Short seminar on Optimal / Stochastic Control and Optimal Transport for ML

Optimal Control, Stochastic Control, and Optimal Transport

Dorian Gailhard

doriangailhard.github.io

July 23, 2026

Why Learn More Mathematics?

You all already know the mathematical foundations of ML:

Linear Algebra Probability Optimization

Why Learn More Mathematics?

You all already know the mathematical foundations of ML:

Linear Algebra Probability Optimization

But there are other powerful mathematical tools which are underused, and I think it would be worth it to learn them:

Why Learn More Mathematics?

You all already know the mathematical foundations of ML:

Linear Algebra Probability Optimization

But there are other powerful mathematical tools which are underused, and I think it would be worth it to learn them:

Why Learn More Mathematics?

You all already know the mathematical foundations of ML:

Linear Algebra Probability Optimization

But there are other powerful mathematical tools which are underused, and I think it would be worth it to learn them:

(Stochastic) Optimal Control, **Optimal Transport**, Differential Geometry, PDEs, Category Theory, Information Geometry...

Why These Topics for ML?

- Many ML models are now **dynamical**: diffusion models, flow models, neural ODEs, world models.

Why These Topics for ML?

- Many ML models are now **dynamical**: diffusion models, flow models, neural ODEs, world models.
- Many ML objectives compare **distributions**: generative modeling, domain adaptation, representation learning.

Why These Topics for ML?

- Many ML models are now **dynamical**: diffusion models, flow models, neural ODEs, world models.
- Many ML objectives compare **distributions**: generative modeling, domain adaptation, representation learning.
- Control gives a language for **steering** trajectories.

Why These Topics for ML?

- Many ML models are now **dynamical**: diffusion models, flow models, neural ODEs, world models.
- Many ML objectives compare **distributions**: generative modeling, domain adaptation, representation learning.
- Control gives a language for **steering** trajectories.
- Transport gives a language for **matching** distributions.

Why These Topics for ML?

- Many ML models are now **dynamical**: diffusion models, flow models, neural ODEs, world models.
- Many ML objectives compare **distributions**: generative modeling, domain adaptation, representation learning.
- Control gives a language for **steering** trajectories.
- Transport gives a language for **matching** distributions.

They also meet each other: dynamic optimal transport and Schrödinger bridges are essentially transport problems written as control problems.

Optimal Control

From Optimization to Optimal Control

Classical optimization:

$$\min_{x \in \mathbb{R}^d} f(x).$$

From Optimization to Optimal Control

Classical optimization:

$$\min_{x \in \mathbb{R}^d} f(x).$$

Optimal control:

$$\min_{u(\cdot)} \int_0^T L(x_t, u_t, t)\, dt + g(x_T)$$

subject to dynamics

$$\dot{x}_t = f(x_t, u_t, t), \qquad x_0 \text{ given.}$$

From Optimization to Optimal Control

Classical optimization:

$$\min_{x \in \mathbb{R}^d} f(x).$$

Optimal control:

$$\min_{u(\cdot)} \int_0^T L(x_t, u_t, t)\, dt + g(x_T)$$

subject to dynamics

$$\dot{x}_t = f(x_t, u_t, t), \qquad x_0 \text{ given.}$$

Optimization variable: not a vector, but a whole function of time u_t.

The Basic Objects

- $x_t \in \mathbb{R}^d$: **state**

The Basic Objects

- $x_t \in \mathbb{R}^d$: **state**
- $u_t \in U$: **control/action**

The Basic Objects

- $x_t \in \mathbb{R}^d$: **state**
- $u_t \in U$: **control/action**
- $\dot{x}_t = f(x_t, u_t, t)$: **dynamics**

The Basic Objects

- $x_t \in \mathbb{R}^d$: **state**
- $u_t \in U$: **control/action**
- $\dot{x}_t = f(x_t, u_t, t)$: **dynamics**
- $L(x_t, u_t, t)$: **running cost**

The Basic Objects

- $x_t \in \mathbb{R}^d$: **state**
- $u_t \in U$: **control/action**
- $\dot{x}_t = f(x_t, u_t, t)$: **dynamics**
- $L(x_t, u_t, t)$: **running cost**
- $g(x_T)$: **terminal cost**

The Basic Objects

- $x_t \in \mathbb{R}^d$: **state**
- $u_t \in U$: **control/action**
- $\dot{x}_t = f(x_t, u_t, t)$: **dynamics**
- $L(x_t, u_t, t)$: **running cost**
- $g(x_T)$: **terminal cost**

Mental picture: every control signal draws a trajectory; the goal is to find the cheapest trajectory.

Example: Minimum-Energy Steering

A very simple control problem:

$$\dot{x}_t = u_t, \qquad x_0 = a, \qquad x_T = b.$$

Example: Minimum-Energy Steering

A very simple control problem:

$$\dot{x}_t = u_t, \qquad x_0 = a, \qquad x_T = b.$$

Minimize kinetic energy:

$$\min_{u(\cdot)} \int_0^T \frac{1}{2} \|u_t\|^2 dt.$$

Example: Minimum-Energy Steering

A very simple control problem:

$$\dot{x}_t = u_t, \qquad x_0 = a, \qquad x_T = b.$$

Minimize kinetic energy:

$$\min_{u(\cdot)} \int_0^T \frac{1}{2} \|u_t\|^2 dt.$$

Since $x_T - x_0 = \int_0^T u_t dt$, the optimal solution is constant velocity:

$$u_t^\star = \frac{b-a}{T}.$$

Example: Minimum-Energy Steering

A very simple control problem:

$$\dot{x}_t = u_t, \qquad x_0 = a, \qquad x_T = b.$$

Minimize kinetic energy:

$$\min_{u(\cdot)} \int_0^T \frac{1}{2} \|u_t\|^2 dt.$$

Since $x_T - x_0 = \int_0^T u_t dt$, the optimal solution is constant velocity:

$$u_t^\star = \frac{b-a}{T}.$$

Intuition: among all trajectories joining a to b in time T, the straight line travelled at constant speed has minimum kinetic energy.

Principle 1: Dynamic Programming

Richard Bellman, *Dynamic Programming*, 1957

Define the value function:

$$V(t,x) = \min_{u(\cdot)} \left[\int_t^T L(x_s, u_s, s)\, ds + g(x_T) \right].$$

Principle 1: Dynamic Programming

Richard Bellman, *Dynamic Programming*, 1957

Define the value function:

$$V(t,x) = \min_{u(\cdot)} \left[\int_t^T L(x_s, u_s, s)\, ds + g(x_T) \right].$$

Bellman's principle: an optimal trajectory remains optimal after any intermediate time.

Principle 1: Dynamic Programming

Richard Bellman, *Dynamic Programming*, 1957

Define the value function:

$$V(t,x) = \min_{u(\cdot)} \left[\int_t^T L(x_s, u_s, s)\, ds + g(x_T) \right].$$

Bellman's principle: an optimal trajectory remains optimal after any intermediate time.

In informal terms:

optimal from now = best first action + optimal from next state.

The Hamilton-Jacobi-Bellman Equation

Richard Bellman, 1957; Fleming and Soner, *Controlled Markov Processes and Viscosity Solutions*, 2006

Assuming the value function is differentiable, it satisfies

$$-\partial_t V(t,x) = \min_{u \in U} \left\{ L(x,u,t) + \nabla_x V(t,x)^\top f(x,u,t) \right\},$$

with terminal condition

$$V(T,x) = g(x).$$

The Hamilton-Jacobi-Bellman Equation

Richard Bellman, 1957; Fleming and Soner, *Controlled Markov Processes and Viscosity Solutions*, 2006

Assuming the value function is differentiable, it satisfies

$$-\partial_t V(t,x) = \min_{u \in U} \left\{ L(x,u,t) + \nabla_x V(t,x)^\top f(x,u,t) \right\},$$

with terminal condition

$$V(T,x) = g(x).$$

Interpretation: ∇V measures how sensitive the optimal future cost is to perturbations of the current state.

The Hamilton-Jacobi-Bellman Equation

Richard Bellman, 1957; Fleming and Soner, *Controlled Markov Processes and Viscosity Solutions*, 2006

Assuming the value function is differentiable, it satisfies

$$-\partial_t V(t,x) = \min_{u \in U} \left\{ L(x,u,t) + \nabla_x V(t,x)^\top f(x,u,t) \right\},$$

with terminal condition

$$V(T,x) = g(x).$$

Interpretation: ∇V measures how sensitive the optimal future cost is to perturbations of the current state.

