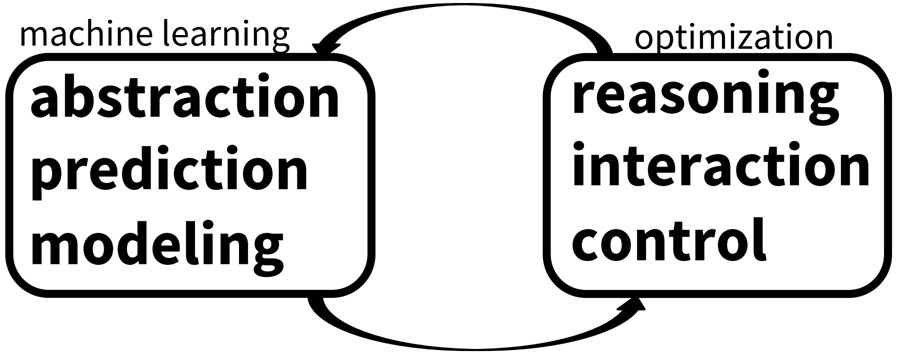
End-to-end learning geometries for graphs, dynamical systems, and regression

Brandon Amos • Meta AI, NYC

http://github.com/bamos/presentations

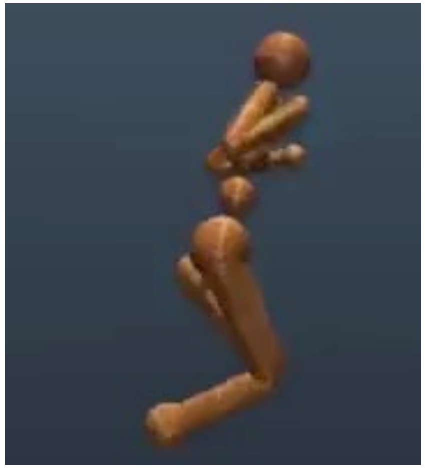
Joint work with Aaron Lou, Aram-Alexandre Pooladian, Carles Domingo-Enrich, Dishank Bansal, Maximilian Nickel, Mustafa Mukadam, Ricky Chen



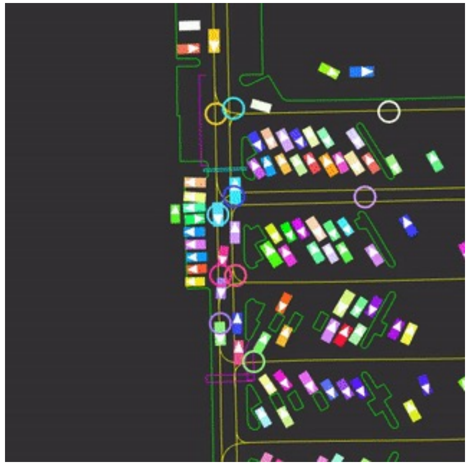
Disclaimer

My main research is on (mostly Euclidean) **optimization**, **control**, and **generative models**

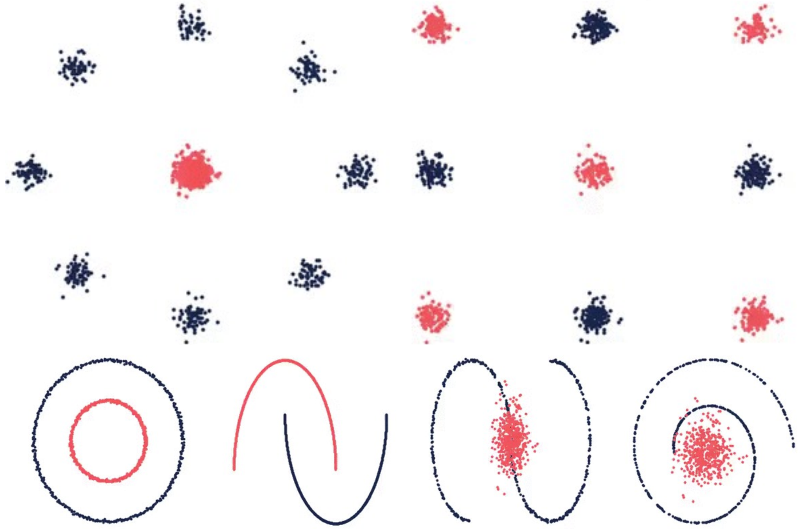
control



multi-agent dynamics



generative modeling and **optimal transport**



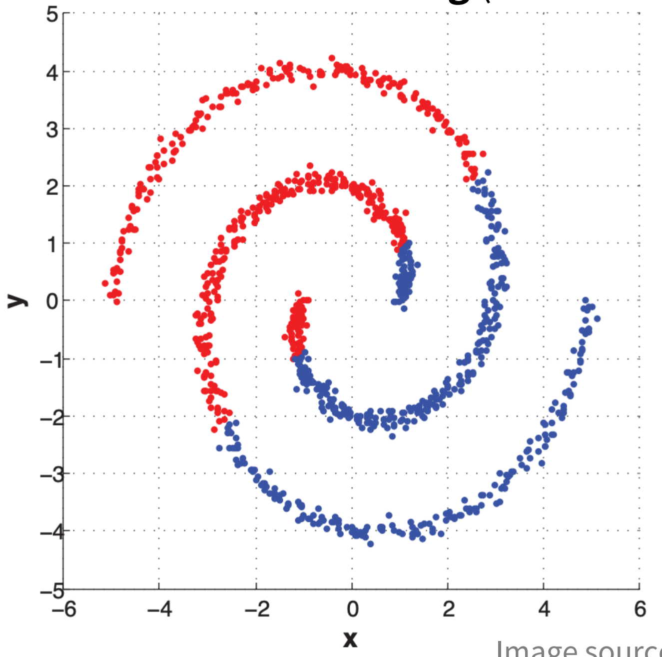
This talk: my perspective on intersections with learning graphs/geometry

Why learn geometries? (e.g., Manifolds, metrics, geodesic paths)

📚 *Machine learning, a probabilistic perspective.* Murphy, 2014.
📚 *Only Bayes should learn a manifold.* Hauberg, 2019.
📚 *Learning Riemannian Manifolds for Geodesic Motion Skills.* Beik-Mohammadi et al., RSS 2021.

Our focus: uncover underlying structure from high-dimensional data for predictions

k-means clustering (Euclidean)



spectral clustering (geometric)

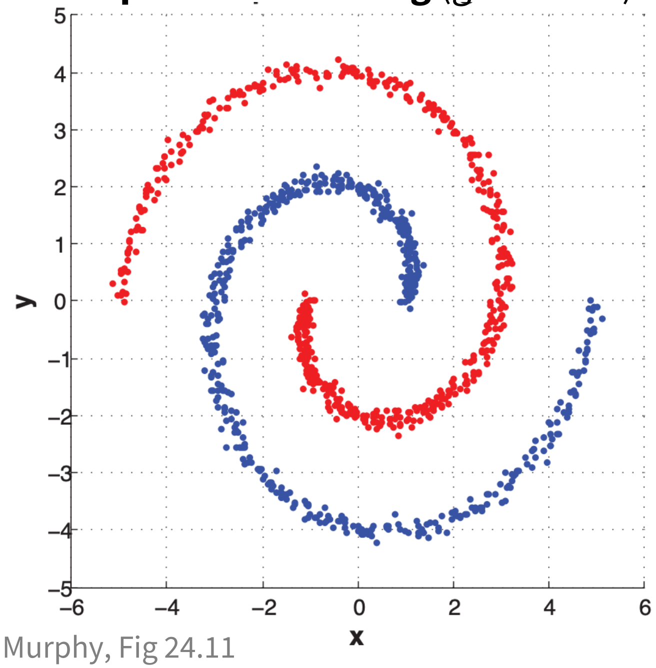


Image source: Murphy, Fig 24.11

Learning motion skills from demonstrations

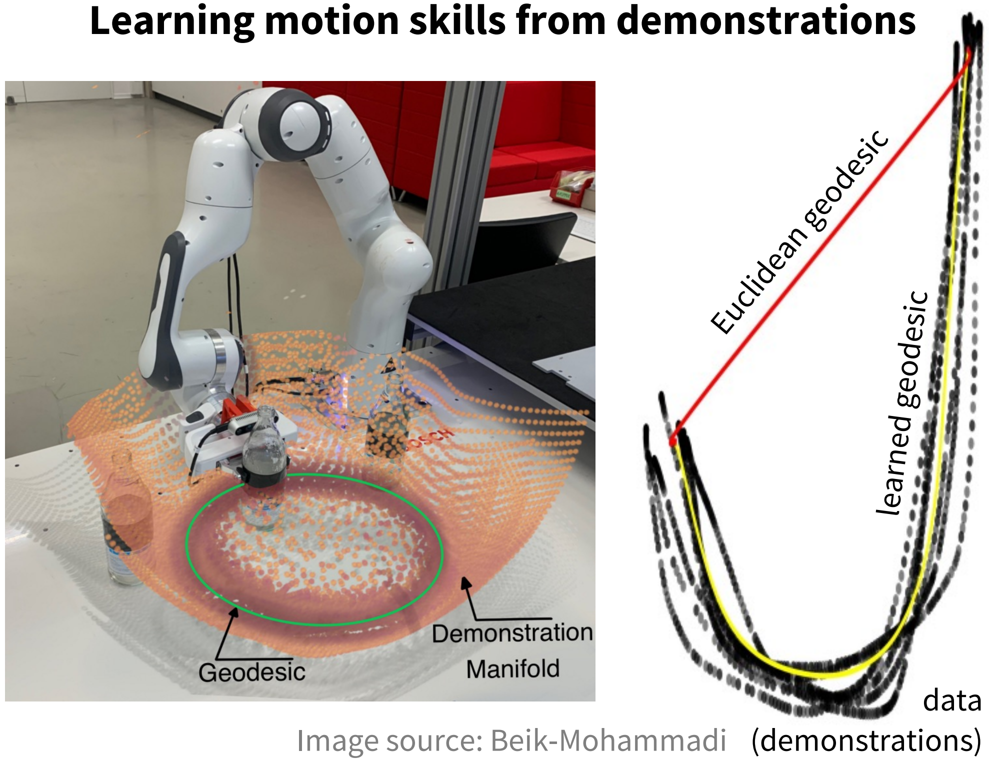


Image source: Beik-Mohammadi

How to learn geometries?

parameterize the geometry → measure how well it works

parameterize the geometry

1. metric (our focus)
2. surface
3. embeddings
4. latent models

measure how well it works

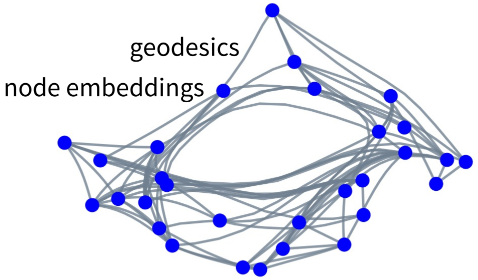
1. fits to the data (standard)
   a) geodesic distances
   b) geodesic paths
   c) surface reconstruction
2. works for a downstream task (our focus)

This talk: end-to-end learning geometries

1. graph embeddings

📚 *Deep Riemannian Manifold Learning.*
Lou, Nickel, Amos, NeurIPS 2020 Geo4dl workshop.

