Machine Learning ⇐ Optimal Transport

IMA Data Science Seminar

https://www.mathcs.emory.edu/~lruthot/slides/2021-LR-IMA.pdf

Lars Ruthotto
Departments of Mathematics and Computer Science
Emory University

lruthotto@emory.edu
@lruthotto
Collaborators and Funding

Emory Funding:
- DMS 1751636
- BSF 2018209
- FA9550-20-1-0372
- ASCR 20-023231

Special Thanks:
- Organizers and staff of IPAM Long Program MLP 2019.
- Osher’s funding AFOSR MURI and ONR
Motivation: Machine Learning $\iff$ Optimal Transport

**Ex 1: Optimal Transport**

Given:
- initial density $\rho_0$
- target density $\rho_1$

Find $f : \mathbb{R}^d \to \mathbb{R}^d$ with minimal transport costs and

$$\rho_0(x) = \rho_1(f(x)) \det \nabla f(x)$$

goal: overcome curse of dimensionality

---

LR, S Osher, W Li, L Nurbekyan, S Wu Fung
An ML Framework for Solving High-Dimensional MFG/C
PNAS 117 (17), 9183-9193, 2020

**Ex 2: Generative Modeling**

Given:
- samples $x_1, \ldots, x_N$
- $\rho_1$ std. Gaussian

Find $f : \mathbb{R}^d \to \mathbb{R}^d$ that maximizes likelihood

$$\sum_{k=1}^{N} \rho_1(f(x_k)) \det \nabla f(x_k)$$

goal: improve reliability and efficiency

---

D Onken, S Wu Fung, X Li, LR
OT-Flow: Fast and Accurate CNF via OT
**Key goal:** create new connections between DL and Computational Mathematics.

→ robustness, interpretability, scalability, efficiency ←

→ solve impossible problems →

Agenda: Machine Learning meets Optimal Transport

ML → OT: New Tricks from Learning
- based on relaxed dynamical optimal transport
- combine macroscopic / microscopic / HJB equations
- neural networks for value function
- combine analytic gradients and automatic differentiation
- generalization to mean field games and control problems

OT → ML: Learning from Old Tricks
- variational inference via continuous normalizing flows
- applications: density estimation, generative modeling
- OT → uniqueness and regularity of dynamics
- HJB, solid numerics, and efficient implementation
- orders of magnitude speedup training and inference

LR, S Osher, W Li, L Nurbekyan, S Wu Fung
A ML Framework for Solving High-Dimensional MFG and MFC
PNAS 117 (17), 9183-9193, 2020

D Onken, S Wu Fung, X Li, LR
OT-Flow: Fast and Accurate CNF via OT
Machine Learning $\rightarrow$ High-Dimensional Optimal Transport
Relaxed Dynamical Optimal Transport (Benamou and Brenier 2000)

Given the initial density, \( \rho_0 \), and the target density, \( \rho_1 \), find the velocity \( v \) that minimizes the discrepancy between the push-forward of \( \rho_0 \) and \( \rho_1 \) and the transport costs, i.e.,

\[
\min_{v, \rho} \mathcal{J}_{\text{MFG}}(\rho, v) \overset{\text{def}}{=} \int_0^1 \int_{\Omega} \frac{1}{2} \| v(x, t) \|^2 \rho(x, t) \, dx \, dt + G(\rho(\cdot, 1), \rho_1)
\]

subject to \( \partial_t \rho + \nabla \cdot (\rho v) = 0 \), \( \rho(\cdot, 0) = \rho_0(\cdot) \)  (CE)

Examples for terminal cost \( G \): \( L_2 \), Kullback Leibler divergence,…

Side note: relaxed OT problem is a potential mean field game (MFG)
Relaxed Dynamic Optimal Transport: A Microscopic View

An agent with initial position $x \in \Omega$ at $t \in [0, T)$ aims at choosing $v_{x,t}$ that minimizes

$$J_{x,t}(v) = \int_{t}^{1} \frac{1}{2} \|v_{x,t}(s)\|^2 ds + G(z_{x,t}(1), \rho(z_{x,t}(1), 1)),$$

where their position changes according to

$$\partial_t z_{x,t}(s) = v_{x,t}(s), \quad t \leq s \leq 1, \quad z_{x,t}(t) = x.$$

