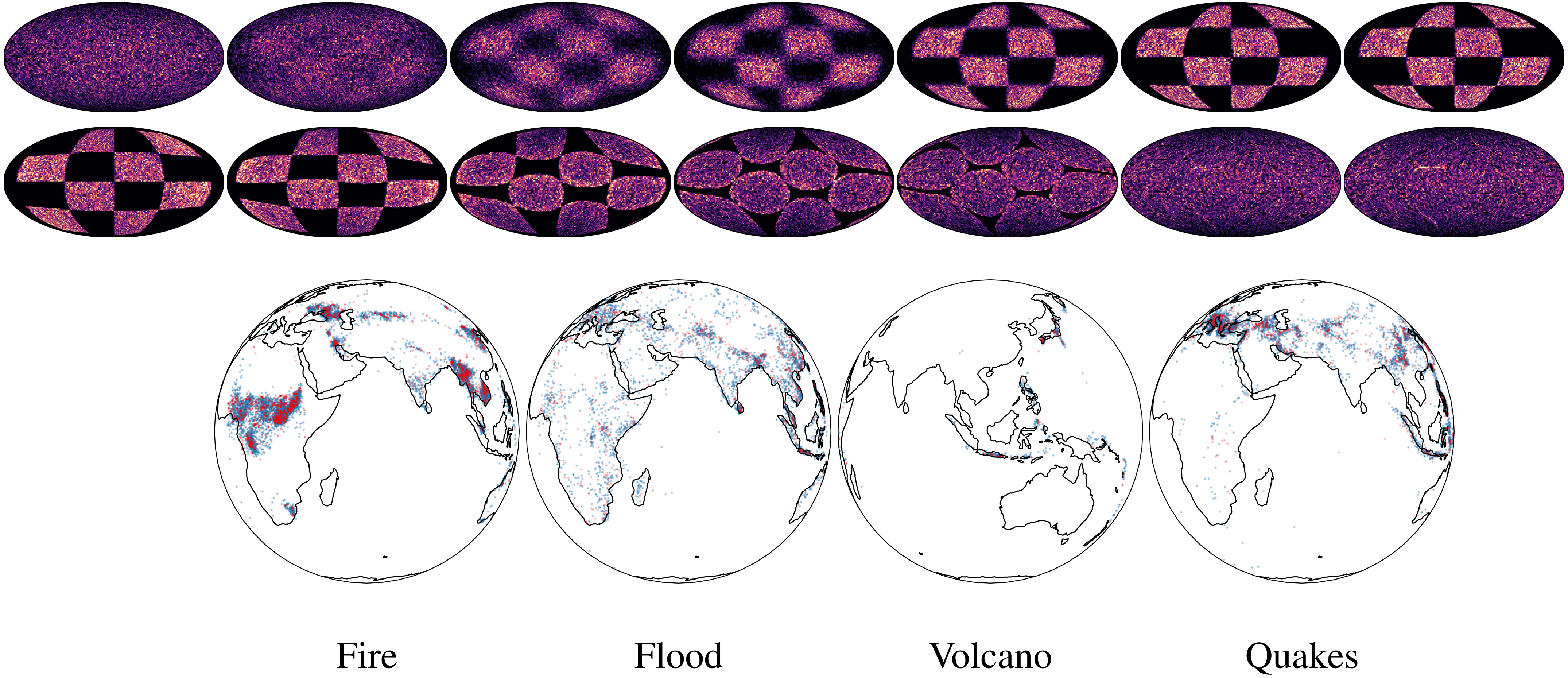


Optimal transport displacement



How to perform machine learning and optimization over non-Euclidean spaces?

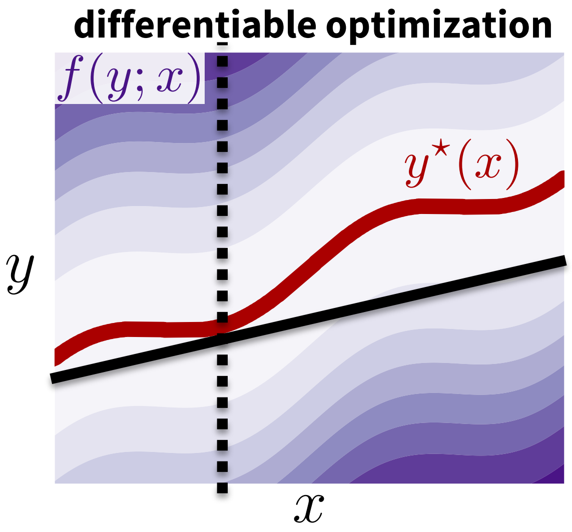
Matching Normalizing Flows and Probability Paths on Manifolds. Ben-Hamu et al., ICML 2022.