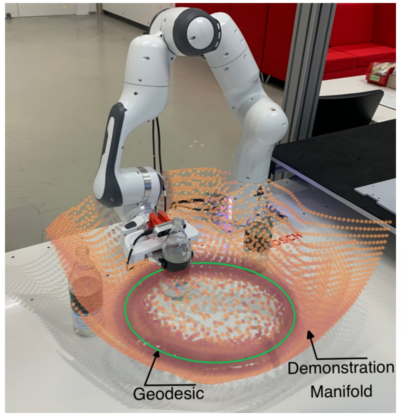
protein graph embedded in $\mathcal{M} = (\mathbb{R}^2, A_\theta)$



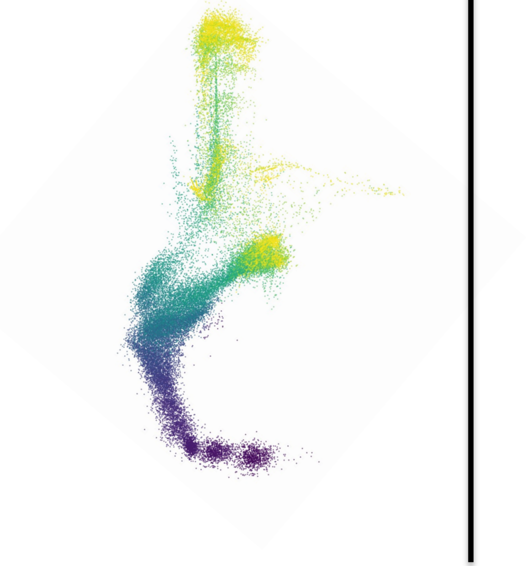
2. physical systems

📚 *Learning Riemannian Manifolds for Geodesic Motion Skills.*
Beik-Mohammadi et al., RSS 2021.
📚 *Riemannian Metric Learning via Optimal Transport.*
Scarvelis and Solomon, ICLR 2023.
📚 *Neural Optimal Transport with Lagrangian Costs.*
Pooladian, Domingo-Enrich, Chen, Amos, arXiv 2023.

robot demonstrations

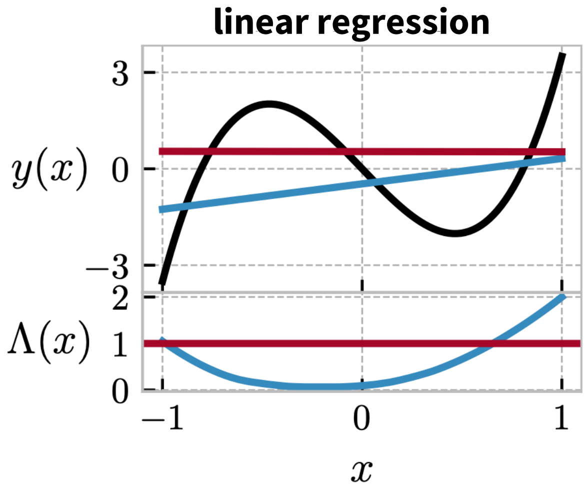


cell populations over time



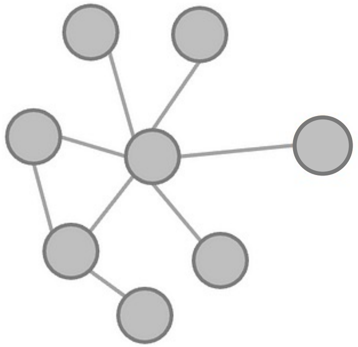
3. regression

📚 *TaskMet: Task-Driven Metric Learning for Model Learning.*
Bansal, Chen, Mukadam, Amos, NeurIPS 2023.

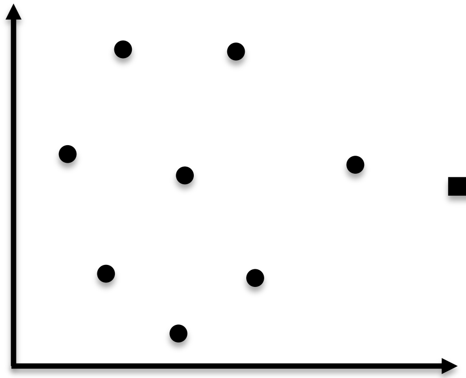


Graph embeddings

graph



node embeddings (e.g., in $\mathbb{R}^2$)



Learning: optimize embeddings to preserve distances (or some other loss)

downstream task

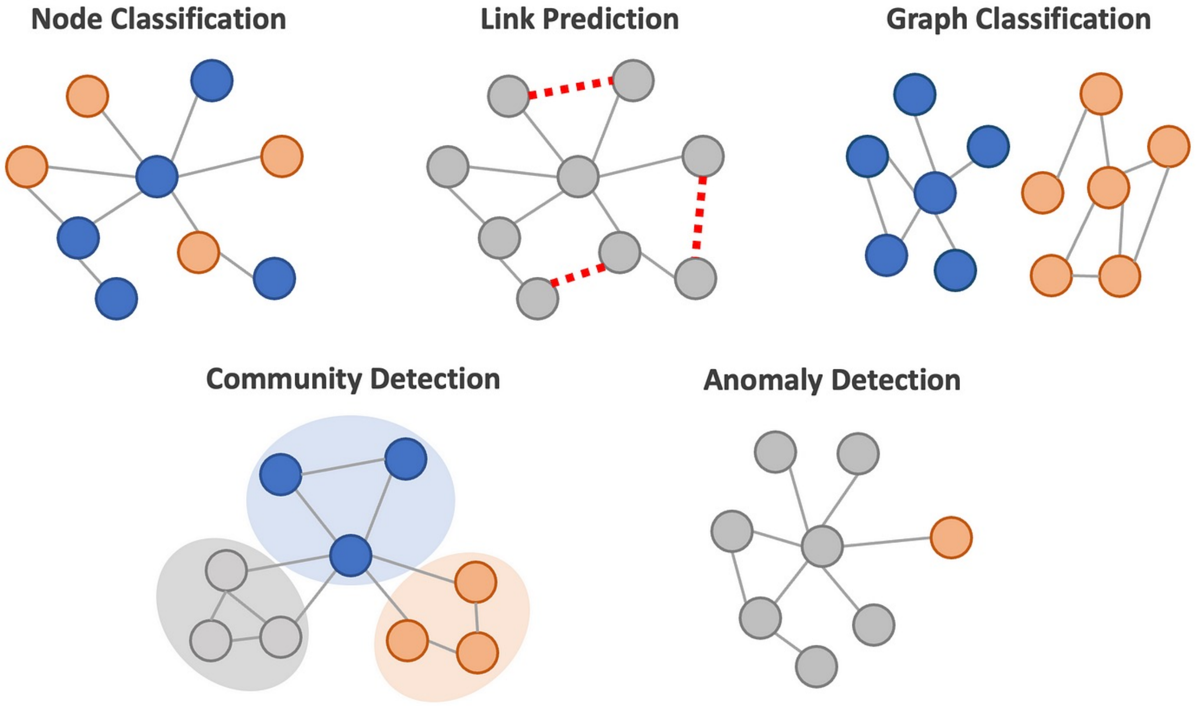
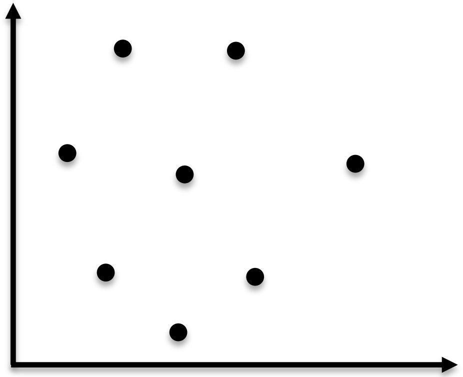


Image source: Masui

What’s the ideal geometry for the embeddings?

- *Spherical and hyperbolic embeddings of data.* Wilson et al., TPAMI 2014.
- *Poincaré Embeddings for Learning Hierarchical Representations.* Nickel and Kiela, NeurIPS 2017.
- *Learning mixed-curvature representations in products of model spaces*. Gu et al., ICLR 2019.

Euclidean space (general)



Spherical space (cycles)

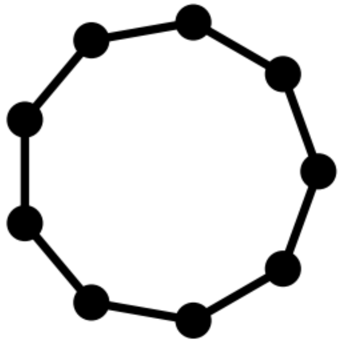


Image source: Gu et al.

Hyperbolic space (hierarchies)

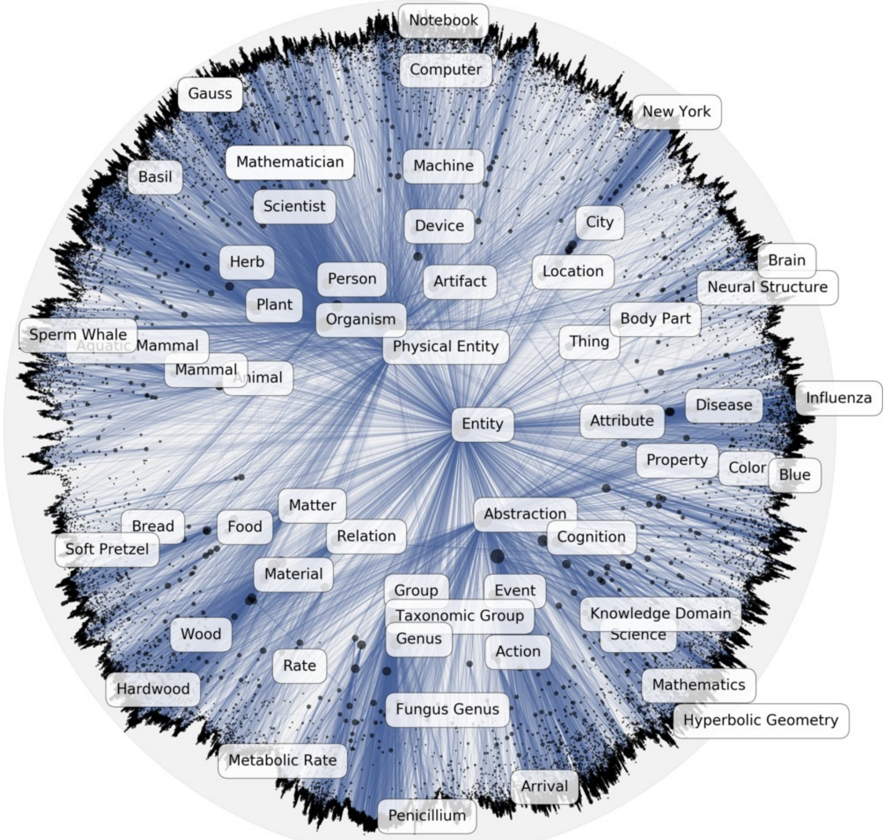
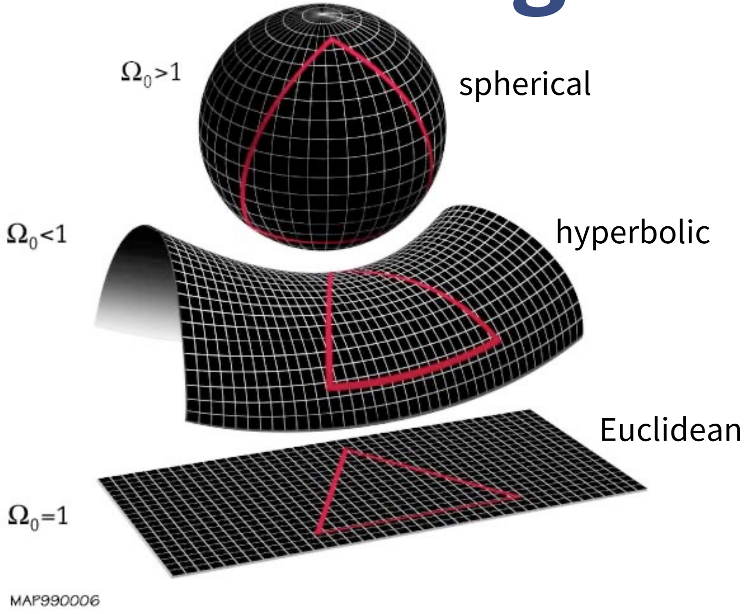


Image source: Nickel et al.



Product manifolds (mixtures)

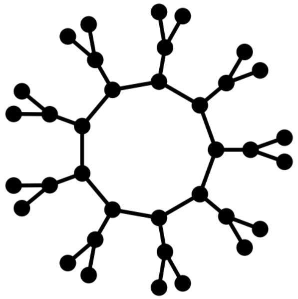


Image source: Gu et al.

Learning the embedding geometry

📚 *Deep Riemannian Manifold Learning.* Lou, Nickel, Amos, NeurIPS 2020 Geo4dl workshop.