In general, V need not be differentiable; the HJB equation is interpreted using viscosity solutions.

Principle 2: Pontryagin Maximum Principle

Pontryagin, Boltyanskii, Gamkrelidze, Mishchenko, *The Mathematical Theory of Optimal Processes*, 1962

Define the Hamiltonian:

$$H(x, u, p, t) = L(x, u, t) + p^\top f(x, u, t).$$

Principle 2: Pontryagin Maximum Principle

Pontryagin, Boltyanskii, Gamkrelidze, Mishchenko, *The Mathematical Theory of Optimal Processes*, 1962

Define the Hamiltonian:

$$H(x, u, p, t) = L(x, u, t) + p^\top f(x, u, t).$$

At an optimum, there exists an adjoint variable (or costate) p_t such that

$$\dot{x}_t = \partial_p H(x_t, u_t, p_t, t),$$

$$\dot{p}_t = -\partial_x H(x_t, u_t, p_t, t),$$

$$u_t^\star \in \arg\min_u H(x_t, u, p_t, t).$$

Principle 2: Pontryagin Maximum Principle

Pontryagin, Boltyanskii, Gamkrelidze, Mishchenko, *The Mathematical Theory of Optimal Processes*, 1962

Define the Hamiltonian:

$$H(x, u, p, t) = L(x, u, t) + p^\top f(x, u, t).$$

At an optimum, there exists an adjoint variable (or costate) p_t such that

$$\dot{x}_t = \partial_p H(x_t, u_t, p_t, t),$$

$$\dot{p}_t = -\partial_x H(x_t, u_t, p_t, t),$$

$$u_t^\star \in \arg\min_u H(x_t, u, p_t, t).$$

Mental picture: optimize a trajectory by coupling a forward state equation and a backward adjoint equation.

HJB vs Pontryagin

Dynamic programming / HJB	**Pontryagin / adjoint**
Global value function $V(t, x)$	One optimal trajectory
PDE in state space	ODE boundary-value problem
Suffers from curse of dimensionality	Often more computationally practical
Policy from arg min in HJB	Control from Hamiltonian minimization

HJB vs Pontryagin

Dynamic programming / HJB	**Pontryagin / adjoint**
Global value function $V(t, x)$	One optimal trajectory
PDE in state space	ODE boundary-value problem
Suffers from curse of dimensionality	Often more computationally practical
Policy from arg min in HJB	Control from Hamiltonian minimization

In ML, the adjoint viewpoint appears naturally in neural ODEs and differentiable simulation.

Optimal Control in ML

Optimal Control and Neural ODEs

Chen, Rubanova, Bettencourt, Duvenaud, *Neural Ordinary Differential Equations*, NeurIPS 2018

A residual network looks like an Euler discretization:

$$x_{k+1} = x_k + hf_\theta(x_k, k).$$

Optimal Control and Neural ODEs

Chen, Rubanova, Bettencourt, Duvenaud, *Neural Ordinary Differential Equations*, NeurIPS 2018

A residual network looks like an Euler discretization:

$$x_{k+1} = x_k + hf_\theta(x_k, k).$$

A neural ODE takes the continuous-depth limit:

$$\dot{x}_t = f_\theta(x_t, t).$$

Optimal Control and Neural ODEs

Chen, Rubanova, Bettencourt, Duvenaud, *Neural Ordinary Differential Equations*, NeurIPS 2018

A residual network looks like an Euler discretization:

$$x_{k+1} = x_k + hf_\theta(x_k, k).$$

A neural ODE takes the continuous-depth limit:

$$\dot{x}_t = f_\theta(x_t, t).$$

Training resembles an optimal control problem:

$$\min_\theta \ \ell(x_T, y) + \int_0^T R(\theta_t)dt.$$

Optimal Control and Neural ODEs

Chen, Rubanova, Bettencourt, Duvenaud, *Neural Ordinary Differential Equations*, NeurIPS 2018

A residual network looks like an Euler discretization:

$$x_{k+1} = x_k + hf_\theta(x_k, k).$$

A neural ODE takes the continuous-depth limit:

$$\dot{x}_t = f_\theta(x_t, t).$$

Training resembles an optimal control problem:

$$\min_\theta \ \ell(x_T, y) + \int_0^T R(\theta_t)dt.$$

Idea: the network learns a vector field that transports inputs into useful representations.

Control View of Deep Learning

Weinan E, *A Proposal on Machine Learning via Dynamical Systems*, 2017; Haber and Ruthotto, 2017

- Layers become time steps.

Control View of Deep Learning

Weinan E, *A Proposal on Machine Learning via Dynamical Systems*, 2017; Haber and Ruthotto, 2017

- Layers become time steps.
- Hidden states become controlled dynamics.

Control View of Deep Learning

Weinan E, *A Proposal on Machine Learning via Dynamical Systems*, 2017; Haber and Ruthotto, 2017

- Layers become time steps.
- Hidden states become controlled dynamics.
- Parameters become controls.

Control View of Deep Learning

Weinan E, *A Proposal on Machine Learning via Dynamical Systems*, 2017; Haber and Ruthotto, 2017

- Layers become time steps.
- Hidden states become controlled dynamics.
- Parameters become controls.
- Backpropagation becomes an adjoint equation.

Control View of Deep Learning

Weinan E, *A Proposal on Machine Learning via Dynamical Systems*, 2017; Haber and Ruthotto, 2017

- Layers become time steps.
- Hidden states become controlled dynamics.
- Parameters become controls.
- Backpropagation becomes an adjoint equation.

This does not mean all deep learning should be solved with classical control, but it gives useful vocabulary for stability, discretization, continuous depth, and constrained dynamics.

Control View of Deep Learning

Weinan E, *A Proposal on Machine Learning via Dynamical Systems*, 2017; Haber and Ruthotto, 2017

- Layers become time steps.
- Hidden states become controlled dynamics.
- Parameters become controls.
- Backpropagation becomes an adjoint equation.

This does not mean all deep learning should be solved with classical control, but it gives useful vocabulary for stability, discretization, continuous depth, and constrained dynamics.

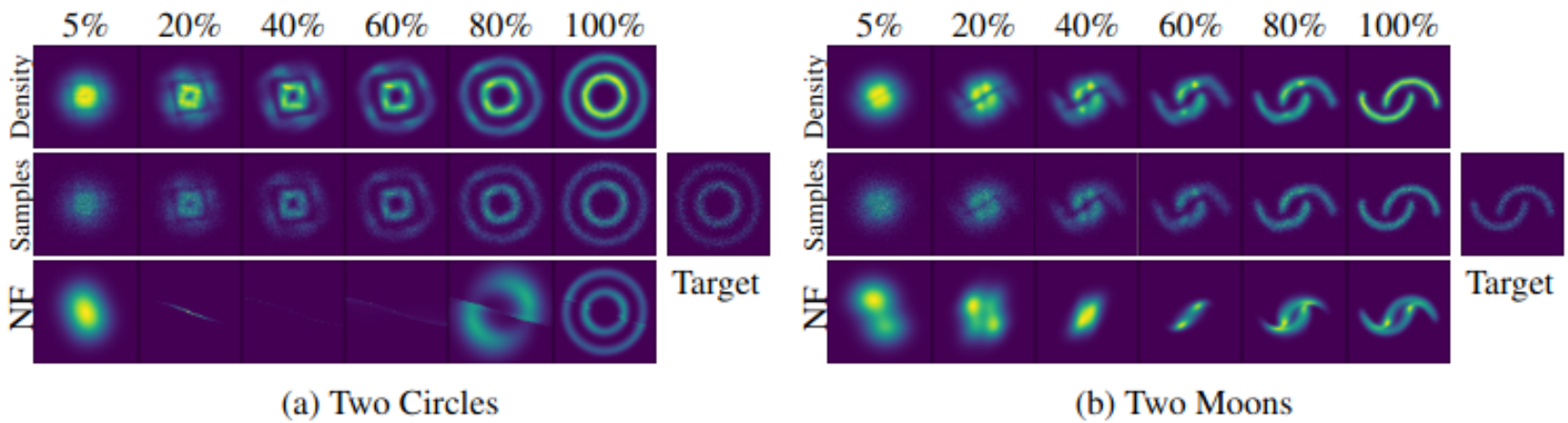


Figure: *Image taken from* Neural Ordinary Differential Equations.

