Machine Learning $\leftrightarrow$ Optimal Transport

Old solutions to new problems and vice versa

Lars Ruthotto
Departments of Mathematics and Computer Science, Emory University
lruthotto@emory.edu
@lruthotto
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 $\leadsto$ 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*
in preparation, available soon.
A Machine Learning Framework for High-Dimensional OT (and more)
Title ML → OT Lag NN Exp OT → CNF Σ

Team and Acknowledgements

Emory Funding: [NSF] DMS 1751636 [BSF] US-Israel BSF 2018209

UCLA Funding: AFOSR MURI FA9550-18-1-0502 and FA9550-18-1-0167, ONR N00014-18-1-2527

Special thanks: Organizers and staff of IPAM Long Program MLP 2019.
Dynamic Optimal Transport (Benamou and Brenier, ’00)

Given the initial density, $\rho_0$, and the target density, $\rho_1$, find the velocity $v$ that renders the push-forward of $\rho_0$ equal to $\rho_1$ and minimizes the transport costs, i.e.,

$$\text{minimize}_{v,\rho} \, \int_0^1 \int_\Omega \frac{1}{2} \|v(x, t)\|^2 \rho(x, t) dx dt$$

subject to

$$\partial_t \rho + \nabla \cdot (\rho v) = 0, \quad \rho(\cdot, 0) = \rho_0(\cdot), \quad \rho(\cdot, 1) = \rho_1(\cdot)$$
Dynamic Optimal Transport (Benamou and Brenier, ’00)

Given the initial density, $\rho_0$, and the target density, $\rho_1$, find the velocity $v$ that renders the push-forward of $\rho_0$ equal to $\rho_1$ and minimizes the transport costs, i.e.,

\[
\begin{align*}
\text{minimize}_{v, \rho} & \quad \int_0^1 \int_{\Omega} \frac{1}{2} \|v(x, t)\|^2 \rho(x, t) \, dx \, dt \\
\text{subject to} & \quad \partial_t \rho + \nabla \cdot (\rho v) = 0, \quad \rho(\cdot, 0) = \rho_0(\cdot), \quad \rho(\cdot, 1) = \rho_1(\cdot)
\end{align*}
\]
Relaxed Dynamical Optimal Transport

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.,

$$\text{minimize}_{v, \rho} J_{\text{MFG}}(\rho, v) \overset{\text{def}}{=} \int_0^1 \int_{\Omega} \frac{1}{2} \|v(x, t)\|^2 \rho(x, t) dx dt + G(\rho(\cdot, 1), \rho_1)$$

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, . . .

Take away: relaxed OT problem is a potential mean field game (MFG)
Relaxed Dynamical Optimal Transport

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.,

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

subject to

$$\partial_t \rho + \nabla \cdot (\rho v) = 0, \quad \rho(\cdot, 0) = \rho_0(\cdot)$$

(CE)

Examples for terminal cost $G$: $L_2$, Kullback Leibler divergence, . . .

Take away: relaxed OT problem is a potential mean field game (MFG)
Relaxed OT: A Microscopic View

A single agent with initial position $x \in \Omega$ aims at choosing $v$ that minimizes

$$J_{x,0}(v) = \int_0^1 \frac{1}{2} \|v(s)\|^2 ds + G(z(1), \rho(z(1), 1)),$$

where their position changes according to

$$\partial_t z(s) = v(s), \quad 0 \leq s \leq 1, \quad z(0) = x.$$

- $G(x, \rho) = \frac{\delta G(\rho, \rho_1)}{\delta \rho}(x)$ (variational derivative of $G$)
- agent interacts with the population through $\rho$ and $G$
- $z(\cdot)$ is characteristic curve of (CE) starting at $x$
Relaxed OT: A Microscopic View

A single agent with initial position $x \in \Omega$ aims at choosing $v$ that minimizes

$$J_{x,0}(v) = \int_0^1 \frac{1}{2} \|v(s)\|^2 ds + G(z(1), \rho(z(1), 1)),$$

where their position changes according to

$$\partial_t z(s) = v(s), \quad 0 \leq s \leq 1, \quad z(0) = x.$$

- $G(x, \rho) = \frac{\delta G(\rho, \rho_1)}{\delta \rho}(x)$ (variational derivative of $G$)
- agent interacts with the population through $\rho$ and $G$
- $z(\cdot)$ is characteristic curve of (CE) starting at $x$

Useful to define the value of an agent’s state $(x, t)$ as

$$\Phi(x, t) = \inf_v J_{x,t}(v)$$
Hamilton-Jacobi-Bellman (HJB) Equation

Lasry & Lions ’06: First-order optimality conditions of relaxed OT are

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

(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)

challenge: forward-backward structure and high-dimensional PDE
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)
Machine Learning for High-Dimensional OT: Overview