Idea: fix space $\mathbb{R}^d$ equip with a parameterized *Riemannian metric* $A_\theta: \mathbb{R}^d \to \mathbb{S}^d$
learn θ from downstream task (e.g., distortion)

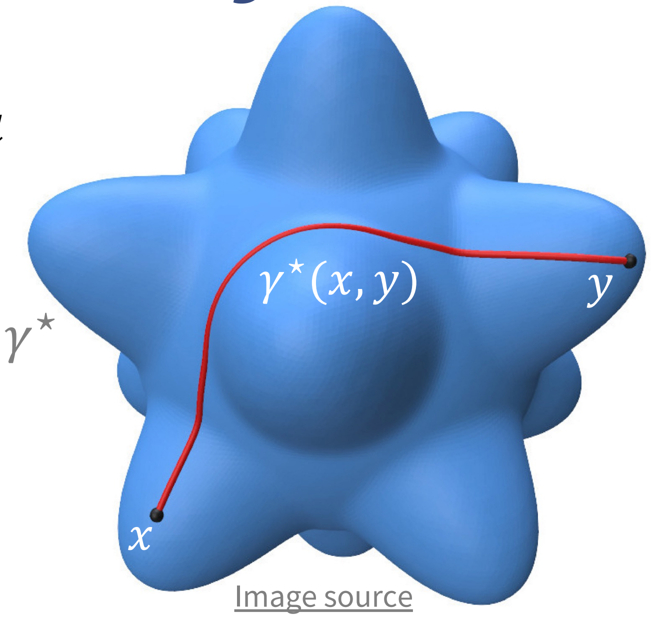


Image source

Geodesic distance given by $d_\theta(x, y) \triangleq \inf_{\gamma \in \mathcal{C}(x,y)} \int_0^1 \|\dot{\gamma}_t\|_{A_\theta(\gamma_t)}\, dt$ with minimizer $\gamma^\star$
⚠️ (numerically solved)

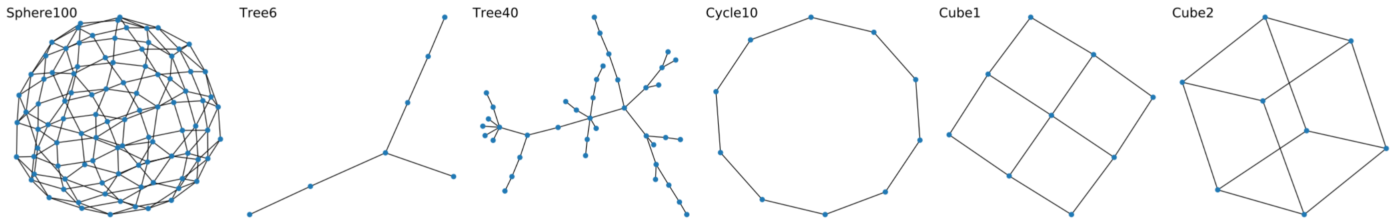
We can **differentiate the manifold operations** w.r.t. θ

Module 1 Pseudocode for Deep Riemannian Manifold Operations.

```
function ManifoldLog(x, y, θ)
    v := BVPSOLVE(x, y, GeoEq_θ)
    save x, y, θ and v for the backwards pass.
    return y
function LogBackwards(∇v)
    ŷ := ManifoldExp(x, v, GeoEq_θ)
    // Construct Jacobian Matrix J
    for i := 1 to n do
        _, j_i, _ = ExpBackwards(e_i)
    J := [j_1, ..., j_n]^⊤
    dx, _, dθ := ExpBackwards(∇v)
    ∇x, ∇θ := −(J^{−1})^⊤ dx, −(J^{−1})^⊤ dθ
    ∇y = −(J^{−1})^⊤ ∇v
    returns ∇x, ∇y, ∇θ

function ManifoldExp(x, v, θ)
    y := ODEINT(x, v, GeoEq_θ)
    save x, v, θ and y for the backwards pass.
    return y
function ExpBackwards(∇y)
    ∇x, ∇v, ∇θ := ODEINTBACKWARDS(∇y)
    return ∇x, ∇v, ∇θ

function ManifoldInterp(x, y, t, θ)
    v := ManifoldLog(x, y, θ)
    return ManifoldExp(x, tv, θ)

function ManifoldDistance(x, y, θ)
    return ||ManifoldLog(x, y, θ)||_{g_θ}
```

Graph embedding results

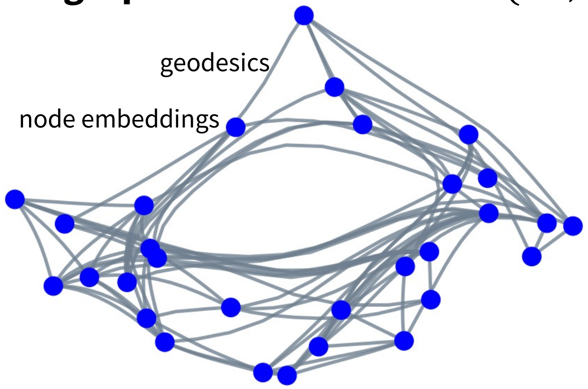
📚 *Deep Riemannian Manifold Learning.* Lou, Nickel, Amos, NeurIPS 2020 Geo4dl workshop.



	Sphere100			Tree6			Tree40		
	2	5	10	2	5	10	2	5	10
Euclidean	42.64 $\pm$ 21.88	2.85 $\pm$ 0.14	2.82 $\pm$ 0.08	3.14 $\pm$ 2.80	1.43 $\pm$ 0.07	1.51 $\pm$ 0.10	44.64 $\pm$ 19.64	7.85 $\pm$ 1.26	5.62 $\pm$ 0.39
Poincare Ball	12.11 $\pm$ 0.60	2.42 $\pm$ 0.04	2.64 $\pm$ 0.05	1.65 $\pm$ 0.04	1.63 $\pm$ 0.00	1.63 $\pm$ 0.00	8.57 $\pm$ 2.13	3.45 $\pm$ 0.12	2.38 $\pm$ 0.12
Lorentz	8.91 $\pm$ 2.25	2.35 $\pm$ 0.05	2.41 $\pm$ 0.19	1.63 $\pm$ 0.5	1.29 $\pm$ 0.05	1.29 $\pm$ 0.04	9.84 $\pm$ 2.56	2.95 $\pm$ 0.62	1.88 $\pm$ 0.21
Sphere	2.49 $\pm$ 0.00	2.11 $\pm$ 0.00	1.75 $\pm$ 0.01	1.69 $\pm$ 0.02	1.71 $\pm$ 0.01	1.71 $\pm$ 0.00	11.61 $\pm$ 0.94	6.83 $\pm$ 0.90	5.51 $\pm$ 0.23
Deep Manifold	102.74 $\pm$ 62.74	2.41 $\pm$ 0.01	2.51 $\pm$ 0.06	2.01 $\pm$ 0.80	1.13 $\pm$ 0.04	1.13 $\pm$ 0.05	71.57 $\pm$ 15.44	4.47 $\pm$ 0.65	2.51 $\pm$ 0.37

	Cycle10			Cube1			Cube2		
	2	5	10	2	5	10	2	5	10
Euclidean	1.67 $\pm$ 0.06	1.91 $\pm$ 0.12	1.79 $\pm$ 0.04	6.21 $\pm$ 1.27	1.83 $\pm$ 0.09	1.74 $\pm$ 0.16	4.37 $\pm$ 0.54	2.05 $\pm$ 0.07	1.98 $\pm$ 0.09
Poincare Ball	1.93 $\pm$ 0.00	1.93 $\pm$ 0.00	1.94 $\pm$ 0.00	1.80 $\pm$ 0.01	1.81 $\pm$ 0.01	1.81 $\pm$ 0.01	2.66 $\pm$ 0.31	2.17 $\pm$ 0.01	2.16 $\pm$ 0.01
Lorentz	1.72 $\pm$ 0.01	1.73 $\pm$ 0.00	1.72 $\pm$ 0.00	5.32 $\pm$ 0.25	1.58 $\pm$ 0.02	1.62 $\pm$ 0.05	4.66 $\pm$ 0.48	2.23 $\pm$ 0.12	2.21 $\pm$ 0.02
Sphere	1.31 $\pm$ 0.00	1.24 $\pm$ 0.00	1.21 $\pm$ 0.00	1.78 $\pm$ 0.05	1.76 $\pm$ 0.01	1.76 $\pm$ 0.01	2.48 $\pm$ 0.04	2.15 $\pm$ 0.05	2.10 $\pm$ 0.01
Deep Manifold	8.23 $\pm$ 5.69	1.77 $\pm$ 0.55	1.34 $\pm$ 0.07	5.99 $\pm$ 1.49	1.55 $\pm$ 0.15	1.53 $\pm$ 0.09	3.69 $\pm$ 1.02	1.65 $\pm$ 0.03	1.58 $\pm$ 0.05

Protein graph embedded in $\mathcal{M} = (\mathbb{R}^2, A_\theta)$



dim	$\mathcal{M}$	Protein	Hamiltonian	Social
5	$\mathbb{E}^5$	4.259 $\pm$ 0.721	5.512 $\pm$ 0.494	2.963 $\pm$ 0.279
	$\mathbb{H}^5$	2.228 $\pm$ 0.033	3.969 $\pm$ 0.511	2.329 $\pm$ 0.083
	$\mathbb{S}^5$	2.940 $\pm$ 0.373	5.596 $\pm$ 0.677	4.195 $\pm$ 0.440
	$\mathbb{E}^2 \times \mathbb{H}^3$	6.629 $\pm$ 0.607	6.545 $\pm$ 1.687	4.241 $\pm$ 0.290
	$\mathbb{E}^2 \times \mathbb{S}^3$	8.684 $\pm$ 0.784	6.184 $\pm$ 0.389	5.484 $\pm$ 1.825
	$\mathbb{H}^2 \times \mathbb{S}^3$	6.851 $\pm$ 0.926	6.414 $\pm$ 0.785	5.643 $\pm$ 1.638
	Deep	2.008 $\pm$ 0.214	3.724 $\pm$ 0.315	2.212 $\pm$ 0.053
10	$\mathbb{E}^{10}$	3.380 $\pm$ 0.347	3.314 $\pm$ 0.238	2.359 $\pm$ 0.078
	$\mathbb{H}^{10}$	2.178 $\pm$ 0.020	2.718 $\pm$ 0.029	2.229 $\pm$ 0.046
	$\mathbb{S}^{10}$	2.219 $\pm$ 0.026	3.485 $\pm$ 0.218	2.834 $\pm$ 0.355
	$\mathbb{E}^5 \times \mathbb{H}^5$	3.339 $\pm$ 0.566	2.964 $\pm$ 0.565	2.523 $\pm$ 0.234
	$\mathbb{E}^5 \times \mathbb{S}^5$	3.621 $\pm$ 0.757	4.117 $\pm$ 0.251	3.423 $\pm$ 0.806
	$\mathbb{H}^5 \times \mathbb{S}^5$	2.709 $\pm$ 0.252	4.318 $\pm$ 0.391	3.506 $\pm$ 0.350
	$\mathcal{S}_4^+$	3.498 $\pm$ 0.239	3.824 $\pm$ 0.599	2.503 $\pm$ 0.163
	$\mathrm{Gr}(5, 2)$	5.088 $\pm$ 1.338	5.228 $\pm$ 0.366	2.885 $\pm$ 0.113
	Deep	1.958 $\pm$ 0.023	2.668 $\pm$ 0.218	2.152 $\pm$ 0.051

Scaling issues beyond ~10 dimensions

📚 *Deep Riemannian Manifold Learning.* Lou, Nickel, Amos, NeurIPS 2020 Geo4dl workshop.