Optimal Control for Image Morphing

Trouvé, Younes, Beg, Miller, LDDMM (1998–2005)

Instead of controlling a point, control an entire deformation of space.

Let $\phi_t : \Omega \to \Omega$ satisfy

$$\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)),$$

where v_t is a smooth velocity field.

Optimal Control for Image Morphing

Trouvé, Younes, Beg, Miller, LDDMM (1998–2005)

Instead of controlling a point, control an entire deformation of space.

Let $\phi_t : \Omega \to \Omega$ satisfy

$$\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)),$$

where v_t is a smooth velocity field.

The image evolves by transporting its pixels:

$$I_t(x) = I_0(\phi_t^{-1}(x)).$$

Optimal Control for Image Morphing

Trouvé, Younes, Beg, Miller, LDDMM (1998–2005)

Instead of controlling a point, control an entire deformation of space.

Let $\phi_t : \Omega \to \Omega$ satisfy

$$\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)),$$

where v_t is a smooth velocity field.

The image evolves by transporting its pixels:

$$I_t(x) = I_0(\phi_t^{-1}(x)).$$

Find the smoothest deformation matching the target image:

$$\min_{v_t} \int_0^1 \|v_t\|_V^2 dt + \lambda \|I_1 - I_{\text{target}}\|^2.$$

Optimal Control for Image Morphing

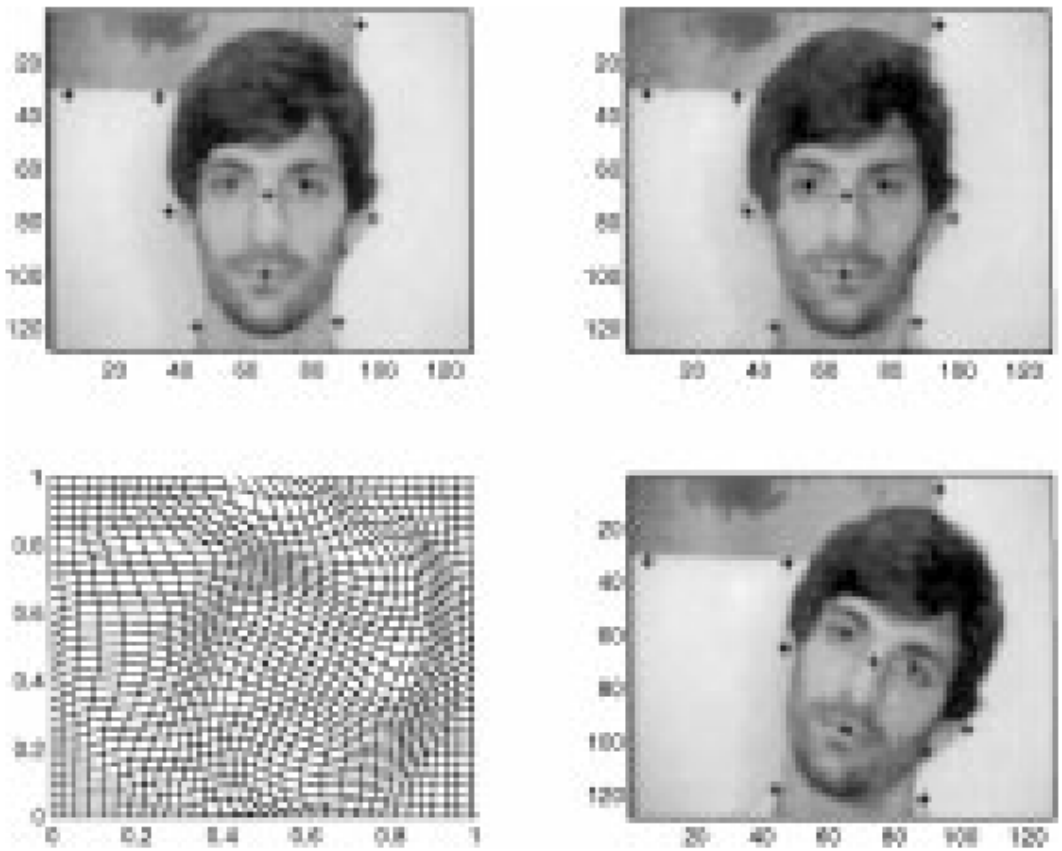
Trouvé, Younes, Beg, Miller, LDDMM (1998–2005)

Idea: learn a continuous flow of diffeomorphisms instead of matching pixels directly.

Optimal Control for Image Morphing

Trouvé, Younes, Beg, Miller, LDDMM (1998–2005)

Idea: learn a continuous flow of diffeomorphisms instead of matching pixels directly.



Stochastic Optimal Control

Adding Noise

Many systems are uncertain:

$$dX_t = f(X_t, u_t, t)dt + \sigma(X_t, t)dW_t.$$

Adding Noise

Many systems are uncertain:

$$dX_t = f(X_t, u_t, t)dt + \sigma(X_t, t)dW_t.$$

Here W_t is Brownian motion, and σdW_t models stochastic perturbations.

Adding Noise

Many systems are uncertain:

$$dX_t = f(X_t, u_t, t)dt + \sigma(X_t, t)dW_t.$$

Here W_t is Brownian motion, and σdW_t models stochastic perturbations.

The objective becomes expected cost:

$$\min_{u(\cdot)} \mathbb{E}\left[\int_0^T L(X_t, u_t, t)dt + g(X_T)\right].$$

Adding Noise

Many systems are uncertain:

$$dX_t = f(X_t, u_t, t)dt + \sigma(X_t, t)dW_t.$$

Here W_t is Brownian motion, and σdW_t models stochastic perturbations.

The objective becomes expected cost:

$$\min_{u(\cdot)} \mathbb{E}\left[\int_0^T L(X_t, u_t, t)dt + g(X_T)\right].$$

Key difficulty: controls affect a distribution over trajectories, not only one path.

The Stochastic HJB Equation

Fleming and Soner, *Controlled Markov Processes and Viscosity Solutions*, 2006

For

$$dX_t = f(X_t, u_t, t)dt + \sigma(X_t, t)dW_t,$$

we formally get

$$-\partial_t V(t,x) = \min_u \left\{ L(x,u,t) + \nabla V^\top f(x,u,t) + \frac{1}{2}\,\mathrm{Tr}\left(\sigma\sigma^\top \nabla^2 V\right) \right\}.$$

The Stochastic HJB Equation

Fleming and Soner, *Controlled Markov Processes and Viscosity Solutions*, 2006

For

$$dX_t = f(X_t, u_t, t)dt + \sigma(X_t, t)dW_t,$$

we formally get

$$-\partial_t V(t,x) = \min_u \Big\{ L(x,u,t) + \nabla V^\top f(x,u,t) + \frac{1}{2}\,\mathrm{Tr}\left(\sigma\sigma^\top \nabla^2 V\right) \Big\}.$$

The new second-order term appears because uncertainty spreads probability mass.

Entropy-Regularized Control

Consider the stochastic control problem

$$\min_u \ \mathbb{E}\left[\int_0^T \left(c(X_t, t) + \frac{1}{2}\|u_t\|^2\right) dt + g(X_T)\right]$$

subject to

$$dX_t = \big(b(X_t, t) + \sigma u_t\big) dt + \sigma \, dW_t.$$

Entropy-Regularized Control

Consider the stochastic control problem

$$\min_u \ \mathbb{E}\left[\int_0^T \left(c(X_t, t) + \frac{1}{2}\|u_t\|^2\right) dt + g(X_T)\right]$$

subject to

$$dX_t = \big(b(X_t, t) + \sigma u_t\big)dt + \sigma\, dW_t.$$

Let $\mathbb{P}$ denote the uncontrolled diffusion and $\mathbb{Q}_u$ the controlled one. Girsanov's theorem gives

$$\mathrm{KL}(\mathbb{Q}_u \,\|\, \mathbb{P}) = \frac{1}{2}\mathbb{E}_{\mathbb{Q}_u}\left[\int_0^T \|u_t\|^2 dt\right].$$

Entropy-Regularized Control

Consider the stochastic control problem

$$\min_u \ \mathbb{E}\left[\int_0^T \left(c(X_t, t) + \frac{1}{2}\|u_t\|^2\right) dt + g(X_T)\right]$$

subject to

$$dX_t = \big(b(X_t, t) + \sigma u_t\big)dt + \sigma\, dW_t.$$

Let $\mathbb{P}$ denote the uncontrolled diffusion and $\mathbb{Q}_u$ the controlled one. Girsanov's theorem gives

$$\mathrm{KL}(\mathbb{Q}_u \,\|\, \mathbb{P}) = \frac{1}{2}\mathbb{E}_{\mathbb{Q}_u}\left[\int_0^T \|u_t\|^2 dt\right].$$

Interpretation: the controller minimizes the task cost while staying as close as possible (in KL divergence) to the natural diffusion.