- $G(x, \rho) = \frac{\delta G(\rho, \rho_1)}{\delta \rho}(x)$ (variational derivative of $G$)
- agent interacts with the population through $\rho$ and $G$
- $z_{x,t}(\cdot)$ is characteristic curve of (CE) starting at $x$
- transformation defined by $f(x) = z_{x,0}(1)$
Hamilton-Jacobi-Bellman (HJB) Equation

First-order optimality conditions of relaxed OT are (Lasry and Lions 2007)

\[-\partial_t \Phi(x, t) + \frac{1}{2} \| \nabla \Phi(x, t) \|^2 = 0, \quad \Phi(x, 1) = G(x, \rho(x, 1))\] (HJB)

and optimal strategy is \( v(x, t) = -\nabla \Phi(x, t) \), which gives

\[\partial_t \rho(x, t) - \nabla \cdot (\rho(x, t) \nabla \Phi(x, t)) = 0, \quad \rho(x, 0) = \rho_0(x)\] (CE)

challenges: forward-backward structure and high-dimensionality of PDE system
Machine Learning for High-Dimensional OT: Overview

Three options for solving the problem

1. minimize $J_{\text{MFG}}$ w.r.t. $(\rho, v)$, or $(\rho, -\nabla \Phi)$ (variational problem)
2. minimize $J_{x,t}$ w.r.t. $v$ or $-\nabla \Phi$ for some points $x$ (microscopic view)
3. compute value function by solving (HJB) and (CE) (high-dimensional PDEs)

Idea: Combine advantages of the above to tackle curse of dimensionality

- formulate as variational problem. minimize $J_{\text{MFG}}(\rho, -\nabla \Phi)$
- eliminate (CE) with Lagrangian PDE solver $\rightsquigarrow\mathbb{H}$ mesh-free, parallel
- parameterize $\Phi$ with NN $\rightsquigarrow\mathbb{H}$ universal approximator, mesh-free, cheap(?)
- penalize violations of (HJB) $\rightsquigarrow\mathbb{H}$ regularity, global convergence(?)
Machine Learning Method
Lagrangian Method for Continuity Equation

Assume $\Phi$ given. Then, the solution to

$$\partial_t \rho(x, t) - \nabla \cdot (\rho(x, t) \nabla \Phi(x, t)) = 0, \quad \rho(x, 0) = \rho_0(x)$$

satisfies

$$\rho(z(x, t), t) \det \nabla z(x, t) = \rho_0(x)$$

along the characteristic curve

$$\partial_t z(x, t) = -\nabla \Phi(z(x, t)), \quad z(x, 0) = x.$$  

instead of computing $\det \nabla z(x, t)$ (cost $\mathcal{O}(d^3)$ flops) use

$$l(x, t) \overset{\text{def}}{=} \log \det(\nabla z(x, t)) = \int_0^1 \Delta \Phi(z(x, t), t) dt$$

Hint: Compute $z$ and $l$ in one ODE solve (parallelize over $x_1, x_2, \ldots$).
Lagrangian Method for Optimal Transport

\[
\begin{align*}
\text{minimize}_\Phi & \quad \mathbb{E}_{\rho_0} \left[ c_L(x, 1) + G(z(x, 1)) + \alpha_1 c_H(x, 1) + \alpha_2 \| \Phi(z(x, 1), 1) - G(z(x, 1)) \| \right] \\
\text{subject to} & \quad \partial_t \begin{pmatrix} z(x, t) \\ l(x, t) \\ c_L(x, t) \\ c_H(x, t) \end{pmatrix} = \begin{pmatrix} -\nabla \Phi(z(x, t), t) \\ -\Delta \Phi(z(x, t), t) \\ \frac{1}{2} \| \nabla \Phi(z(x, t), t) \|^2 \\ |\partial_t \Phi(z(x, t), t) + \frac{1}{2} \| \nabla \Phi(z(x, t), t) \|^2| \end{pmatrix}, \quad t \in (0, 1] \\
& \quad z(x, 0) = x, \quad l(x, 0) = c_L(x, 0) = c_H(x, 0) = 0
\end{align*}
\]