Computing high-dimensional geodesics in a continuous space is hard

$$d_\theta(x,y) \triangleq \inf_{\gamma \in \mathcal{C}(x,y)} \int_0^1 \|\dot{\gamma}_t\|_{A_\theta(\gamma_t)}\, dt$$

(very easy and scalable on, e.g., Euclidean, hyperbolic, spherical, and mixture spaces)

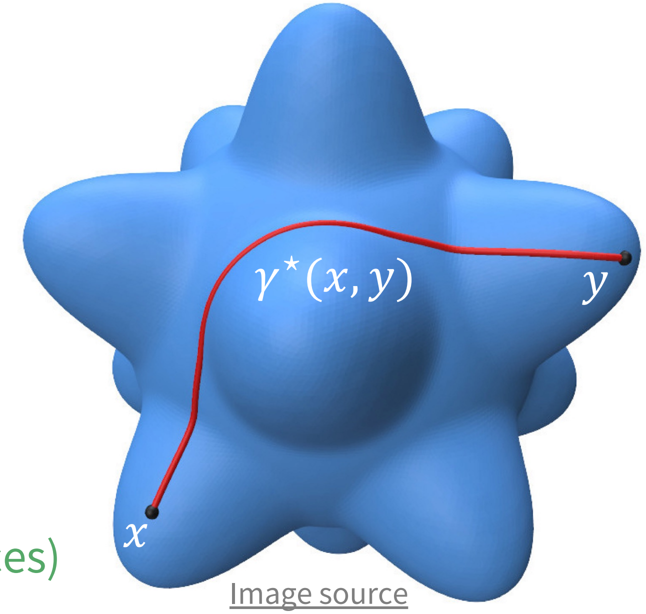


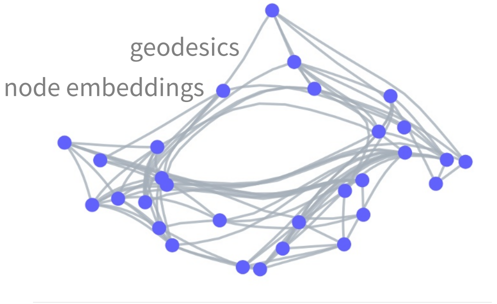
Image source

This talk: end-to-end learning geometries

1. graph embeddings

📚 *Deep Riemannian Manifold Learning.*
Lou, Nickel, Amos, NeurIPS 2020 Geo4dl workshop.

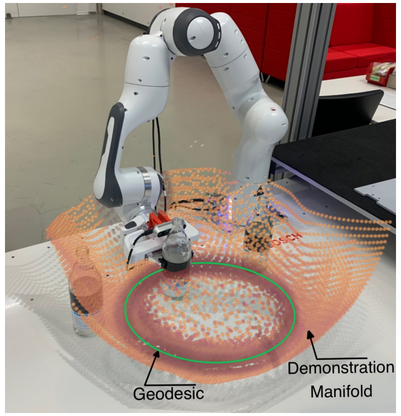
protein graph embedded in $\mathcal{M} = (\mathbb{R}^2, A_\theta)$



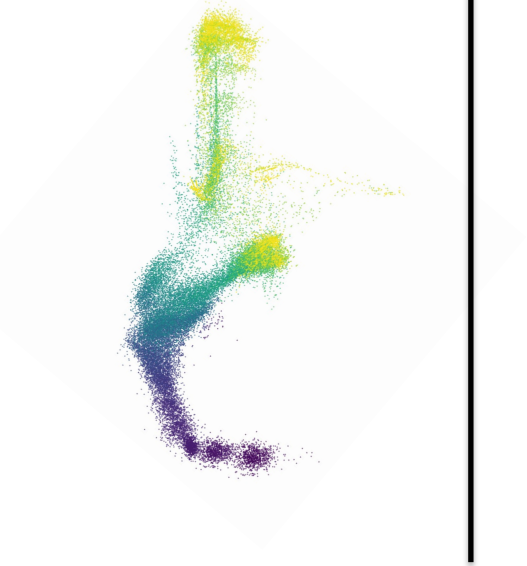
2. physical systems

📚 *Learning Riemannian Manifolds for Geodesic Motion Skills.*
Beik-Mohammadi et al., RSS 2021.
📚 *Riemannian Metric Learning via Optimal Transport.*
Scarvelis and Solomon, ICLR 2023.
📚 *Neural Optimal Transport with Lagrangian Costs.*
Pooladian, Domingo-Enrich, Chen, Amos, arXiv 2023.

robot demonstrations

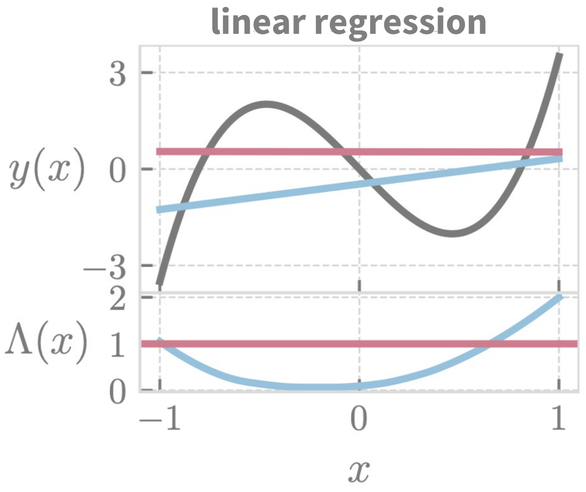


cell populations over time



3. regression

📚 *TaskMet: Task-Driven Metric Learning for Model Learning.*
Bansal, Chen, Mukadam, Amos, NeurIPS 2023.



Modeling dynamical systems

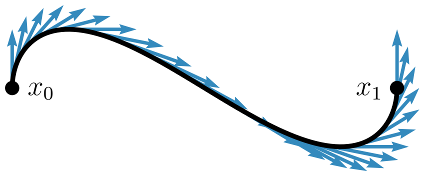
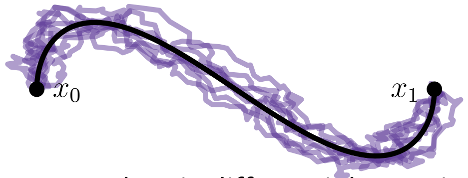
> *Mechanics is the paradise of the mathematical sciences, because by means of it one comes to the fruits of mathematics.*
> da Vinci (1459-1519), Notebooks, v. 1, ch. 20.

Quote also given at the beginning of *Geometric Control of Mechanical Systems*, Bullo and Lewis, 2000.

1. Make observations

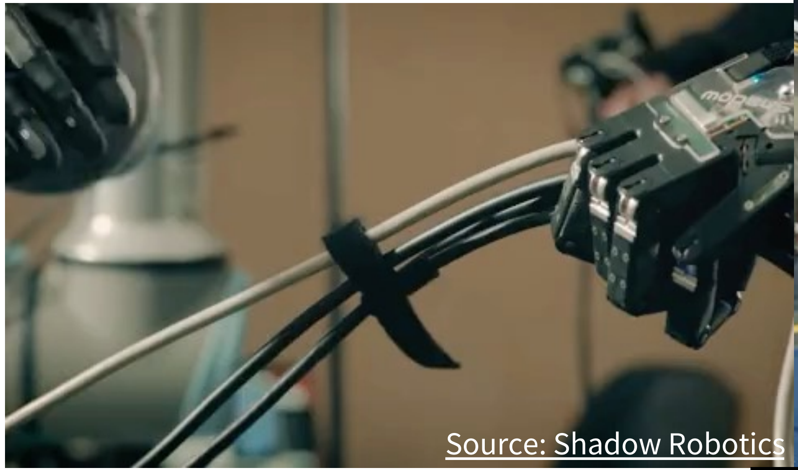
Learning Neural Event Functions for Ordinary Differential Equations.
Ricky T. Q. Chen, Brandon Amos, Maximilian Nickel, ICLR 2021.

2. Come up with a theory

	Continuous time	Discrete time
Deterministic	$\dot{x}_t = f(x_t)$  e.g., differential equations (ODE/PDEs)	$x_{t+1} = f(x_t)$ e.g., Turing machines, games
Stochastic	$\mathrm{d}x_t = f(x_t)\mathrm{d}t + F(x_t)\mathrm{d}B_t$  e.g., stochastic differential equations	$x_{t+1} = f(x_t, w_t)$ $w_t \sim p(w)$ e.g., Markov chains

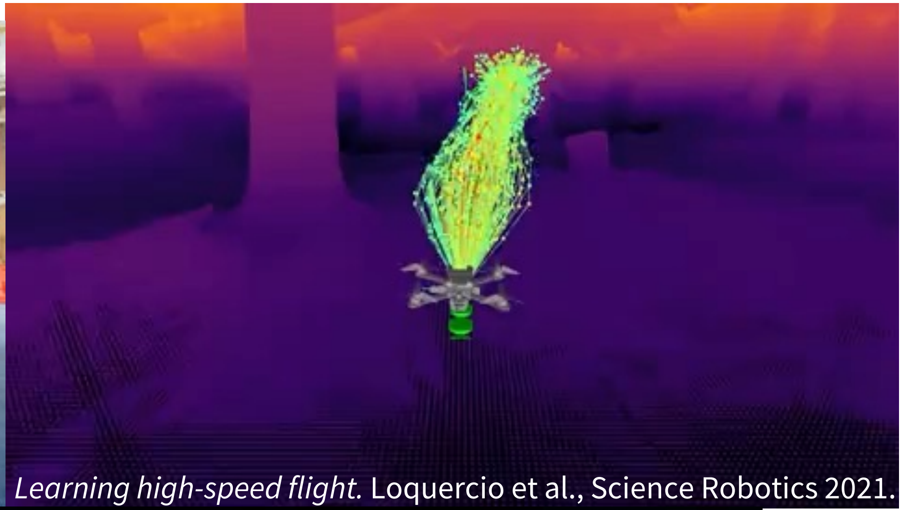
Controlled dynamical systems and robotics

often via the **Newton-Euler** equations of motion $M(q_t)\ddot{q}_t + n(q_t, \dot{q}_t) = \tau(q_t) + Bu_t$



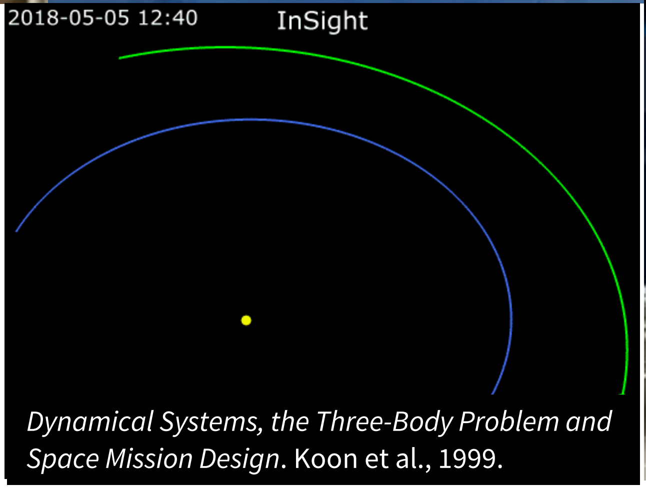
Source: Shadow Robotics

Source: Boston Dynamics



Learning high-speed flight. Loquercio et al., Science Robotics 2021.

Source: SpaceX



Dynamical Systems, the Three-Body Problem and Space Mission Design. Koon et al., 1999.

Source: Waymo

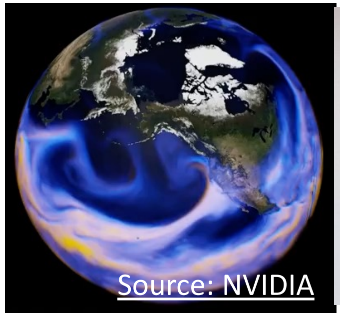
Machine learning way of learning dynamics

1. Collect data of the system 2. Throw neural networks at it

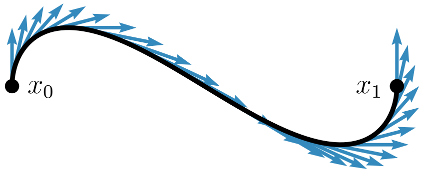
Continuous time

Discrete time

Deterministic



$$\dot{x}_t = f_\theta(x_t)$$ a neural network



e.g., Neural ODEs/PDEs, neural operators

FourCastNet, Pathak et al., 2022.

Learning Neural Constitutive Laws. Ma et al., ICML 2023.