Stochastic Control in ML

Control as Inference

Todorov, 2009; Levine, 2018

Instead of optimizing controls directly, optimize a trajectory distribution:

$$\max_q \ \mathbb{E}_q[R(\tau)] - \mathrm{KL}(q \| p_0),$$

where p_0 is the uncontrolled dynamics and $R(\tau) = \sum_t r(s_t, a_t)$ is the reward obtained during the trajectory.

Control as Inference

Todorov, 2009; Levine, 2018

Instead of optimizing controls directly, optimize a trajectory distribution:

$$\max_q \ \mathbb{E}_q[R(\tau)] - \mathrm{KL}(q \| p_0),$$

where p_0 is the uncontrolled dynamics and $R(\tau) = \sum_t r(s_t, a_t)$ is the reward obtained during the trajectory.

The optimal distribution is

$$q^\star(\tau) \propto p_0(\tau) e^{R(\tau)},$$

Control as Inference

Todorov, 2009; Levine, 2018

Instead of optimizing controls directly, optimize a trajectory distribution:

$$\max_q \ \mathbb{E}_q[R(\tau)] - \mathrm{KL}(q \| p_0),$$

where p_0 is the uncontrolled dynamics and $R(\tau) = \sum_t r(s_t, a_t)$ is the reward obtained during the trajectory.

The optimal distribution is

$$q^\star(\tau) \propto p_0(\tau) e^{R(\tau)},$$

Interpretation: high-reward trajectories become exponentially more likely while remaining close to the natural dynamics.

Control as Inference

Todorov, 2009; Levine, 2018

Instead of optimizing controls directly, optimize a trajectory distribution:

$$\max_q \ \mathbb{E}_q[R(\tau)] - \mathrm{KL}(q \| p_0),$$

where p_0 is the uncontrolled dynamics and $R(\tau) = \sum_t r(s_t, a_t)$ is the reward obtained during the trajectory.

The optimal distribution is

$$q^\star(\tau) \propto p_0(\tau) e^{R(\tau)},$$

Interpretation: high-reward trajectories become exponentially more likely while remaining close to the natural dynamics.

Connection: this is the same entropy-regularized control problem as the previous slide, viewed as inference over trajectory distributions.

Diffusion Models as Stochastic Dynamics

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole, *Score-Based Generative Modeling through SDEs*, ICLR 2021

Diffusion models define a noising process:

$$dX_t = b(X_t, t)dt + \sigma(t)dW_t,$$

then learn to reverse it.

Diffusion Models as Stochastic Dynamics

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole, *Score-Based Generative Modeling through SDEs*, ICLR 2021

Diffusion models define a noising process:

$$dX_t = b(X_t, t)dt + \sigma(t)dW_t,$$

then learn to reverse it.

Reverse sampling also has an SDE form:

$$dX_t = \tilde{b}_\theta(X_t, t)dt + \tilde{\sigma}(t)d\bar{W}_t.$$

Diffusion Models as Stochastic Dynamics

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole, *Score-Based Generative Modeling through SDEs*, ICLR 2021

Diffusion models define a noising process:

$$dX_t = b(X_t, t)dt + \sigma(t)dW_t,$$

then learn to reverse it.

Reverse sampling also has an SDE form:

$$dX_t = \tilde{b}_\theta(X_t, t)dt + \tilde{\sigma}(t)d\bar{W}_t.$$

So generation is already a stochastic trajectory.

Diffusion Models as Stochastic Dynamics

Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole, *Score-Based Generative Modeling through SDEs*, ICLR 2021

Diffusion models define a noising process:

$$dX_t = b(X_t, t)dt + \sigma(t)dW_t,$$

then learn to reverse it.

Reverse sampling also has an SDE form:

$$dX_t = \tilde{b}_\theta(X_t, t)dt + \tilde{\sigma}(t)d\bar{W}_t.$$

So generation is already a stochastic trajectory.

Natural question: can we control this trajectory to satisfy extra properties?

Guiding Diffusion Models with Stochastic Control

Uehara et al., 2024; Tang, 2024; Azangulov et al., *Adaptive Diffusion Guidance via Stochastic Optimal Control*, 2025

Suppose a pretrained diffusion model samples from $p_0(x)$.

Guiding Diffusion Models with Stochastic Control

Uehara et al., 2024; Tang, 2024; Azangulov et al., *Adaptive Diffusion Guidance via Stochastic Optimal Control*, 2025

Suppose a pretrained diffusion model samples from $p_0(x)$.

We want samples with high property/reward:

$$R(x) \quad \text{large.}$$

Guiding Diffusion Models with Stochastic Control

Uehara et al., 2024; Tang, 2024; Azangulov et al., *Adaptive Diffusion Guidance via Stochastic Optimal Control*, 2025

Suppose a pretrained diffusion model samples from $p_0(x)$.

We want samples with high property/reward:

$$R(x) \quad \text{large.}$$

A control view writes:

$$\max_u \mathbb{E}\left[R(X_0) - \lambda \int_0^T \|u_t\|^2 dt\right],$$

where u_t modifies the reverse diffusion drift.

Guiding Diffusion Models with Stochastic Control

Uehara et al., 2024; Tang, 2024; Azangulov et al., *Adaptive Diffusion Guidance via Stochastic Optimal Control*, 2025

Suppose a pretrained diffusion model samples from $p_0(x)$.

We want samples with high property/reward:

$$R(x) \quad \text{large.}$$

A control view writes:

$$\max_u \mathbb{E}\left[R(X_0) - \lambda \int_0^T \|u_t\|^2 dt\right],$$

where u_t modifies the reverse diffusion drift.

Intuition: do not retrain the whole model; add just enough control during sampling to move toward desirable regions.

Examples of Controllable Generation

Examples:

- Molecules: steer toward high binding affinity, low toxicity, drug-likeness.

Examples of Controllable Generation

Examples:

- Molecules: steer toward high binding affinity, low toxicity, drug-likeness.
- Images: steer toward aesthetic score, object attributes, classifier labels.

Examples of Controllable Generation

Examples:

- Molecules: steer toward high binding affinity, low toxicity, drug-likeness.
- Images: steer toward aesthetic score, object attributes, classifier labels.
- Materials: steer toward target stability or physical properties.

Examples of Controllable Generation

Examples:

- Molecules: steer toward high binding affinity, low toxicity, drug-likeness.
- Images: steer toward aesthetic score, object attributes, classifier labels.
- Materials: steer toward target stability or physical properties.
- Graphs/hypergraphs: steer toward validity, motifs, spectrum, size, or task score.

Examples of Controllable Generation

Examples:

- Molecules: steer toward high binding affinity, low toxicity, drug-likeness.
- Images: steer toward aesthetic score, object attributes, classifier labels.
- Materials: steer toward target stability or physical properties.
- Graphs/hypergraphs: steer toward validity, motifs, spectrum, size, or task score.

Important warning: reward optimization can produce adversarial samples if the reward is imperfect.

Schrödinger Bridges

Schrödinger, 1931; Léonard, 2014; De Bortoli et al., *Diffusion Schrödinger Bridge*, NeurIPS 2021

Given two distributions μ_0 and μ_T, find the most likely stochastic process connecting them, relative to a reference diffusion.

Schrödinger Bridges

Schrödinger, 1931; Léonard, 2014; De Bortoli et al., *Diffusion Schrödinger Bridge*, NeurIPS 2021

Given two distributions μ_0 and μ_T, find the most likely stochastic process connecting them, relative to a reference diffusion.