- $z$ and $l = \log \text{det}$ needed to solve continuity eq. (CE)
- $c_L$ and $c_H$ accumulate cost along characteristic
- $\alpha_1, \alpha_2$: penalty parameters for HJB violation
- Discretize dynamics with $n_t$ steps of Runge-Kutta-4
- Discretize $\mathbb{E}$ with Monte Carlo
- Can use SA (SGD, ADAM, . . . ) or SAA (BFGS, Newton, . . . ) methods
- No grid needed and computation can be parallelized over $x$

Next, parameterize $\Phi$ with NN. Needed: $\nabla \Phi$ and $\Delta \Phi$
Neural Network Model for Value Function

Let \( s = (x, t) \in \mathbb{R}^{d+1} \) and use (NN + quadratic) model for value function

\[
\Phi(s, \theta) = w^\top N(s, \theta_N) + \frac{1}{2} s^\top A s + c^\top s + b, \quad \theta = (w, \theta_N, \text{vec}(A), c, b)
\]

\( N(s, \theta_N) \) is an \( M \)-layer ResNet with weights \( \theta_N = (\text{vec}(K_0), \ldots, \text{vec}(K_M), b_0, \ldots, b_M) \).

forward propagation:

\[
\begin{align*}
    u_0 &= \sigma(K_0 s + b_0) \\
    u_1 &= u_0 + h \sigma(K_1 u_0 + b_1) \\
    &\vdots & \vdots \\
    u_M &= u_{M-1} + h \sigma(K_M u_{M-1} + b_M),
\end{align*}
\]

Output: \( w^\top u_M = w^\top N(s, \theta_N) \)

Remark: need also \( \nabla_s \Phi \) and \( \Delta x \Phi \)

1. automatic differentiation, limited to matrix-vector products

\[
\Delta x \Phi(s, \theta) = \sum_{k=1}^{d} e_k^\top \nabla_x^2 \Phi(s, \theta) e_k
\]

2. trace estimators add inaccuracy

3. better to derive derivatives manually

4. efficient algorithm \( \sim \mathcal{O}(m^2 \cdot d) \) flops

5. implementation easily parallelizes
Computing the Laplacian of Value Function

\[ \Delta \Phi(s, \theta) = \text{tr} \left( E^T (\nabla_s^2(N(s, \theta_N)w) + A)E \right) \quad \text{for} \quad E = \text{eye}(d+1, d) \]

Second term trivial. Focus on NN part and use forward mode for first layer

\[ t_0 = \text{tr} \left( E^T \nabla_s (K_0^T \text{diag}(\sigma''(K_0s + b_0))z_1)E \right) \]
\[ = (\sigma''(K_0s + b_0) \odot z_1)^T ((K_0E) \odot (K_0E))1, \]

(\odot \text{Hadamard product, } 1 = \text{ones}(d, 1))

Get \[ \Delta(N(s, \theta_N)w) = t_0 + h \sum_{i=1}^{M} t_i \text{ where for } i \geq 1 \]

\[ t_i = \text{tr} \left( J_{i-1}^T \nabla_s (K_i^T \text{diag}(\sigma''(K_iu_{i-1}(s) + b_i))z_{i+1})J_{i-1} \right) \]
\[ = (\sigma''(K_iu_{i-1} + b_i) \odot z_{i+1})^T ((K_iJ_{i-1}) \odot (K_iJ_{i-1}))1. \]

Here, \[ J_{i-1} = \nabla_s u_{i-1} \in \mathbb{R}^{m \times d} \] is a Jacobian matrix (update during forward pass)

overall cost when \[ K_0 \in \mathbb{R}^{m \times (d+1)} \] is \[ O(m^2 \cdot d) \text{ FLOPS} \]
Experiment 1: Benefit of HJB Penalty

HJB penalty improves accuracy and (!) lowers computational costs
Experiment 2: Scalability to Higher Dimensions