Three options for solving the problem

1. minimize $\mathcal{J}_{\text{MFG}}$ 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
Machine Learning for High-Dimensional OT: Overview

Three options for solving the problem

1. minimize $\mathcal{J}_{\text{MFG}}$ 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 $\mathcal{J}_{\text{MFG}}(\rho, -\nabla \Phi)$
- eliminate (CE) with Lagrangian PDE solver $\leadsto$ mesh-free, parallel
- parameterize $\Phi$ with NN $\leadsto$ universal approximator, mesh-free, cheap(?)
- penalize violations of (HJB) $\leadsto$ regularity, global convergence(?)

Title ML $\rightarrow$ OT, Lag, NN, Exp $\rightarrow$ OT $\rightarrow$ CNF $\Sigma$
Lagrangian Method
Lagrangeian 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

$$\partial_t z(x, t) = -\nabla \Phi(z(x, t)), \quad z(x, 0) = x.$$
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 $O(d^3)$ 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

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

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

- \( z \) and \( l = \log \det \) 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
Deep Learning Revolution (?)

- deep learning: use neural networks (from ≈ 1950’s) with many hidden layers
- able to "learn" complicated patterns from data
- applications: classification, face recognition, segmentation, driverless cars, ...
- recent success fueled by: massive data sets, computing power
- A few recent references:
  - Data Scientist: Sexiest Job of the 21st Century, Harvard Business Rev ’17
  - A radical new neural network design could overcome big challenges in AI, MIT Tech Review ’18
Deep Learning Revolution (?)

\[
\begin{align*}
Y_{j+1} &= \sigma(K_j Y_j + b_j) \\
Y_{j+1} &= Y_j + \sigma(K_j Y_j + b_j) \\
Y_{j+1} &= Y_j + \sigma(K_{j,2}\sigma(K_{j,1} Y_j + b_{j,1}) + b_{j,2}) \\
\vdots
\end{align*}
\]

(Notation: $Y_j$ : features, $K_j, b_j$ : weights, $\sigma$ : activation)

depth learning: use neural networks (from $\approx 1950$’s) with many hidden layers
able to "learn" complicated patterns from data
applications: classification, face recognition, segmentation, driverless cars, . . .
recent success fueled by: massive data sets, computing power
A few recent references:

- Data Scientist: Sexiest Job of the 21st Century, Harvard Business Rev ’17
- A radical new neural network design could overcome big challenges in AI, MIT Tech Review ’18
Neural Network Model for Value Function

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

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

$N(s, \theta_N)$ is an $M$-layer ResNet with weights $\theta_N = (\text{vec}(K_0), \ldots, \text{vec}(K_M), b_0, \ldots, b_M)$. 
Neural Network Model for Value Function

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

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

\( N(s, \theta_N) \) 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:

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

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

\[
\begin{align*}
    u_0 &= s \\
    u_1 &= \sigma(K_0u_0 + b_0) \\
    \vdots & \quad \vdots \\
    u_M &= u_{M-1} + h\sigma(K_Mu_{M-1} + b_M),
\end{align*}
\]

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

backward propagation:

\[
\begin{align*}
    z_{M+1} &= w \\
    z_M &= z_{M+1} + hK_M^\top \text{diag}(\sigma'(K_Mu_{M-1} + b_M))z_{M+1}, \\
        & \quad \vdots \\
    z_1 &= z_2 + hK_1^\top \text{diag}(\sigma'(K_1u_0 + b_1))z_2, \\
    z_0 &= K_0^\top \text{diag}(\sigma'(K_0s + b_0))z_1,
\end{align*}
\]

Output: \( z_0 = \nabla_s(w^\top N(s, \theta_N)) \)

Next: Compute \( \Delta \Phi(s, \theta) = \text{tr} \left( E^\top \left( \nabla_s^2 N(s, \theta_N)w + A \right) E \right) \),
Computing the Laplacian of Value Function

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

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

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

\[ t_0 = \text{tr} \left( E^\top \nabla_s (K_0^\top \text{diag}(\sigma''(K_0s + b_0))z_1)E \right) \]
\[ = (\sigma''(K_0s + b_0) \odot z_1)^\top ((K_0E) \odot (K_0E)) \mathbf{1}, \]

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

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

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

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

Here, \( J_{i-1} = \nabla_s u_{i-1}^T \in \mathbb{R}^{m \times d} \) is a Jacobian matrix (update during forward pass)
Computing the Laplacian of Value Function