One formulation:

$$\min_{\mathbb{P}} \mathrm{KL}(\mathbb{P} \,\|\, \mathbb{P}_0) \quad \text{s.t.} \quad X_0 \sim \mu_0,\ X_T \sim \mu_T.$$

Schrödinger Bridges

Schrödinger, 1931; Léonard, 2014; De Bortoli et al., *Diffusion Schrödinger Bridge*, NeurIPS 2021

Given two distributions μ_0 and μ_T, find the most likely stochastic process connecting them, relative to a reference diffusion.

One formulation:

$$\min_{\mathbb{P}} \mathrm{KL}(\mathbb{P} \,\|\, \mathbb{P}_0) \quad \text{s.t.} \quad X_0 \sim \mu_0,\ X_T \sim \mu_T.$$

Interpretation: stochastic optimal transport with an entropy/KL regularization.

Schrödinger Bridges

Schrödinger, 1931; Léonard, 2014; De Bortoli et al., *Diffusion Schrödinger Bridge*, NeurIPS 2021

Given two distributions μ_0 and μ_T, find the most likely stochastic process connecting them, relative to a reference diffusion.

One formulation:

$$\min_{\mathbb{P}} \mathrm{KL}(\mathbb{P} \,\|\, \mathbb{P}_0) \quad \text{s.t.} \quad X_0 \sim \mu_0,\ X_T \sim \mu_T.$$

Interpretation: stochastic optimal transport with an entropy/KL regularization.

ML connection: generative modeling by learning a controlled diffusion bridge from noise to data.

Optimal Transport

What is Optimal Transport?

Monge, 1781; Kantorovich, 1942; Villani, *Optimal Transport*, 2008; Peyré and Cuturi, *Computational Optimal Transport*, 2019

Given two probability distributions μ and ν, optimal transport asks:

What is Optimal Transport?

Monge, 1781; Kantorovich, 1942; Villani, *Optimal Transport*, 2008; Peyré and Cuturi, *Computational Optimal Transport*, 2019

Given two probability distributions μ and ν, optimal transport asks:

What is the cheapest way to move the mass of μ into ν?

What is Optimal Transport?

Monge, 1781; Kantorovich, 1942; Villani, *Optimal Transport*, 2008; Peyré and Cuturi, *Computational Optimal Transport*, 2019

Given two probability distributions μ and ν, optimal transport asks:

What is the cheapest way to move the mass of μ into ν?

With cost $c(x, y)$, the Kantorovich problem is

$$\min_{\pi \in \Pi(\mu,\nu)} \int c(x, y)\, d\pi(x, y),$$

where $\Pi(\mu, \nu)$ is the set of couplings with marginals μ and ν.

The Coupling

A coupling $\pi(x, y)$ tells how much mass from x is sent to y.

The Coupling

A coupling $\pi(x, y)$ tells how much mass from x is sent to y.

Marginal constraints:

$$\int \pi(x, y)dy = \mu(x), \qquad \int \pi(x, y)dx = \nu(y).$$

The Coupling

A coupling $\pi(x, y)$ tells how much mass from x is sent to y.

Marginal constraints:

$$\int \pi(x, y)dy = \mu(x), \qquad \int \pi(x, y)dx = \nu(y).$$

In discrete form:

$$\min_{P \geq 0} \langle C, P \rangle \quad \text{s.t.} \quad P\mathbf{1} = a,\ P^\top \mathbf{1} = b.$$

The Coupling

A coupling $\pi(x, y)$ tells how much mass from x is sent to y.

Marginal constraints:

$$\int \pi(x, y)dy = \mu(x), \qquad \int \pi(x, y)dx = \nu(y).$$

In discrete form:

$$\min_{P \geq 0} \langle C, P \rangle \quad \text{s.t.} \quad P\mathbf{1} = a, \; P^\top \mathbf{1} = b.$$

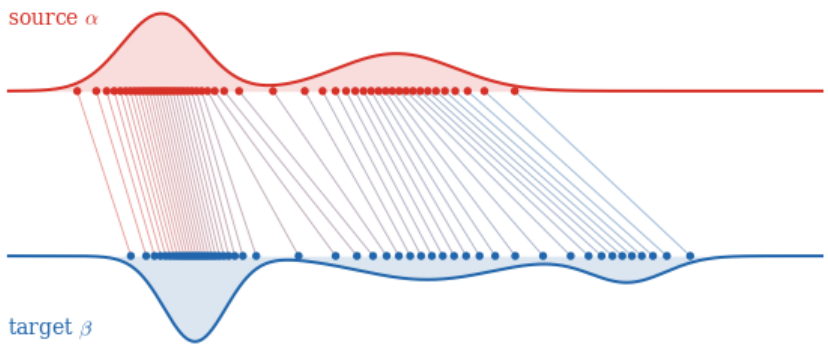


Figure: Example of an OT plan between two distributions. (*Image taken from Gabriel Peyré's website.)*

Wasserstein Distance

For $c(x, y) = \|x - y\|^p$, define

$$W_p(\mu, \nu) = \left(\min_{\pi \in \Pi(\mu,\nu)} \int \|x - y\|^p d\pi(x, y) \right)^{1/p}.$$

Wasserstein Distance

For $c(x, y) = \|x - y\|^p$, define

$$W_p(\mu, \nu) = \left(\min_{\pi \in \Pi(\mu,\nu)} \int \|x - y\|^p d\pi(x, y) \right)^{1/p}.$$

Unlike KL divergence, Wasserstein distance uses the geometry of the space.

Wasserstein Distance

For $c(x, y) = \|x - y\|^p$, define

$$W_p(\mu, \nu) = \left(\min_{\pi \in \Pi(\mu,\nu)} \int \|x - y\|^p d\pi(x, y) \right)^{1/p}.$$

Unlike KL divergence, Wasserstein distance uses the geometry of the space.

Example intuition: Two non-overlapping distributions can have infinite/undefined KL in one direction, but finite Wasserstein distance.

Kantorovich Duality

Villani, 2008; Peyré and Cuturi, 2019

The primal problem searches for an optimal transport plan.

Kantorovich Duality

Villani, 2008; Peyré and Cuturi, 2019

The primal problem searches for an optimal transport plan.

The dual instead assigns a *value* to mass at each location:

$$\sup_{\varphi,\psi} \int \varphi(x)\, d\mu(x) + \int \psi(y)\, d\nu(y)$$

subject to

$$\varphi(x) + \psi(y) \leq c(x, y),$$

where $c(x, y)$ is the transport cost.

Kantorovich Duality

Villani, 2008; Peyré and Cuturi, 2019

The primal problem searches for an optimal transport plan.

The dual instead assigns a *value* to mass at each location:

$$\sup_{\varphi,\psi} \int \varphi(x)\, d\mu(x) + \int \psi(y)\, d\nu(y)$$

subject to

$$\varphi(x) + \psi(y) \leq c(x, y),$$

where $c(x, y)$ is the transport cost.

For the Wasserstein-1 distance, one can shows

$$W_1(\mu, \nu) = \sup_{\|f\|_{\mathrm{Lip}} \leq 1} \left(\mathbb{E}_\mu[f] - \mathbb{E}_\nu[f]\right).$$

Entropic Regularization and Sinkhorn

Cuturi, *Sinkhorn Distances: Lightspeed Computation of Optimal Transport*, NeurIPS 2013

Computing exact OT can be expensive. Add an entropy penalty:

$$\min_{P \in U(a,b)} \langle C, P \rangle + \varepsilon \sum_{ij} P_{ij}(\log P_{ij} - 1).$$

Entropic Regularization and Sinkhorn

Cuturi, *Sinkhorn Distances: Lightspeed Computation of Optimal Transport*, NeurIPS 2013

Computing exact OT can be expensive. Add an entropy penalty:

$$\min_{P \in U(a,b)} \langle C, P \rangle + \varepsilon \sum_{ij} P_{ij}(\log P_{ij} - 1).$$

The optimal transport plan has a simple form:

$$P^\star = \mathrm{diag}(u)\, K\, \mathrm{diag}(v), \qquad K_{ij} = e^{-C_{ij}/\varepsilon}.$$

Only the scaling vectors u, v remain unknown.