<table>
<thead>
<tr>
<th>$d$</th>
<th>$n$</th>
<th>$\mathcal{L}$</th>
<th>$\mathcal{G}$</th>
<th>$C_{\text{HJB}}$</th>
<th>time/iter (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2,304</td>
<td>9.99e+00</td>
<td>7.01e-01</td>
<td>1.17e+00</td>
<td>2.038</td>
</tr>
<tr>
<td>10</td>
<td>6,400</td>
<td>1.01e+01</td>
<td>8.08e-01</td>
<td>1.21e+00</td>
<td>8.256</td>
</tr>
<tr>
<td>50</td>
<td>16,384</td>
<td>1.01e+01</td>
<td>6.98e-01</td>
<td>2.94e+00</td>
<td>81.764</td>
</tr>
<tr>
<td>100</td>
<td>36,864</td>
<td>1.01e+01</td>
<td>8.08e-01</td>
<td>1.21e+00</td>
<td>301.043</td>
</tr>
</tbody>
</table>

Qualitatively similar results in all dimensions / moderate growth of runtime
Experiment: Comparison with Eulerian Solver

Eulerian scheme (Haber and Horesh 2015):

- dynamical OT formulation
- conservative finite volume
- leads to convex optimization
- solved to high accuracy with Newton’s method

<table>
<thead>
<tr>
<th></th>
<th># parameters</th>
<th>$J_{MFG}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eulerian, fine</td>
<td>3,080,448</td>
<td>1.066e+01 (100.00%)</td>
</tr>
<tr>
<td>Eulerian, coarse</td>
<td>376,960</td>
<td>1.082e+01 (101.47%)</td>
</tr>
<tr>
<td>MFGnet ($n_t = 2$)</td>
<td>637</td>
<td>1.072e+01 (100.59%)</td>
</tr>
<tr>
<td>MFGnet ($n_t = 8$)</td>
<td>637</td>
<td>1.063e+01 (99.69%)</td>
</tr>
</tbody>
</table>
Experiment: Comparison of Value Functions

Take away: Eulerian (≈ 3M parameters) and Lagrangian-ML (637 parameters) give comparable accuracy.
Extension: Mean Field Games / Mean Field Control

Model large populations of rational agents playing non-cooperative differential game.

\[
\min_{v, \rho} \mathcal{J}_{MFG}(v, \rho) \overset{\text{def}}{=} \int_0^1 \int_{\mathbb{R}^d} L(x, v(x, t)) \rho(x, t) dx dt + \int_0^1 \mathcal{F}(\rho(\cdot, t)) dt + \mathcal{G}(\rho(\cdot, 1))
\]

subject to

\[
\partial_t \rho(x, t) + \nabla \cdot (\rho(x, t)v(x, t)) = 0, \quad \rho(x, 0) = \rho_0(x),
\]

Use running costs \(\mathcal{F}\) to model, e.g.,

- congestion

\[
\mathcal{F}_E(\rho) = \int_{\mathbb{R}^d} \rho(x) \log(\rho(x)) dx
\]

- spatio-temporal preference

\[
\mathcal{F}_P(\rho) = \int_{\mathbb{R}^d} Q(x) \rho(x, t) dx
\]
More To Watch

Levon Nurbekyan @ IPAM Opening Workshop

*Computational methods for mean-field games*

Samy Wu Fung @ Emory Scientific Computing Seminar

*A GAN-based Approach for High-Dimensional Stochastic Mean Field Games*

https://bit.ly/3cELBmW

https://youtu.be/Z-GA61AZAO0
Optimal Transport $\rightarrow$ Continuous Normalizing Flows
Continuous Normalizing Flows (CNF)

Likelihood Maximization

Given samples $x_1, x_2, \ldots, x_N \in \mathbb{R}^d$, find a velocity $v$ that maximizes the likelihood of the samples w.r.t. the push-forward of the standard normal distribution $\rho_1$, i.e.,

$$v, z$$

subject to

$$z(x_k, 0) = x_k$$ for all $k$.