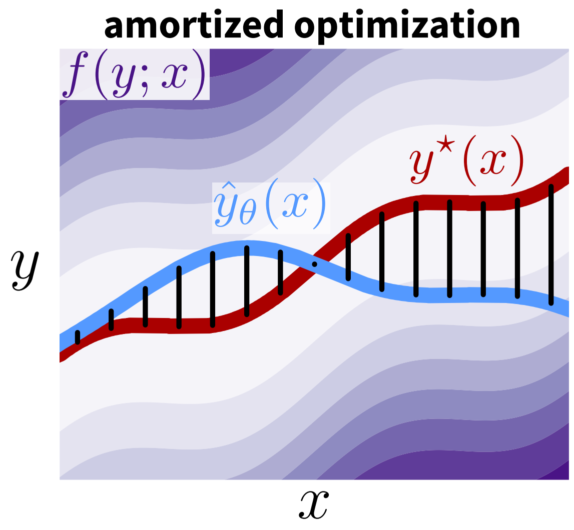


Summary

Optimization expresses **non-trivial reasoning operations**

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





Learning with differentiable and amortized optimization

Brandon Amos • Meta AI (FAIR) NYC

http://github.com/bamos/presentations

[ICML 2017] *Differentiable QPs: OptNet*
[ICML 2017] *Input-convex neural networks*
[NeurIPS 2017] *Differentiable Task-based Model Learning*
[NeurIPS 2018] *Differentiable MPC for End-to-end Planning and Control*
[ICLR 2018] *Learning Awareness Models*
[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: ∇higher*
[ICML 2020] *Differentiable Cross-Entropy Method*
[L4DC 2020] *Objective Mismatch in MBRL*
[MLCB 2020] *Neural Potts Model*
[ICML 2021] *Differentiable Combinatorial Optimization: CombOptNet*
[AISTATS 2021] *Gromov-DTW time series alignment*
[ICML 2021] *Riemannian Convex Potential Maps*
[L4DC 2021] *On the model-based stochastic value gradient*
[ICLR 2021] *Learning Neural Event Functions for ODEs*
[ICLR 2021] *Neural Spatio-Temporal Point Processes*
[NeurIPS 2021] *Online planning via RL amortization*
[ICML 2022] *Matching Flows and Probability Paths on Manifolds*
[NeurIPS 2022] *Theseus: Differentiable Nonlinear Optimization*
[NeurIPS 2022] *Differentiable Voronoi tessellation*
[NeurIPS 2022] *Nocturne self-driving benchmark*
[ICLR 2022] *Cross-Domain Imitation Learning via Optimal Transport*
[arXiv 2022] *Meta Optimal Transport*
[ICLR 2023] *On amortizing convex conjugates for optimal transport*
[L4DC 2023] *End-to-End Learning to Warm-Start for QPs*
[Foundations and Trends in ML 2023] *Tutorial on amortized optimization*

Collaborators: Akshay Agrawal, Andrew Gordon Wilson, Anselm Paulus, Arnaud Fickinger, Byron Boots, Denis Yarats, Edward Grefenstette, Eugene Vinitsky, Franziska Meier, Georg Martius, Giulia Luise, Heli Ben-Hamu, Hengyuan Hu, Ievgen Redko, Ivan Jimenez, Jacob Sacks, Jakob Foerster, Joseph Ortiz, Laurent Dinh, Luis Pineda, Marc Deisenroth, Maximilian Nickel, Michal Rolínek, Mikael Henaff, Misha Denil, Mustafa Mukadam, Nando de Freitas, Nathan Lambert, Noam Brown, Omry Yadan, Priya Donti, Ricky Chen, Roberto Calandra, Samuel Cohen, Samuel Stanton, Shane Barratt, Shobha Venkataraman, Soumith Chintala, Stephen Boyd, Steven Diamond, Stuart Russell, Tom Erez, Tom Sercu, Vít Musil, Xiaomeng Yang, Yann LeCun, Yaron Lipman, Yuval Tassa, Zeming Lin, Zico Kolter