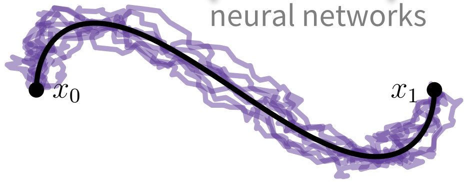
$$x_{t+1} = f_\theta(x_t)$$ a neural network

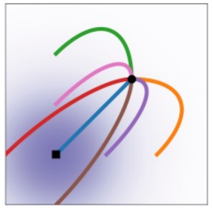
e.g., RNNs, LSTMs, Transformers
for language and other discrete-time sequential data

Stochastic

$$\mathrm{d}x_t = f_\theta(x_t)\mathrm{d}t + F_\theta(x_t)\mathrm{d}B_t$$

neural networks





e.g., Neural SDEs, diffusion models, flow matching

Deep unsupervised learning using nonequilibrium thermodynamics. Sohl-Dickstein et al., ICML 2015.
Score-Based Generative Modeling through Stochastic Differential Equations. Song et al., ICLR 2021.
Flow Matching for Generative Modeling. Lipman et al., ICLR 2023.
Stochastic Interpolants. Albergo et al., ICLR 2023.

$$x_{t+1} = f_\theta(x_t, w_t)$$
$$w_t \sim p_\theta(w)$$
neural networks

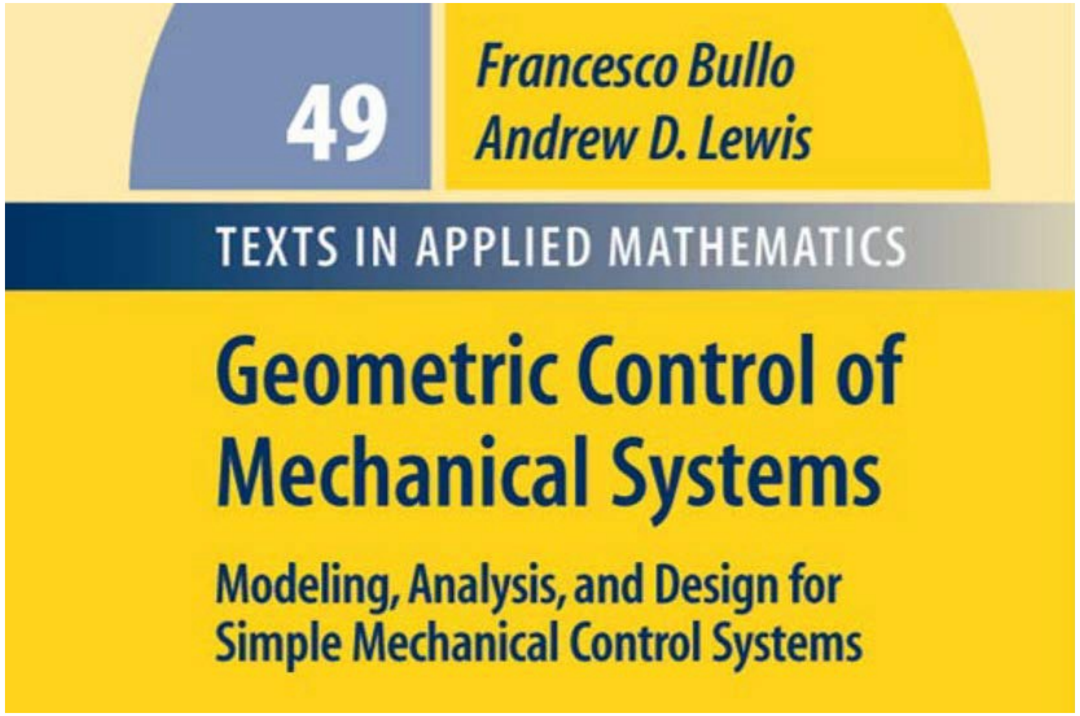
e.g., RNNs with stochastic states

A Recurrent Latent Variable Model for Sequential Data. Chung et al., NeurIPS 2015.
Sequential Neural Models with Stochastic Layers. Fraccaro et al., NeurIPS 2016.

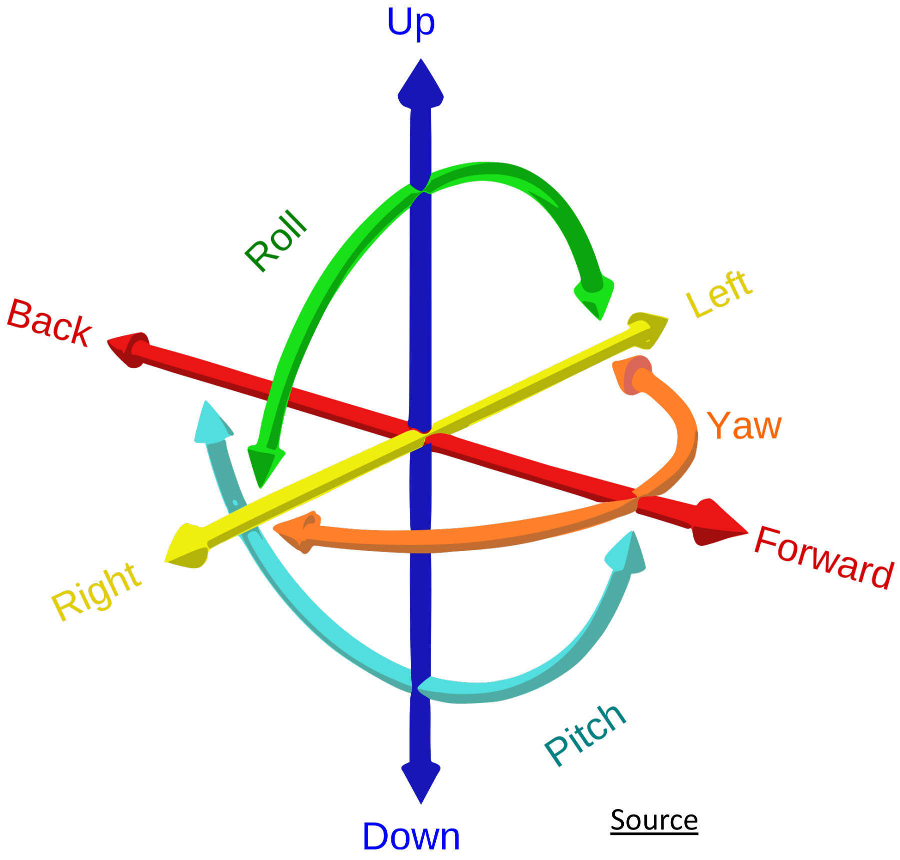
From Euclidean to geometric systems

why? modeling rotations, symmetries, obstacles, other parts of the open physical world

$$\dot{x}_t = f(x_t) \quad \text{where } x_t \in \mathcal{M},\ \dot{x}_t \in \mathcal{T}_{x_t}\mathcal{M}$$



degrees of freedom of a rigid body in 3D space



Source

What if we don't know the geometry? Learn it!

setting 1: geometry is the demonstration manifold

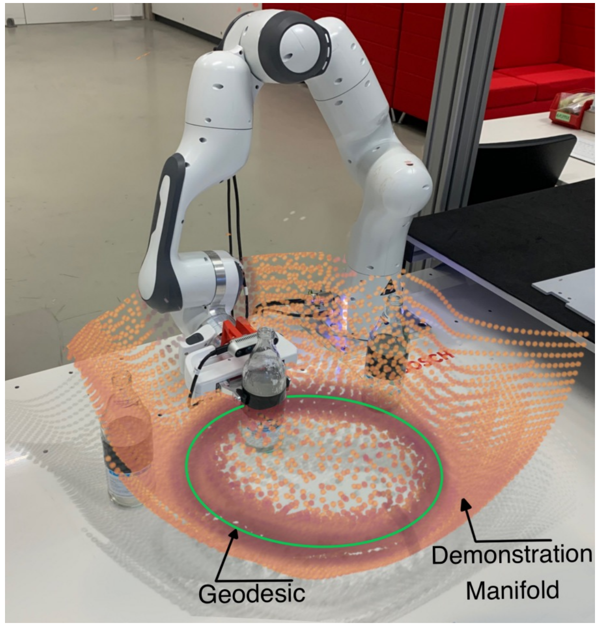
📚 *Learning Riemannian Manifolds for Geodesic Motion Skills.* Beik-Mohammadi et al., RSS 2021.

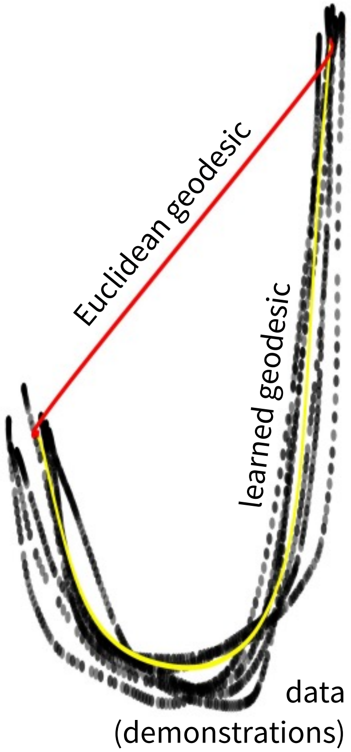
1. fit a VAE to the observations
2. induce a metric from it
3. compute geodesics under the induced metric

$$f_{\boldsymbol{\theta}}(\boldsymbol{z}) = \mu_{\boldsymbol{\theta}}(\boldsymbol{z}) + \mathrm{diag}(\epsilon)\sigma_{\boldsymbol{\theta}}(\boldsymbol{z}), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_D).$$

$$\bar{\boldsymbol{M}}(\boldsymbol{z}) = \boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}(\boldsymbol{z}) + \boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}(\boldsymbol{z})$$

$$d_\theta(x, y) \triangleq \inf_{\gamma \in \mathcal{C}(x,y)} \int_0^1 \frac{1}{2} \|\dot{\gamma}_t\|_{M_\theta(\gamma_t)} \, dt$$





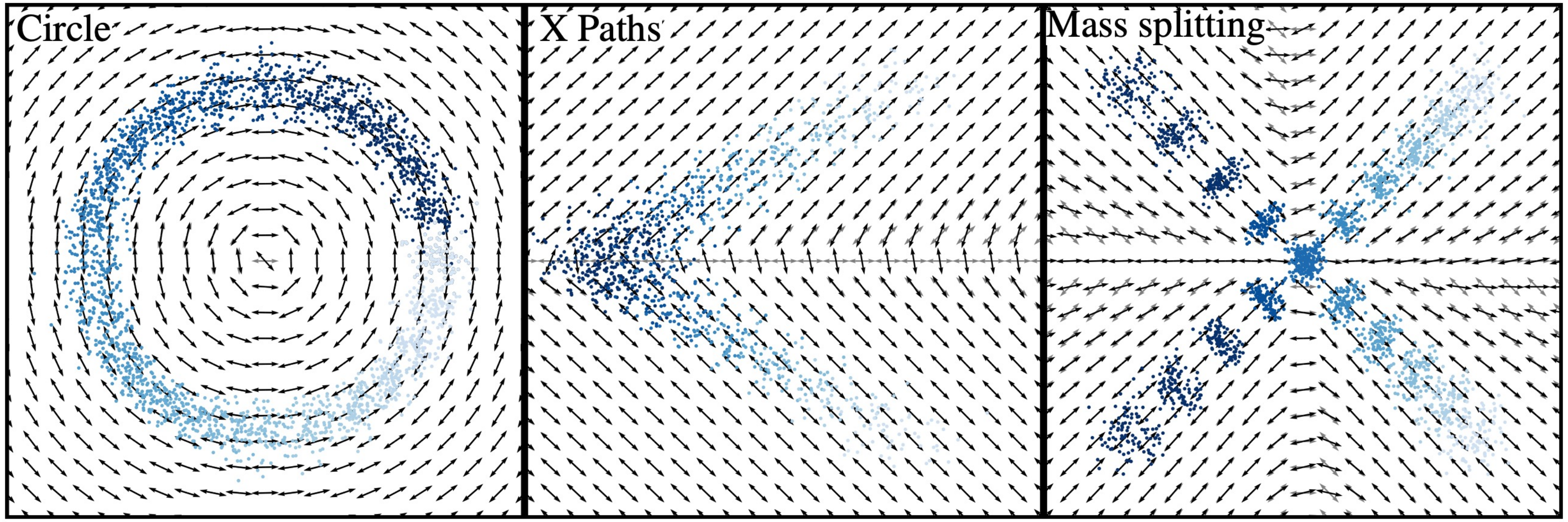
geodesic solver: with splines (usually in 2 or 3 dimensions)