Recall: $l(x_k, 1) = \log \det(\nabla z(x_k, 1))$

---

W. Grathwohl et al.

OT-Flow: Regularized Continuous Normalizing Flow

Given samples $x_1, x_2, \ldots, x_N \sim \rho_0$, find the value function $\Phi$ such that the flow given by $v = -\nabla \Phi$ maximizes the likelihood of the samples w.r.t. the standard normal distribution $\rho_1$, i.e.,

\[
\min_{\Phi} \mathbb{E}_{x \sim \rho_0} \left[ \frac{1}{2} \|z(x, 1)\|^2 - l(x, 1) + c_L(x, 1) + \alpha_1 c_H(x, 1) \right]
\]

subject to

\[
\frac{\partial}{\partial t} \begin{pmatrix} z(x, t) \\ l(x, t) \\ c_L(x, t) \\ c_H(x, t) \end{pmatrix} = \begin{pmatrix} -\nabla \Phi(z(x, t), t) \\ -\Delta \Phi(z(x, t), t) \\ \frac{1}{2} \|\nabla \Phi(z(x, t), t)\|^2 \\ |\partial_t \Phi(z(x, t), t) + \frac{1}{2} \|\nabla \Phi(z(x, t), t)\|^2| \end{pmatrix}
\]

$z(x, 0) = x$, $l(x, 0) = c_L(x, 0) = c_H(x, 0) = 0$
Trace Computation: Runtime and Accuracy

- Exact computation with automatic differentiation (AD)

\[ \text{trace}(\nabla v(x)) = \sum_{i=1}^{d} e_i^\top (\nabla v(x)^\top e_i) \]

- exact  \( O(m \cdot d^2) \) FLOPS

- trace estimator with AD

\[ \text{trace}(\nabla v(x)) = \mathbb{E}_w \left[w^\top (\nabla v(x)^\top w)\right] \]

\[ \approx \frac{1}{S} \sum_{k=1}^{S} (w_k)^\top (\nabla v(x)^\top w_k) \]

- inexact  \( O(m \cdot S \cdot d) \) FLOPS

**OT-Flow:** exact trace computation (highly parallel) using \( O(m^2 \cdot d) \) FLOPS.
OT-Flow: Two-Dimensional Examples

- moons
- circles
- pinwheel
- checkerboard

samples
density estimate
OT-Flow vs. FFJORD, RNODE: UCI Datasets

- OT-Flow yields competitive accuracy w.r.t. MMD
- FFJORD, RNODE: between $2\times$ and $22\times$ more weights
- OT-Flow considerably faster in training and testing.
OT-Flow Example: Generative Modeling MNIST

- let \( y_1, y_2, \ldots \in \mathbb{R}^{768} \) MNIST images
- train encoder \( E : \mathbb{R}^{784} \rightarrow \mathbb{R}^{128} \) and decoder \( D : \mathbb{R}^{128} \rightarrow \mathbb{R}^{784} \) s.t. 
  \( D(E(y)) \approx y \)
- latent space representation of data 
  \( x_j = E(y_j) \) for all \( j \).
- train OT-Flow \( f \) that maps \( \{x_j\}_j \) to 
  \( \rho_1 \sim \mathcal{N}(0, I_{128}) \)
- interpolate between two images \( y_1, y_2 \) in latent space and get new image 
  \[ y(\lambda) = D(f^{-1}(\lambda f(E(y_1)) + (1-\lambda) f(E(y_2)))) \]
Conclusions
# OT-Flow - Fast Continuous Normalizing Flows in PyTorch

**GitHub Repository:** [EmoryMLIP/OT-Flow](https://github.com/EmoryMLIP/OT-Flow)

- **Julia implementation for more general MFGs:** [MFGnet.jl](https://github.com/EmoryMLIP/MFGnet.jl)
Σ: Machine Learning meets Optimal Transport

Machine Learning → Optimal Transport
- ML attractive for **high-dimensional** PDEs, control, . . .
- MFGnet: mesh-free solver for variational problem and combine. . .
  - microscopic: Lagrangian method for continuity and HJB eqs.
  - macroscopic: variational problem, new penalties for HJB eq.
- details matter: models, numerics, architecture, training, . . .
- surprise: ML solution competitive to convex programming

Optimal Transport → Continuous Normalizing Flows
- OT regularization: well-posed simplifies time integration
- discretize-then-optimize + HJB penalty → very few time steps
- don’t take chances: use exact trace computation
- OT-Flow speeds up training and testing by $\approx 10x$

---

LR, S Osher, W Li, L Nurbekyan, S Wu Fung
A ML Framework for Solving High-Dimensional MFG and MFC
PNAS 117 (17), 9183-9193, 2020

D Onken, S Wu Fung, X Li, LR
OT-Flow: Fast and Accurate CNF via OT