Entropic Regularization and Sinkhorn

Cuturi, *Sinkhorn Distances: Lightspeed Computation of Optimal Transport*, NeurIPS 2013

Computing exact OT can be expensive. Add an entropy penalty:

$$\min_{P \in U(a,b)} \langle C, P \rangle + \varepsilon \sum_{ij} P_{ij}(\log P_{ij} - 1).$$

The optimal transport plan has a simple form:

$$P^\star = \mathrm{diag}(u)\, K\, \mathrm{diag}(v), \qquad K_{ij} = e^{-C_{ij}/\varepsilon}.$$

Only the scaling vectors u, v remain unknown.

Sinkhorn alternates

$$u \leftarrow \frac{a}{Kv}, \qquad v \leftarrow \frac{b}{K^\top u},$$

until the row and column sums equal the desired marginals.

Entropic Regularization and Sinkhorn

Cuturi, *Sinkhorn Distances: Lightspeed Computation of Optimal Transport*, NeurIPS 2013

Computing exact OT can be expensive. Add an entropy penalty:

$$\min_{P \in U(a,b)} \langle C, P \rangle + \varepsilon \sum_{ij} P_{ij}(\log P_{ij} - 1).$$

The optimal transport plan has a simple form:

$$P^\star = \mathrm{diag}(u)\, K\, \mathrm{diag}(v), \qquad K_{ij} = e^{-C_{ij}/\varepsilon}.$$

Only the scaling vectors u, v remain unknown.

Sinkhorn alternates

$$u \leftarrow \frac{a}{Kv}, \qquad v \leftarrow \frac{b}{K^\top u},$$

until the row and column sums equal the desired marginals.

Why this matters: simple matrix-vector products replace a large constrained optimization, making OT practical for modern ML.

Sinkhorn Intuition

- Small ε: sharper, closer to true OT, but harder numerically.

Sinkhorn Intuition

- Small ε: sharper, closer to true OT, but harder numerically.
- Large ε: smoother, faster, but more blurred.

Sinkhorn Intuition

- Small ε: sharper, closer to true OT, but harder numerically.
- Large ε: smoother, faster, but more blurred.
- Differentiable Sinkhorn losses are easy to plug into neural networks.

Sinkhorn Intuition

- Small ε: sharper, closer to true OT, but harder numerically.
- Large ε: smoother, faster, but more blurred.
- Differentiable Sinkhorn losses are easy to plug into neural networks.

.

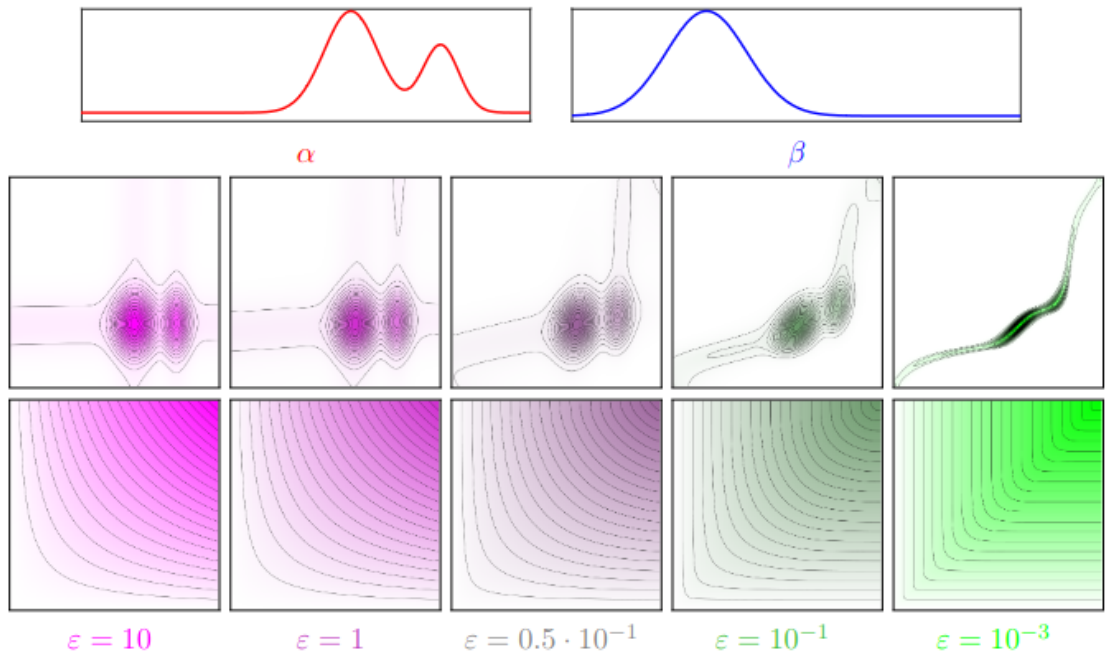


Figure: *Image taken from* Computational Optimal Transport *by Gabriel Peyré and Marco Cuturi.*

Dynamic Optimal Transport

Benamou and Brenier, *A Computational Fluid Mechanics Solution to the Monge-Kantorovich Problem*, 2000

For W_2, optimal transport can be written dynamically:

$$W_2^2(\mu_0, \mu_1) = \inf_{\rho_t, v_t} \int_0^1 \int \|v_t(x)\|^2 \rho_t(x) dx dt$$

subject to the continuity equation

$$\partial_t \rho_t + \nabla \cdot (\rho_t v_t) = 0,$$

and boundary conditions

$$\rho_0 = \mu_0, \qquad \rho_1 = \mu_1.$$

Dynamic Optimal Transport

Benamou and Brenier, *A Computational Fluid Mechanics Solution to the Monge-Kantorovich Problem*, 2000

For W_2, optimal transport can be written dynamically:

$$W_2^2(\mu_0, \mu_1) = \inf_{\rho_t, v_t} \int_0^1 \int \|v_t(x)\|^2 \rho_t(x) dx dt$$

subject to the continuity equation

$$\partial_t \rho_t + \nabla \cdot (\rho_t v_t) = 0,$$

and boundary conditions

$$\rho_0 = \mu_0, \qquad \rho_1 = \mu_1.$$

This is optimal control of a density.

Gradient Flows in Wasserstein Space

Jordan, Kinderlehrer, Otto, 1998; Ambrosio, Gigli, Savaré, 2005

Ordinary gradient descent:

$$x_{k+1} = \arg\min_x \left\{ f(x) + \frac{1}{2\tau}\|x - x_k\|^2 \right\}.$$

Gradient Flows in Wasserstein Space

Jordan, Kinderlehrer, Otto, 1998; Ambrosio, Gigli, Savaré, 2005

Ordinary gradient descent:

$$x_{k+1} = \arg\min_x \left\{ f(x) + \frac{1}{2\tau} \|x - x_k\|^2 \right\}.$$

Replace points by probability distributions and Euclidean distance by the Wasserstein distance:

$$\rho_{k+1} = \arg\min_\rho \left\{ \mathcal{F}(\rho) + \frac{1}{2\tau} W_2^2(\rho, \rho_k) \right\}.$$

Gradient Flows in Wasserstein Space

Jordan, Kinderlehrer, Otto, 1998; Ambrosio, Gigli, Savaré, 2005

Ordinary gradient descent:

$$x_{k+1} = \arg\min_x \left\{ f(x) + \frac{1}{2\tau} \|x - x_k\|^2 \right\}.$$

Replace points by probability distributions and Euclidean distance by the Wasserstein distance:

$$\rho_{k+1} = \arg\min_\rho \left\{ \mathcal{F}(\rho) + \frac{1}{2\tau} W_2^2(\rho, \rho_k) \right\}.$$

Key idea: the Wasserstein distance defines the geometry of the space of distributions.

Optimal Transport in ML

Application: Wasserstein GAN

Arjovsky, Chintala, Bottou, *Wasserstein GAN*, ICML 2017

GAN training aims to match the generated distribution

$$G_\theta(z) \sim \mu_\theta$$

to the data distribution

$$x \sim \mu_{\text{data}}.$$

Application: Wasserstein GAN

Arjovsky, Chintala, Bottou, *Wasserstein GAN*, ICML 2017

GAN training aims to match the generated distribution

$$G_\theta(z) \sim \mu_\theta$$

to the data distribution

$$x \sim \mu_{\text{data}}.$$

Using the Kantorovich–Rubinstein duality,

$$W_1(\mu_{\text{data}}, \mu_\theta) = \sup_{\|f\|_{\text{Lip}} \leq 1} \left(\mathbb{E}_{x \sim \mu_{\text{data}}}[f(x)] - \mathbb{E}_z[f(G_\theta(z))]\right).$$

Application: Wasserstein GAN

Arjovsky, Chintala, Bottou, *Wasserstein GAN*, ICML 2017

GAN training aims to match the generated distribution

$$G_\theta(z) \sim \mu_\theta$$

to the data distribution

$$x \sim \mu_{\text{data}}.$$

Using the Kantorovich–Rubinstein duality,

$$W_1(\mu_{\text{data}}, \mu_\theta) = \sup_{\|f\|_{\text{Lip}} \le 1} \left(\mathbb{E}_{x \sim \mu_{\text{data}}}[f(x)] - \mathbb{E}_z[f(G_\theta(z))]\right).$$

The critic is a neural network approximating the optimal 1-Lipschitz potential f.