What if we don't know the geometry? Learn it!

setting 2: geometry is in the state space (particle system with unpaired data)

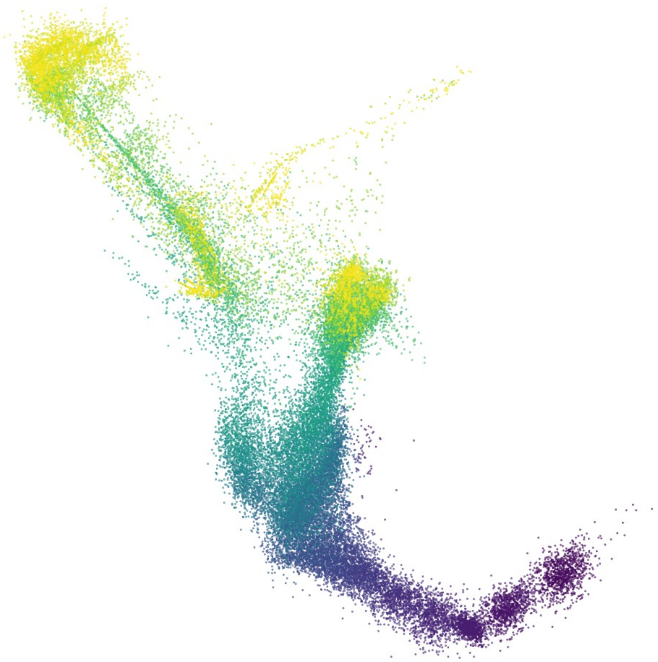
📚 *Riemannian Metric Learning via Optimal Transport.* Scarvelis and Solomon, ICLR 2023.
📚 *Neural Optimal Transport with Lagrangian Costs.* Pooladian, Domingo-Enrich, Chen, Amos, arXiv 2023.

synthetic data



(smallest) eigenvectors of A (■ learned metric ■ ground-truth) ■ data (lighter colors=later time)

populations of cells over time



What if we don't know the geometry? Learn it!

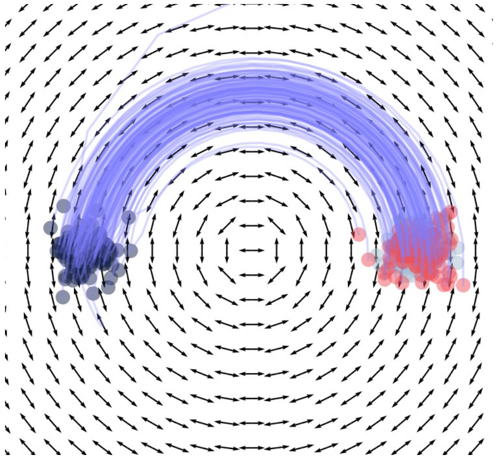
setting 2: geometry is in the state space (particle system with unpaired data)

📚 *Riemannian Metric Learning via Optimal Transport.* Scarvelis and Solomon, ICLR 2023.
📚 *Neural Optimal Transport with Lagrangian Costs.* Pooladian, Domingo-Enrich, Chen, Amos, arXiv 2023.

parameterize the geometry → solve OT problem in that geometry → improve the metric

via a metric A

so the OT cost is lower



$$\mathrm{OT}_c(\mu,\nu) := \inf_{\pi\in\Gamma(\mu,\nu)} \iint_{\mathcal{X}\times\mathcal{Y}} c(x,y)\,\mathrm{d}\pi(x,y)\,,$$

$$c(x,y) := \inf_{\gamma\in\mathcal{C}(x,y)} \left\{ \int_0^1 \mathcal{L}(\gamma_t,\dot{\gamma}_t,t)\,\mathrm{d}t \right\}$$

$$\mathcal{L}(x,v,t) := \frac{1}{2}\|v\|^2_{A(x)} = \frac{1}{2}v^\top A(x)v,$$

What if we don’t know the geometry? Learn it!

setting 2: geometry is in the state space (particle system with unpaired data)

📚 *Riemannian Metric Learning via Optimal Transport.* Scarvelis and Solomon, ICLR 2023.
📚 *Neural Optimal Transport with Lagrangian Costs.* Pooladian, Domingo-Enrich, Chen, Amos, arXiv 2023.

Table 1. Alignment scores ℓ_{align} for metric recovery in Fig. 4. (higher is better)

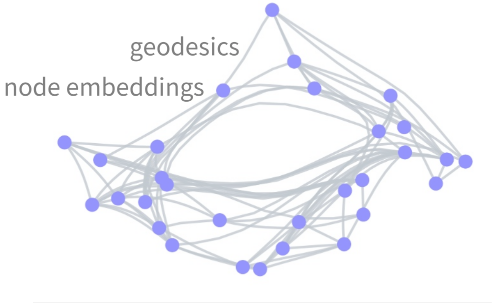
	Circle	Mass Splitting	X Paths
Scarvelis and Solomon (2023)	0.995	0.839	0.916
Our approach	$\mathbf{0.997 \pm 0.002}$	$\mathbf{0.986 \pm 0.001}$	$\mathbf{0.957 \pm 0.001}$

This talk: end-to-end learning geometries

1. graph embeddings

📚 *Deep Riemannian Manifold Learning.*
Lou, Nickel, Amos, NeurIPS 2020 Geo4dl workshop.

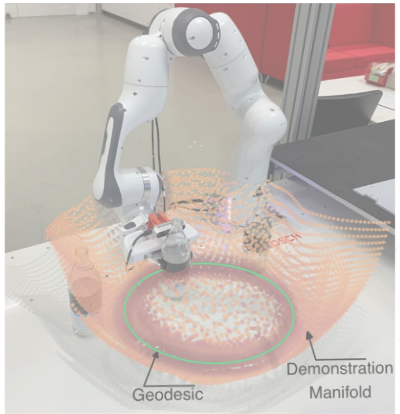
protein graph embedded in $\mathcal{M} = (\mathbb{R}^2, A_\theta)$



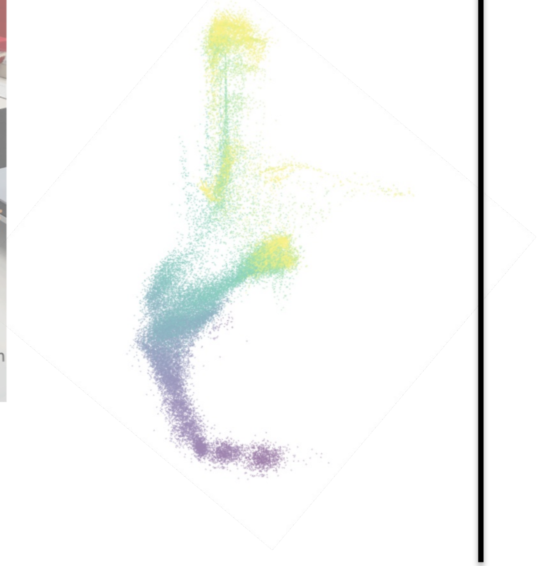
2. physical systems

📚 *Learning Riemannian Manifolds for Geodesic Motion Skills.*
Beik-Mohammadi et al., RSS 2021.
📚 *Riemannian Metric Learning via Optimal Transport.*
Scarvelis and Solomon, ICLR 2023.
📚 *Neural Optimal Transport with Lagrangian Costs.*
Pooladian, Domingo-Enrich, Chen, Amos, arXiv 2023.

robot demonstrations



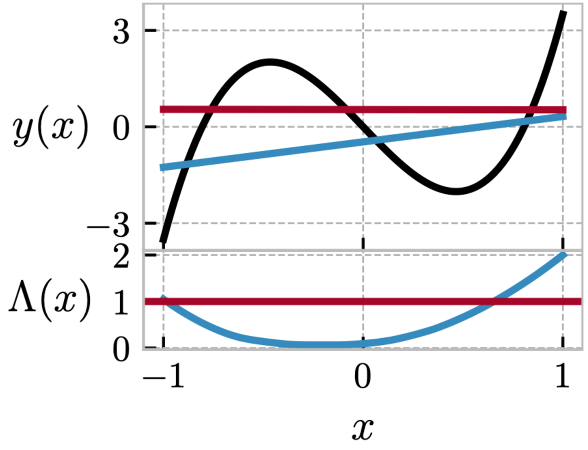
cell populations over time



3. regression

📚 *TaskMet: Task-Driven Metric Learning for Model Learning.*
Bansal, Chen, Mukadam, Amos, NeurIPS 2023.

linear regression

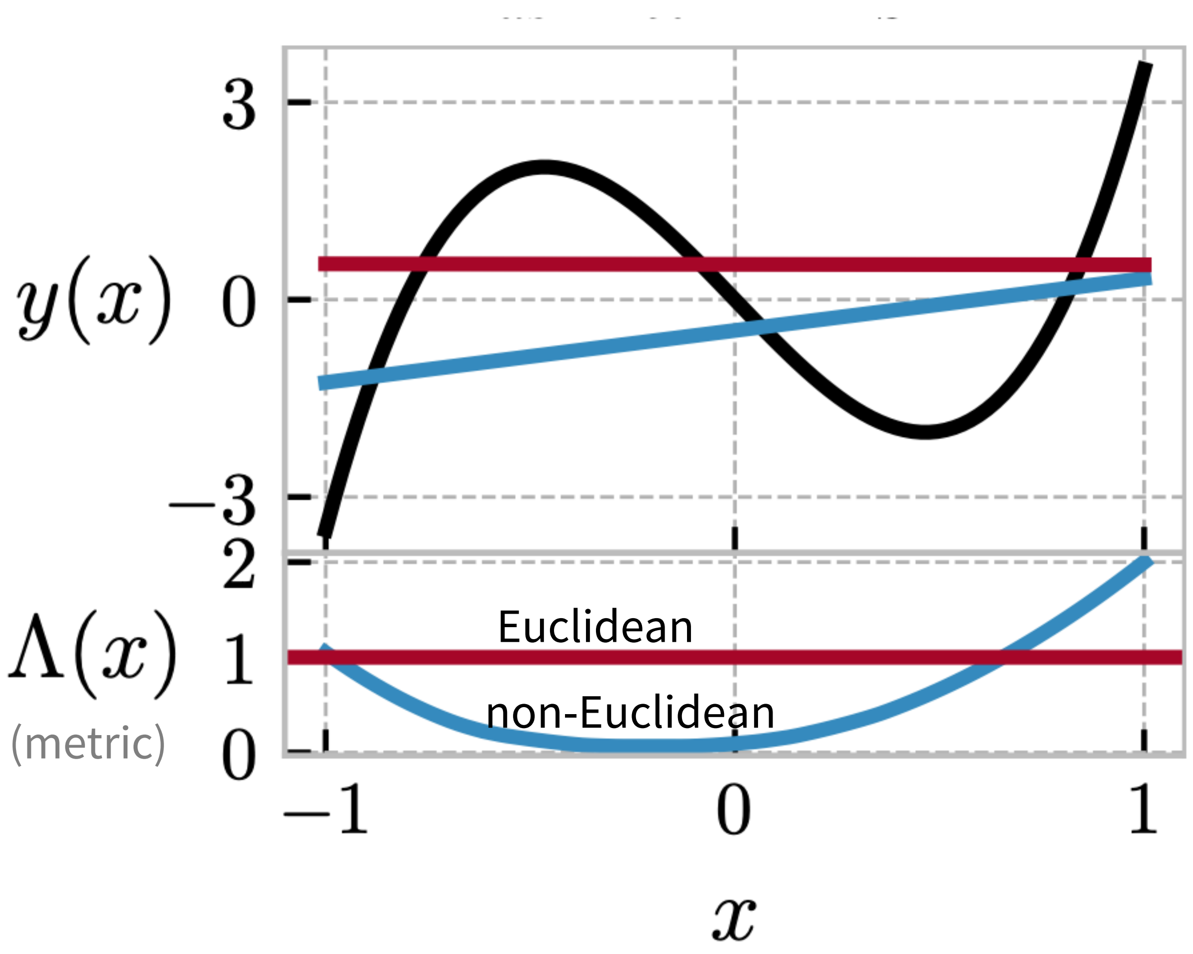


The geometry of the prediction space

📚 *TaskMet: Task-Driven Metric Learning for Model Learning*. Bansal, Chen, Mukadam, Amos, NeurIPS 2023.