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

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 \( \Delta (N(s, \theta_N)w) = t_0 + h \sum_{i=1}^{M} t_i \) 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, \( J_{i-1} = \nabla_s u_{i-1}^T \in \mathbb{R}^{m \times d} \) is a Jacobian matrix (update during forward pass)

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

HJB penalty improves accuracy and(!) lowers computational costs
Experiment 3: Comparison with Eulerian Solver

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

---

E Haber, R Horesh
A Multilevel Method for the Solution of Time Dependent Optimal Transport,
NM-TMA 8(1), 2015.
Experiment 3: Comparison with Eulerian Solver

Eulerian scheme:

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

Comparison:

<table>
<thead>
<tr>
<th></th>
<th># parameters</th>
<th>$I_{MFG}$</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>

E Haber, R Horesh
Experiment 3: Comparison of Value Functions

\[ \rho_0, \text{ initial density} \]
\[ \rho_1, \text{ target density} \]
\[ t = 0, \text{ initial time} \]
\[ t = 1, \text{ final time} \]

\[ \Phi_{\text{Lag}}(\cdot, t), \text{Lagrangian ML} \]
\[ \Phi_{\text{Eul}}(\cdot, t), \text{Eulerian FV} \]

error, \[ |\Phi_{\text{Lag}}(\cdot, t) - \Phi_{\text{Eul}}(\cdot, t)| \]

Take away: Eulerian (≈ 3M parameters) and Lagrangian-ML (637 parameters) give comparable accuracy.
Experiment 3: Comparison of Value Functions

$\rho_0$, initial density
$\rho_1$, target density
initial time, $t = 0$
final time, $t = 1$
$\Phi_{\text{Lag}}(\cdot, t)$, Lagrangian ML
$\Phi_{\text{Eul}}(\cdot, t)$, Eulerian FV
error, $|\Phi_{\text{Lag}}(\cdot, t) - \Phi_{\text{Eul}}(\cdot, t)|$

Take away: Eulerian ($\approx 3M$ 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.
Extension: Mean Field Games / Mean Field Control

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

\[
\minimize_{v, \rho} J_{MFG}(v, \rho) \overset{\text{def}}{=} \int_0^1 \int_{\mathbb{R}^d} L(x, v(x, t)) \rho(x, t)dxdt + \int_0^1 F(\rho(\cdot, t))dt + G(\rho(\cdot, 1))
\]

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 \( F \) to model, e.g.,

- congestion

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

- spatio-temporal preference

\[
F_P(\rho) = \int_{\mathbb{R}^d} Q(x) \rho(x, t)dx
\]
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

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.,

$$
\max_{v, z} \frac{1}{N} \sum_{k=1}^{N} \rho_1(z(x_k, 1)) \cdot \det \nabla (z(x_k, 1))
$$

subject to $\partial_t z(x_k, t) = v(z(x_k, t), t)$,

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

W Grathwohl et al.
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.,

$$\min_{v, z} G_{CNF}(v, z) := \frac{1}{N} \sum_{k=1}^{N} \left( \frac{1}{2} \| z(x_k, 1) \|^2 - l(x_k, 1) \right)$$

subject to

$$\frac{\partial}{\partial t} \begin{pmatrix} z(x_k, s) \\ l(x_k, s) \end{pmatrix} = \begin{pmatrix} v(z(x_k, s), s) \\ \text{trace}(\nabla_v(z(x_k, s), s)) \end{pmatrix}$$

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

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

W Grathwohl et al.

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.,

\[
\min_{v, z} G_{CNF}(v, z) := \frac{1}{N} \sum_{k=1}^{N} \left( \frac{1}{2} \| z(x_k, 1) \|^2 - l(x_k, 1) \right)
\]

subject to

\[
\partial_t \begin{pmatrix} z(x_k, s) \\ l(x_k, s) \end{pmatrix} = \begin{pmatrix} v(z(x_k, s), s) \\ \text{trace}(\nabla v(z(x_k, s), s)) \end{pmatrix}
\]

with \(z(x_k, 0) = x_k\) and \(l(x_k, 0) = 0\) for all \(k\).