Application: Wasserstein GAN

Arjovsky, Chintala, Bottou, *Wasserstein GAN*, ICML 2017

GAN training aims to match the generated distribution

$$G_\theta(z) \sim \mu_\theta$$

to the data distribution

$$x \sim \mu_{\text{data}}.$$

Using the Kantorovich–Rubinstein duality,

$$W_1(\mu_{\text{data}}, \mu_\theta) = \sup_{\|f\|_{\text{Lip}} \leq 1} \left(\mathbb{E}_{x \sim \mu_{\text{data}}}[f(x)] - \mathbb{E}_z[f(G_\theta(z))]\right).$$

The critic is a neural network approximating the optimal 1-Lipschitz potential f.

The generator is trained to minimize this estimated Wasserstein distance.

Application: Wasserstein GAN

Arjovsky, Chintala, Bottou, *Wasserstein GAN*, ICML 2017

GAN training aims to match the generated distribution

$$G_\theta(z) \sim \mu_\theta$$

to the data distribution

$$x \sim \mu_{\text{data}}.$$

Using the Kantorovich–Rubinstein duality,

$$W_1(\mu_{\text{data}}, \mu_\theta) = \sup_{\|f\|_{\text{Lip}} \le 1} \left(\mathbb{E}_{x \sim \mu_{\text{data}}}[f(x)] - \mathbb{E}_z[f(G_\theta(z))] \right).$$

The critic is a neural network approximating the optimal 1-Lipschitz potential f.

The generator is trained to minimize this estimated Wasserstein distance.

Why it helped: meaningful gradients can exist even when generated and real distributions barely overlap.

Optimal Transport for Domain Adaptation

Courty, Flamary, Tuia, Rakotomamonjy, *Optimal Transport for Domain Adaptation*, TPAMI 2017

In domain adaptation:

$$\mu_S \neq \mu_T,$$

but labels are cheap in source and expensive in target.

Optimal Transport for Domain Adaptation

Courty, Flamary, Tuia, Rakotomamonjy, *Optimal Transport for Domain Adaptation*, TPAMI 2017

In domain adaptation:

$$\mu_S \neq \mu_T,$$

but labels are cheap in source and expensive in target.

OT can align source and target samples by computing a coupling.

Optimal Transport for Domain Adaptation

Courty, Flamary, Tuia, Rakotomamonjy, *Optimal Transport for Domain Adaptation*, TPAMI 2017

In domain adaptation:

$$\mu_S \neq \mu_T,$$

but labels are cheap in source and expensive in target.

OT can align source and target samples by computing a coupling.

The coupling tells which source samples should influence which target samples.

Optimal Transport for Domain Adaptation

Courty, Flamary, Tuia, Rakotomamonjy, *Optimal Transport for Domain Adaptation*, TPAMI 2017

In domain adaptation:

$$\mu_S \neq \mu_T,$$

but labels are cheap in source and expensive in target.

OT can align source and target samples by computing a coupling.

The coupling tells which source samples should influence which target samples.

Intuition: use geometry to transfer labels across domains.

Sinkhorn Losses and Generative Models

Genevay et al., *Learning Generative Models with Sinkhorn Divergences*, AISTATS 2018; Feydy et al., *Interpolating between OT and MMD*, AISTATS 2019

For empirical distributions:

$$\hat{\mu} = \frac{1}{n} \sum_i \delta_{x_i}, \qquad \hat{\nu} = \frac{1}{m} \sum_j \delta_{y_j},$$

Sinkhorn gives a differentiable loss between point clouds.

Sinkhorn Losses and Generative Models

Genevay et al., *Learning Generative Models with Sinkhorn Divergences*, AISTATS 2018; Feydy et al., *Interpolating between OT and MMD*, AISTATS 2019

For empirical distributions:

$$\hat{\mu} = \frac{1}{n} \sum_i \delta_{x_i}, \qquad \hat{\nu} = \frac{1}{m} \sum_j \delta_{y_j},$$

Sinkhorn gives a differentiable loss between point clouds.

Applications include:

- generative modeling
- point cloud registration
- set prediction
- distributional representation learning

Gromov-Wasserstein Distances

Mémoli, 2011; Peyré, Cuturi, Solomon, *Gromov-Wasserstein Averaging*, ICML 2016; Vayer et al., 2019

What if two objects do not live in the same ambient space?

Gromov-Wasserstein Distances

Mémoli, 2011; Peyré, Cuturi, Solomon, *Gromov-Wasserstein Averaging*, ICML 2016; Vayer et al., 2019

What if two objects do not live in the same ambient space?

Gromov-Wasserstein compares internal distances:

$$\min_{\pi \in \Pi(\mu,\nu)} \sum_{i,j,k,l} \left| d_X(x_i, x_k) - d_Y(y_j, y_l) \right|^2 \pi_{ij} \pi_{kl}.$$

Gromov-Wasserstein Distances

Mémoli, 2011; Peyré, Cuturi, Solomon, *Gromov-Wasserstein Averaging*, ICML 2016; Vayer et al., 2019

What if two objects do not live in the same ambient space?

Gromov-Wasserstein compares internal distances:

$$\min_{\pi \in \Pi(\mu,\nu)} \sum_{i,j,k,l} \left| d_X(x_i, x_k) - d_Y(y_j, y_l) \right|^2 \pi_{ij} \pi_{kl}.$$

Why ML people care: graphs, shapes, meshes, molecules, and relational data often do not have aligned coordinates.

Gromov-Wasserstein at Scale, Beyond Squared Norms

G. Houry, J. Feydy, F.-X. Vialard, 2026

Classical Gromov-Wasserstein is a powerful tool for comparing structured data.

Gromov-Wasserstein at Scale, Beyond Squared Norms

G. Houry, J. Feydy, F.-X. Vialard, 2026

Classical Gromov-Wasserstein is a powerful tool for comparing structured data.

Unfortunately, solving the GW problem is computationally expensive.

Gromov-Wasserstein at Scale, Beyond Squared Norms

G. Houry, J. Feydy, F.-X. Vialard, 2026

Classical Gromov-Wasserstein is a powerful tool for comparing structured data.

Unfortunately, solving the GW problem is computationally expensive.

- cubic (or worse) computational complexity

Gromov-Wasserstein at Scale, Beyond Squared Norms

G. Houry, J. Feydy, F.-X. Vialard, 2026

Classical Gromov-Wasserstein is a powerful tool for comparing structured data.

Unfortunately, solving the GW problem is computationally expensive.

- cubic (or worse) computational complexity
- quadratic memory requirements

Gromov-Wasserstein at Scale, Beyond Squared Norms

G. Houry, J. Feydy, F.-X. Vialard, 2026

Classical Gromov-Wasserstein is a powerful tool for comparing structured data.

Unfortunately, solving the GW problem is computationally expensive.

- cubic (or worse) computational complexity
- quadratic memory requirements
- difficult to apply to large point clouds or graphs

Gromov-Wasserstein at Scale, Beyond Squared Norms

G. Houry, J. Feydy, F.-X. Vialard, 2026

Classical Gromov-Wasserstein is a powerful tool for comparing structured data.

Unfortunately, solving the GW problem is computationally expensive.

- cubic (or worse) computational complexity
- quadratic memory requirements
- difficult to apply to large point clouds or graphs

Question: can we preserve the flexibility of GW while making it scalable enough for modern machine learning?

Instead of directly optimizing the classical GW objective,

Gromov-Wasserstein at Scale, Beyond Squared Norms

G. Houry, J. Feydy, F.-X. Vialard, 2026

Instead of directly optimizing the classical GW objective,

the paper shows that a broad family of distortion functions can be rewritten as a sequence of much simpler problems.