linear regression

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \|f_\theta(x) - y\|^2_{\Lambda(x)}$$



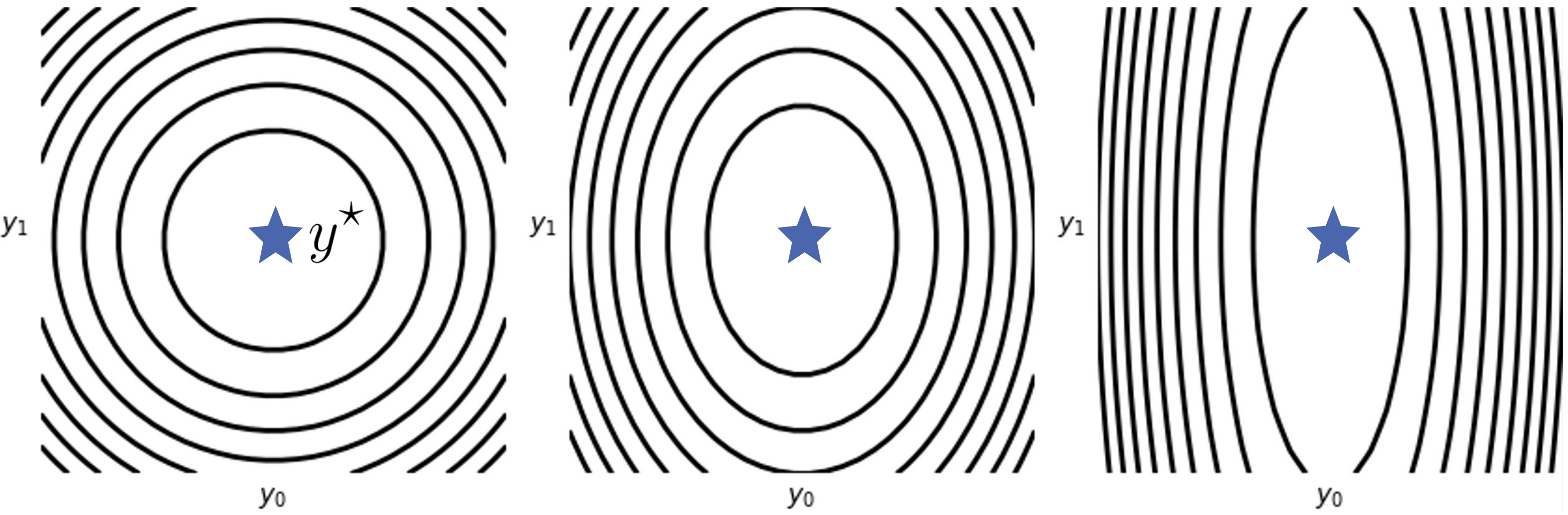
Why learn the prediction space geometry?

📚 *TaskMet: Task-Driven Metric Learning for Model Learning*. Bansal, Chen, Mukadam, Amos, NeurIPS 2023.

Reason 2. Emphasize predictions for **class 1 over 2**

Reason 3. Emphasize predictions for **image A over B**

Reason 1. Emphasize predictions for y_1 **by making the loss higher**

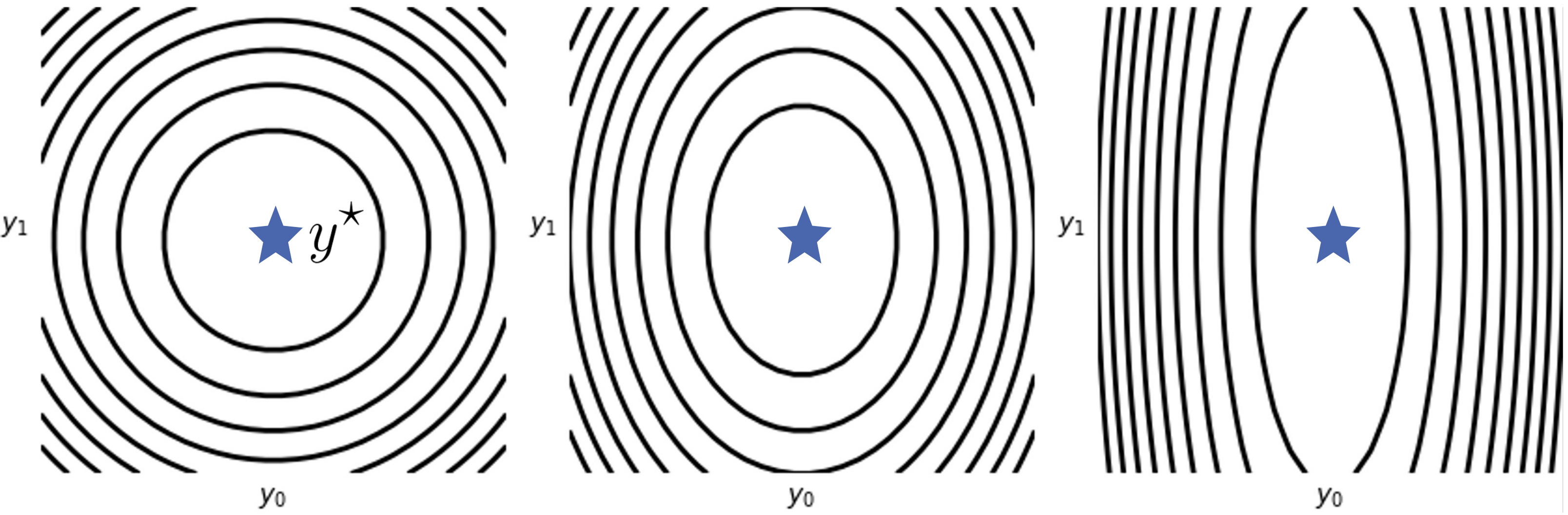


Why learn the prediction space geometry?

📚 *TaskMet: Task-Driven Metric Learning for Model Learning*. Bansal, Chen, Mukadam, Amos, NeurIPS 2023.

⚠️ **problem:** where does the geometry come from?

Reason 1. Emphasize predictions for y_1 **by making the loss higher**

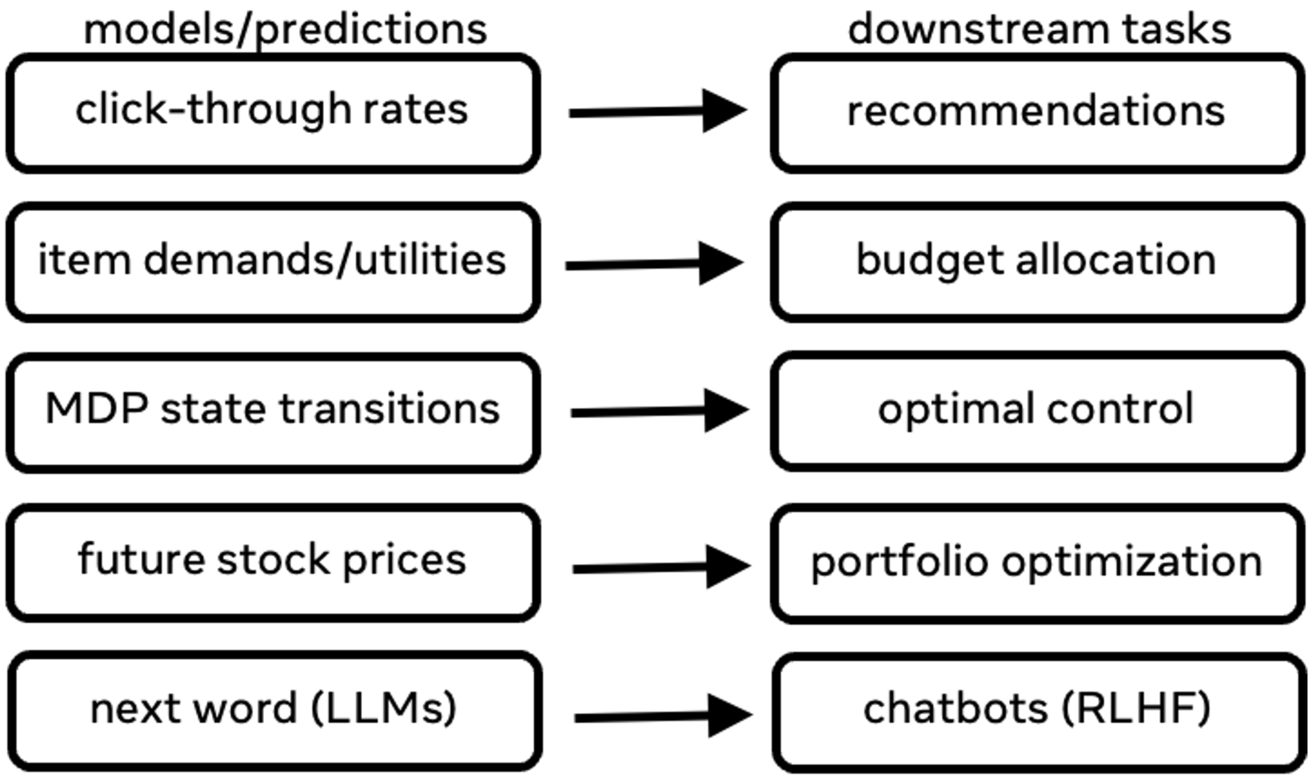


Reason 2. Emphasize predictions for **class 1 over 2**

Reason 3. Emphasize predictions for **image A over B**

Prediction geometry comes from downstream tasks

📚 *TaskMet: Task-Driven Metric Learning for Model Learning*. Bansal, Chen, Mukadam, Amos, NeurIPS 2023.



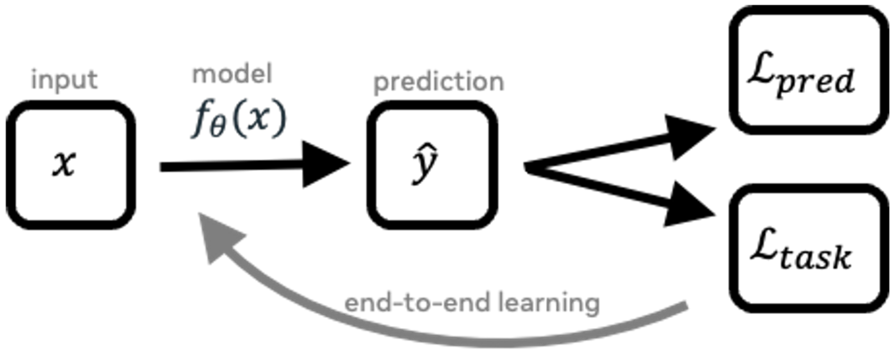
(MSE, likelihood) $\mathcal{L}_{pred}(\theta) \neq \mathcal{L}_{task}(\theta)$

How to use the task information?

📚 *TaskMet: Task-Driven Metric Learning for Model Learning*. Bansal, Chen, Mukadam, Amos, NeurIPS 2023.

Standard end-to-end task-based learning

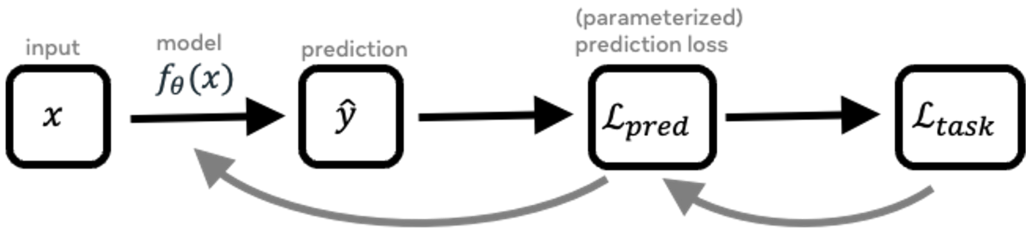
📚 *Task-based end-to-end model learning in stochastic optimization*. Donti, Amos, and Kolter, NeurIPS 2017.
📚 *Decision-Focused Learning for Combinatorial Optimization*. Wilder et al., AAAI 2019.
📚 *Smart "Predict, then optimize."* Elmachtoub and Grigas, Management Science 2022.