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

\[\text{W Grathwohl et al.}
\]
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.,

\[
\minimize_{v, z} G_{\text{CNF}}(v, z) := \frac{1}{N} \sum_{k=1}^{N} \left( \frac{1}{2} \|z(x_k, 1)\|^2 - l(x_k, 1) \right)
\]

subject to

\[
\partial_t \begin{pmatrix} z(x_k, s) \\ l(x_k, s) \end{pmatrix} = \begin{pmatrix} v(z(x_k, s), s) \\ \text{trace}(\nabla v(z(x_k, s), s)) \end{pmatrix}
\]

with \(z(x_k, 0) = x_k\) and \(l(x_k, 0) = 0\) 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 \( x_1, x_2, \ldots, x_N \in \mathbb{R}^d \), find the value function \( \Phi \) such that the flow given by \( \nu = -\nabla \Phi \) maximizes the likelihood of the samples w.r.t. the standard normal distribution \( \rho_1 \), i.e.,

\[
\min_{\nu, \Phi} \frac{1}{N} \sum_{k=1}^{N} \left[ \frac{1}{2} \| z(x_k, 1) \|^2 - l(x_k, 1) + \beta_1 c_L(x_k, 1) + \beta_2 c_H(x_k, 1) \right]
\]

subj. to \( \partial_t z(x_k, t) = \nu(z(x_k, t), t), \quad z(x_k, 0) = x_k \quad \forall k \)
OT-Flow: Regularized Continuous Normalizing Flow

Given samples $x_1, x_2, \ldots, x_N \in \mathbb{R}^d$, find the value function $\Phi$ such that the flow given by $\nu = -\nabla \Phi$ maximizes the likelihood of the samples w.r.t. the standard normal distribution $\rho_1$, i.e.,

$$
\min_{\nu, z} \frac{1}{N} \sum_{k=1}^{N} \left[ \frac{1}{2} \|z(x_k, 1)\|^2 - l(x_k, 1) + \beta_1 c_L(x_k, 1) + \beta_2 c_H(x_k, 1) \right]
$$

subj. to $\partial_t z(x_k, t) = \nu(z(x_k, t), t)$, $z(x_k, 0) = x_k$ $\forall k$
OT-Regularized Continuous Normalizing Flow

OT-Flow: Regularized Continuous Normalizing Flow

Given samples $x_1, x_2, \ldots, x_N \in \mathbb{R}^d$, find the value function $\Phi$ such that the flow given by $\nu = -\nabla \Phi$ maximizes the likelihood of the samples w.r.t. the standard normal distribution $\rho_1$, i.e.,

$$\min_{\nu, z} \frac{1}{N} \sum_{k=1}^{N} \left[ \frac{1}{2} \|z(x_k, 1)\|^2 - l(x_k, 1) + \beta_1 c_L(x_k, 1) + \beta_2 c_H(x_k, 1) \right]$$

subj. to $\partial_t z(x_k, t) = \nu(z(x_k, t), t), \quad z(x_k, 0) = x_k \quad \forall k$

- 📋 provides uniqueness
- ⏰ more efficient time integration

L Yang, GE Karniadakis
Potential Flow Generator with $L_2$ OT Regularity for Generative Models. 

L Zhang, Weinan E, L Wang
Monge-Ampère Flow for Generative Modeling, 

C Finlay, JH Jacobsen, L Nurbekyan, AM Oberman
How to train your neural ODE, 
Trace Computation: Runtime and Accuracy

- **Exact computation with automatic differentiation (AD)**

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

- \(\text{exact} \quad O(m \cdot d^2) \text{ FLOPS}\)

- **trace estimator with AD**

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

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

- \(\text{inexact} \quad O(m \cdot S \cdot d) \text{ FLOPS}\)
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
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](image)

  - 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](image)

  - 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

![Samples and Density Estimates](image-url)
OT-Flow vs. FFJORD: Comparison for UCI Datasets

- FFJORD slightly superior to OT-Flow w.r.t. MMD
- FFJORD needs between $2 \times$ and $22 \times$ more weights
- Speed up of OT-Flow: between $11 \times$ and $32 \times$ (training) and $4 \times$ and $131 \times$ (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
MFGnet.jl - Julia Package

https://github.com/EmoryMLIP/MFGnet.jl

Coming soon: pyMFGnet for pytorch
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 ≈ 10x

ML/OT: lots of synergies and opportunities

---

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*
in preparation, available soon.
Σ: Machine Learning meets Optimal Transport

Machine Learning $\rightarrow$ 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 $\rightarrow$ Continuous Normalizing Flows

- OT regularization: well-posed simplifies time integration 
- discretize-then-optimize + HJB penalty $\rightarrow$ 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
in preparation, available soon.
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 ≈ 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
in preparation, available soon.
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*
in preparation, available soon.
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 $\approx 10x$

**ML/OT: lots of synergies and opportunities**

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*
in preparation, available soon.
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 $\approx 10x$

**ML/OT: lots of synergies and opportunities**

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*
in preparation, available soon.