Gromov-Wasserstein at Scale, Beyond Squared Norms

G. Houry, J. Feydy, F.-X. Vialard, 2026

Instead of directly optimizing the classical GW objective,

the paper shows that a broad family of distortion functions can be rewritten as a sequence of much simpler problems.

Gromov-Wasserstein $\Longrightarrow$ Optimal Transport + Feature Alignment

Gromov-Wasserstein at Scale, Beyond Squared Norms

G. Houry, J. Feydy, F.-X. Vialard, 2026

Instead of directly optimizing the classical GW objective,

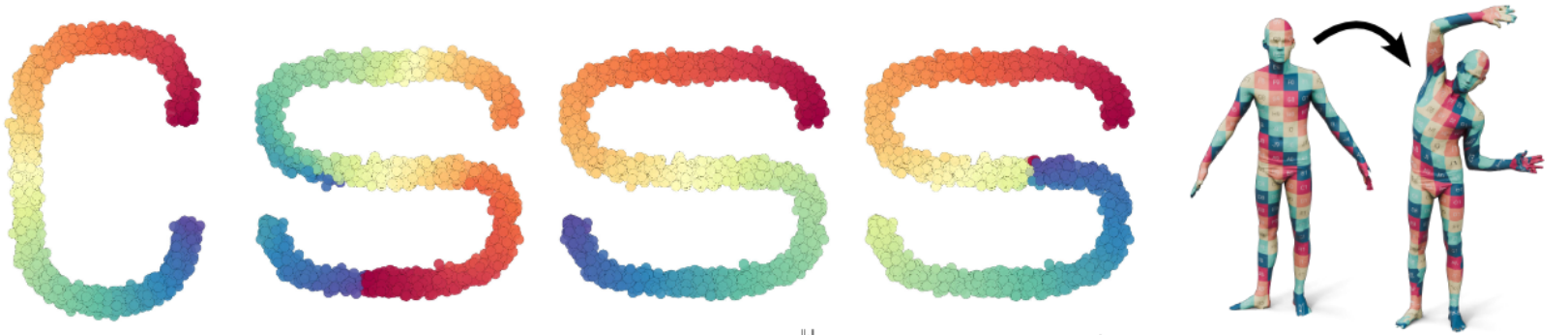
the paper shows that a broad family of distortion functions can be rewritten as a sequence of much simpler problems.

Gromov-Wasserstein $\Longrightarrow$ Optimal Transport + Feature Alignment

Intuition:
Rather than comparing every pair of distances simultaneously, compare suitable feature representations of the geometry.

Gromov-Wasserstein at Scale, Beyond Squared Norms

G. Houry, J. Feydy, F.-X. Vialard, 2026



(a) Source distribution (b) $c_X, c_Y = \|\cdot\|^2$ (c) $c_X, c_Y = 1 - e^{-\frac{\|\cdot\|}{0.5}}$ (d) $c_X, c_Y = \|\cdot\|^{1/2}$ (e) $N = 177k,\ c_X, c_Y = \|\cdot\|_{\text{geodesic}}$

Figure 1. (a) We optimize the GW objective of Eq. (2) to match a source distribution of points in the unit square ("C") with a target ("S"). Colors let us visualize the destination of every source point. (b) The preservation of squared distances between points has been studied extensively, but prioritizes the alignment of principal axes over smoothness. We propose a scalable GW solver for a broad class of penalties that may promote the preservation of topology (c) or find a balance between local and global structure (d). (e) This opens the door to applications on high-resolution data, such as those silhouettes sampled with 177k points each.

Conclusion

Control and Transport Are Not Separate

Control language	**Transport language**
State trajectory x_t	Distribution path ρ_t
Control u_t or velocity v_t	Transport plan / flow
Cost over time	Cost of moving mass
HJB / Pontryagin	Kantorovich dual / continuity equation
Stochastic control	Schrödinger bridge

Control and Transport Are Not Separate

Control language	**Transport language**
State trajectory x_t	Distribution path ρ_t
Control u_t or velocity v_t	Transport plan / flow
Cost over time	Cost of moving mass
HJB / Pontryagin	Kantorovich dual / continuity equation
Stochastic control	Schrödinger bridge

Takeaway: OT is often optimal control where the state is a probability distribution.

What to Remember

- **Optimal control** asks for the best trajectory under dynamics.

What to Remember

- **Optimal control** asks for the best trajectory under dynamics.
- **Stochastic control** asks for the best distribution over trajectories under noise.

What to Remember

- **Optimal control** asks for the best trajectory under dynamics.
- **Stochastic control** asks for the best distribution over trajectories under noise.
- **Optimal transport** asks for the best way to move one distribution into another.

What to Remember

- **Optimal control** asks for the best trajectory under dynamics.
- **Stochastic control** asks for the best distribution over trajectories under noise.
- **Optimal transport** asks for the best way to move one distribution into another.
- **Entropic OT / Schrödinger bridges** connect stochastic control and transport.

What to Remember

- **Optimal control** asks for the best trajectory under dynamics.
- **Stochastic control** asks for the best distribution over trajectories under noise.
- **Optimal transport** asks for the best way to move one distribution into another.
- **Entropic OT / Schrödinger bridges** connect stochastic control and transport.
- **ML applications** include neural ODEs, controllable diffusion, domain adaptation, generative modeling, graph/shape matching, and distribution losses.

References I


R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.


L. Pontryagin, V. Boltyanskii, R. Gamkrelidze, E. Mishchenko. *The Mathematical Theory of Optimal Processes*. 1962.


W. Fleming and H. Soner. *Controlled Markov Processes and Viscosity Solutions*. Springer, 2006.


R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud. Neural Ordinary Differential Equations. NeurIPS, 2018.


W. E. A Proposal on Machine Learning via Dynamical Systems. Communications in Mathematics and Statistics, 2017.


E. Haber and L. Ruthotto. Stable Architectures for Deep Neural Networks. Inverse Problems, 2017.


E. Todorov. Efficient Computation of Optimal Actions. PNAS, 2009.


H. J. Kappen. Path Integrals and Symmetry Breaking for Optimal Control Theory. Journal of Statistical Mechanics, 2005.


S. Levine. Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review. arXiv, 2018.


Y. Song, J. Sohl-Dickstein, D. Kingma, A. Kumar, S. Ermon, B. Poole. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR, 2021.


V. De Bortoli, J. Thornton, J. Heng, A. Doucet. Diffusion Schrödinger Bridge with Applications to Score-Based Generative Modeling. NeurIPS, 2021.


W. Tang. Fine-Tuning of Diffusion Models via Stochastic Control: Entropy Regularization and Beyond. arXiv, 2024.

References II


I. Azangulov et al. Adaptive Diffusion Guidance via Stochastic Optimal Control. arXiv, 2025.


C. Villani. *Optimal Transport: Old and New*. Springer, 2008.


G. Peyré and M. Cuturi. Computational Optimal Transport. Foundations and Trends in Machine Learning, 2019.


M. Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. NeurIPS, 2013.


J.-D. Benamou and Y. Brenier. A Computational Fluid Mechanics Solution to the Monge-Kantorovich Mass Transfer Problem. Numerische Mathematik, 2000.


R. Jordan, D. Kinderlehrer, F. Otto. The Variational Formulation of the Fokker-Planck Equation. SIAM Journal on Mathematical Analysis, 1998.


M. Arjovsky, S. Chintala, L. Bottou. Wasserstein GAN. ICML, 2017.


N. Courty, R. Flamary, D. Tuia, A. Rakotomamonjy. Optimal Transport for Domain Adaptation. IEEE TPAMI, 2017.


A. Genevay, G. Peyré, M. Cuturi. Learning Generative Models with Sinkhorn Divergences. AISTATS, 2018.


J. Feydy, T. Séjourné, F.-X. Vialard, S.-I. Amari, A. Trouvé, G. Peyré. Interpolating between Optimal Transport and MMD using Sinkhorn Divergences. AISTATS, 2019.


F. Mémoli. Gromov-Wasserstein Distances and the Metric Approach to Object Matching. Foundations of Computational Mathematics, 2011.


G. Peyré, M. Cuturi, J. Solomon. Gromov-Wasserstein Averaging of Kernel and Distance Matrices. ICML, 2016.

References III

Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, M. Le. Flow Matching for Generative Modeling. ICLR, 2023.