✔ Uses predictive and task information
❌ Prediction model may forget about the prediction task
❌ Prediction model may not generalize beyond the training task

Our approach: TaskMet

Task information only influences the prediction loss, not the model
Why? To retain the original prediction task



✔ Uses predictive and task information
✔ Prediction model is better at predicting
✔ Prediction model more likely to generalize to other tasks
✔ Competitive performance with other task-based learning methods

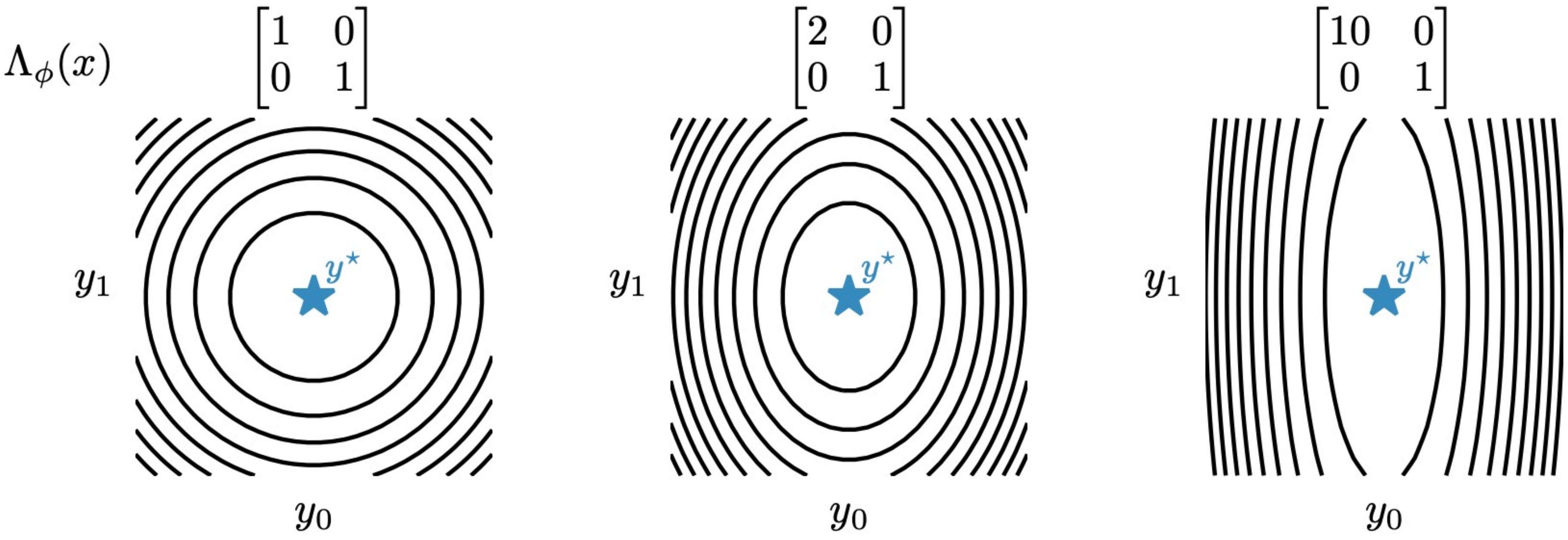
Parameterizing the geometry: Mahalanobis metrics

📚 *TaskMet: Task-Driven Metric Learning for Model Learning*. Bansal, Chen, Mukadam, Amos, NeurIPS 2023.

$$\mathcal{L} = \mathbb{E}_{(x,y^\star)\sim\mathcal{D}}\left[\|\hat{y}_\theta(x) - y^\star\|^2_{\Lambda_\phi(x)}\right]$$

$$\|x\|_M \triangleq \left(x^\top M x\right)^{1/2}$$

1. **relative importance of features**
   up/down-weighting based on importance
2. **relative importance of samples**
   via heteroscedastic metric $\Lambda(x)$



End-to-end learning the geometry

📚 *TaskMet: Task-Driven Metric Learning for Model Learning*. Bansal, Chen, Mukadam, Amos, NeurIPS 2023.

Formulate as a **bilevel** optimization problem

$$\begin{tikzcd}[column sep=huge]
\Lambda_\phi \arrow[r, bend left, "\arg\min_\theta \mathcal{L}_{\text{pred}}(\theta,\phi)"] & \theta^\star(\phi) \arrow[l, bend left, "\frac{\partial\theta^\star(\phi)}{\partial\phi} \text{ via IFT}"] \arrow[r, bend left, "\hat{y} = f_{\theta^\star(\phi)}(x)"] & \mathcal{L}_{\text{task}} \arrow[l, bend left, "\nabla_\theta \mathcal{L}_{\text{task}}"]
\end{tikzcd}$$

x: features
y: targets
Λ_ϕ: metric
$\mathcal{L}_{\text{pred}}(\theta,\phi) := \mathbb{E}\left[\|f_\theta(x) - y\|^2_{\Lambda_\phi(x)}\right]$

$$\boxed{\begin{aligned} \phi^\star &:= \operatorname*{argmin}_\phi \mathcal{L}_{\text{task}}(\theta^\star(\phi)) \\ \text{where } \theta^\star(\phi) &= \operatorname*{argmin}_\theta \mathcal{L}_{\text{pred}}(\theta,\phi) \end{aligned}}$$

optimize the metric with the task loss

solve the regression problem under that metric

$$\nabla_\phi \mathcal{L}_{\text{task}}(\theta^\star(\phi)) = \nabla_\theta \mathcal{L}_{\text{task}}(\theta)\Big|_{\theta=\theta^\star(\phi)} \cdot \underbrace{\frac{\partial\, \theta^\star(\phi)}{\partial\phi}}_{\left(\frac{\partial \mathcal{L}_{\text{pred}}(\theta,\phi)}{\partial^2\theta}\right)^{-1} \cdot \frac{\partial \mathcal{L}_{\text{pred}}(\theta,\phi)}{\partial\phi\,\partial\theta}\Big|_{\theta=\theta^\star(\phi)}}$$

calculate using the **implicit function theorem**

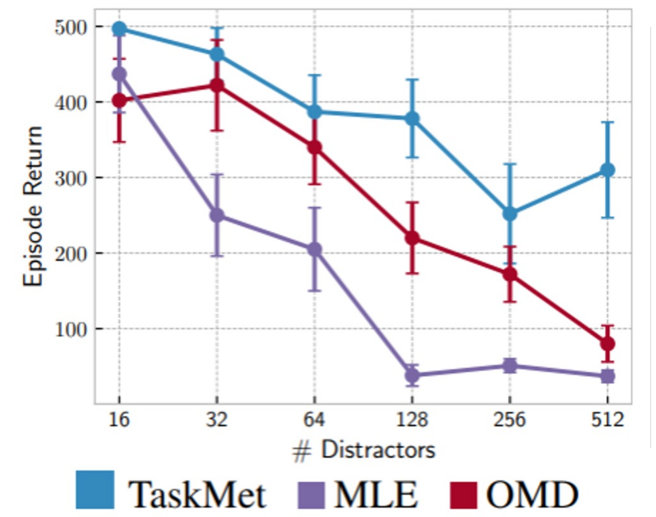
Experimental results

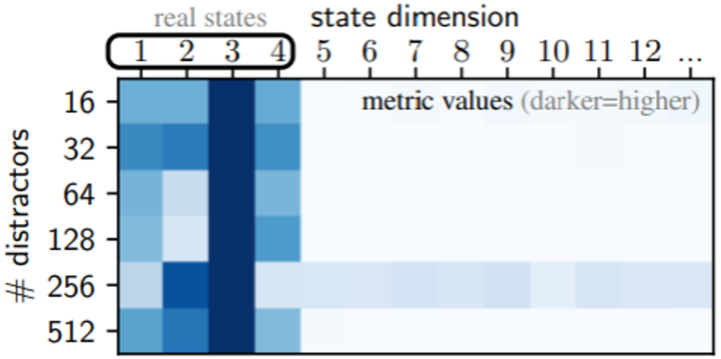
📚 *TaskMet: Task-Driven Metric Learning for Model Learning*. Bansal, Chen, Mukadam, Amos, NeurIPS 2023.

Learning MDP dynamics with distractors

📚 *Control-oriented model-based reinforcement learning with implicit differentiation.* Nikishin et al., AAAI 2022.

✔ state-of-the-art task performance
✔ lower prediction error





Standard DFL settings

📚 *Decision-focused learning without decision-making.* Shah et al., NeurIPS 2022.

✔ near-optimal task performance
✔ lower prediction error

		Problems		
Method	α	Cubic	Budget	Portfolio
MSE		-0.96 ± 0.02	0.54 ± 0.17	0.33 ± 0.03
DFL	0	0.61 ± 0.74	0.91 ± 0.06	0.25 ± 0.02
DFL	10	0.62 ± 0.74	0.81 ± 0.11	0.34 ± 0.03
LODL	0	0.96 ± 0.005	0.84 ± 0.105	0.17 ± 0.05
LODL	10	-0.95 ± 0.005	0.58 ± 0.14	0.30 ± 0.03
TaskMet		0.96 ± 0.005	0.83 ± 0.12	0.33 ± 0.03

End-to-end learning geometries for graphs, dynamical systems, and regression

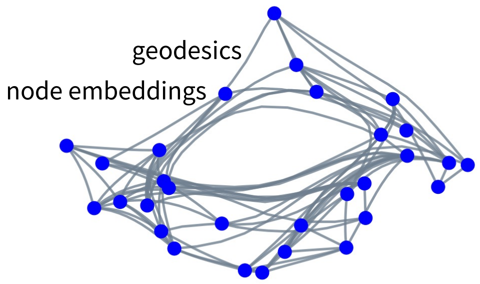
Brandon Amos • Meta AI, NYC

http://github.com/bamos/presentations

1. graph embeddings

Deep Riemannian Manifold Learning.
Lou, Nickel, Amos, NeurIPS 2020 Geo4dl workshop.

protein graph embedded in $\mathcal{M} = (\mathbb{R}^2, A_\theta)$



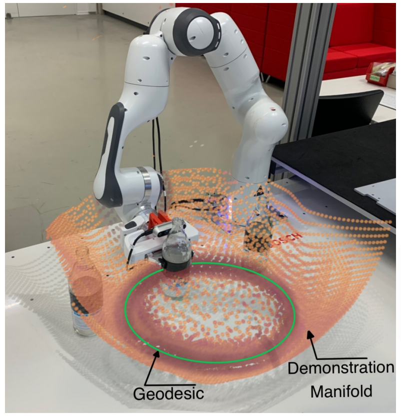
2. physical systems

Learning Riemannian Manifolds for Geodesic Motion Skills.
Beik-Mohammadi et al., RSS 2021.

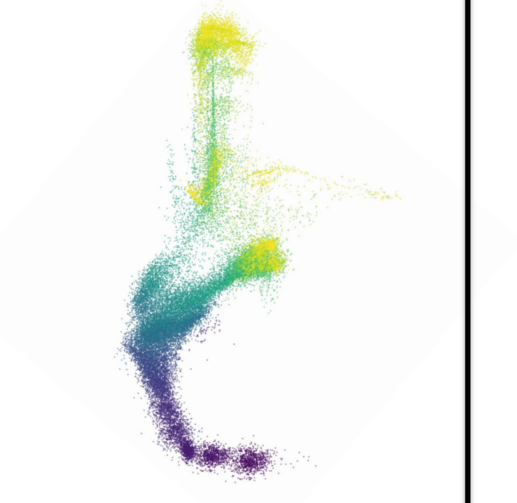
Riemannian Metric Learning via Optimal Transport.
Scarvelis and Solomon, ICLR 2023.

Neural Optimal Transport with Lagrangian Costs.
Pooladian, Domingo-Enrich, Chen, Amos, arXiv 2023.

robot demonstrations



cell populations over time



3. regression

TaskMet: Task-Driven Metric Learning for Model Learning.
Bansal, Chen, Mukadam, Amos, NeurIPS 2023.

linear regression