

# Machine Learning for Inverse Problems and Data Assimilation: A Practical Session

**Ricardo Baptista**

Statistical Sciences  
University of Toronto



Collaborators: Eviatar Bach, Edo Calvello, Bohan Chen,  
Daniel Sanz-Alonso, Andrew Stuart

CWI Autumn School: Scientific Machine Learning and Numerical Methods

October 27 2025

# Some background first on me

## Trajectory from aviation to probabilistic modeling

- ▶ BSc in aerospace engineering
- ▶ PhD in computational science from MIT
- ▶ Instructor in applied mathematics at Caltech
- ▶ Postdoc at Amazon Stores AI
- ▶ Current: faculty at UToronto

## Current research interests

- ▶ Computational Bayesian inference
- ▶ Data assimilation
- ▶ Measure transport
- ▶ Mathematics of probabilistic machine learning



## 1 Transport for Inverse Problems

- Variational Inference
- Data Amortization

## 2 Learning Filters for Data Assimilation

- Application to 3DVar

Code: [www.github.com/baptistar/MLforIPDA](https://www.github.com/baptistar/MLforIPDA)

# Table of Contents

## 1 Transport for Inverse Problems

- Variational Inference
- Data Amortization

## 2 Learning Filters for Data Assimilation

- Application to 3DVar

**Goal:** Approximate the posterior distribution for parameters  $u$  given data  $y$

$$\pi^y(u) = \frac{1}{Z} \underbrace{I(y|u)}_{Likelihood} \underbrace{\rho(u)}_{Prior}$$

Approximate  $\pi^y(u) \approx q(u)$  by minimizing the KL divergence over a family of distributions  $\mathcal{Q}$ :

$$\min_{q \in \mathcal{Q}} D_{KL}(q || \pi^y)$$

Minimizing the *reverse* KL is equivalent to

$$\arg \min_q D_{KL}(q || \pi^y) = \arg \min_q \mathbb{E}_{u \sim q} [\log q(u) - \log \rho(u) - \log I(y|u)]$$

## Takeaway:

- ▶ Minimization is tractable without knowing  $Z$
- ▶ Requires evaluating  $I(y|\cdot)$ ; alternative is to use  $D_E$  with  $\rho$  samples

# Gaussian Variational Inference

## Gaussian Posterior Approximations:

$$\mathcal{Q} = \{q = \mathcal{N}(m, \Sigma), m \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d \times d}, \Sigma \succ 0\}$$

## Two Gaussian Families:

- ▶ Mean-field:

$$q(u) = \prod_{i=1}^d q_i(u_i), \quad q_i = \mathcal{N}(m_i, \sigma_i^2)$$

- ▶ General multivariate distribution:

$$\Sigma = LL^T, \quad L \in \mathbb{R}^{d \times d}$$

## Optimization Problem:

$$\begin{aligned} & \arg \min_q \mathbb{E}_{u \sim q} [\log q(u) - \log \rho(u) - \log I(y|u)] \\ &= \arg \min_{m, \Sigma} -\frac{1}{2} \log |\Sigma| - \mathbb{E}_{u \sim q} [\log \rho(u) - \log I(y|u)] \end{aligned}$$

## Application: Bio-chemical oxygen demand model

**Parameters:**

$$A \sim U(0.4, 1.2), \text{ and } B \sim U(0.01, 0.31) \quad A \perp\!\!\!\perp B,$$

Transform independent standard Gaussian variables  $u$  to uniform  $(A, B)$  using CDF:

$$\Phi : \mathbb{R} \rightarrow [0, 1]$$

$$A = 0.4 + 0.4\Phi(u_1)$$

$$B = 0.01 + 0.15\Phi(u_2)$$

**Prior:**

$$u \sim \mathcal{N}(0, I_2)$$

**Forward Model:**

$$\mathfrak{B}(t; u) = A(1 - \exp(-Bt))$$

$$G(u) = [\mathfrak{B}(1; u), \mathfrak{B}(2; u), \mathfrak{B}(3; u), \mathfrak{B}(4; u), \mathfrak{B}(5; u)]$$

$$y = G(u) + \eta, \quad \eta \sim \mathcal{N}(0, \Gamma), \quad \Gamma = 10^{-3} I_5$$

**Goal:** Sample  $u \in \mathbb{R}^2$  conditioned on a data measurement  $y \in \mathbb{R}^5$

# Transport Variational Inference

**Transport Approximation:** Choose reference distribution  $\varrho \in \mathcal{P}(\mathbb{R}^d)$

$$\mathcal{Q} = \{q = T(\cdot; \theta)_\# \varrho, \theta \in \mathbb{R}^p\}$$

Sample posterior using:  $T(z; \theta) \sim q, \quad z \sim \varrho$

## Optimization Problem

$$\arg \min_{\theta} -\mathbb{E}_{z \sim \varrho} [\log \rho \circ T(z, \theta) + \log l(y | T(z, \theta)) + \log \det \nabla_z T(z, \theta)]$$

Given i.i.d. samples  $z^{(1)}, \dots, z^{(N)}$  from the reference distribution  $\varrho$ :

$$\arg \min_{\theta} -\frac{1}{N} \sum_{i=1}^N [\log \rho \circ T(z^{(i)}, \theta) + \log l(y | T(z^{(i)}, \theta)) + \log \det \nabla_z T(z^{(i)}, \theta)]$$

**Requirement:**  $T$  must be smooth and monotone:  $\nabla_z T \succ 0$

## Normalizing Flows Define Invertible Transports

**Idea:** Compose  $K$  invertible maps  $L_j$  to create a *flow*

$$T(z; \theta) = L_K \circ L_{K-1} \circ \cdots \circ L_1(z)$$

## Normalizing Flows Define Invertible Transports

**Idea:** Compose  $K$  invertible maps  $L_j$  to create a *flow*

$$T(z; \theta) = L_K \circ L_{K-1} \circ \cdots \circ L_1(z)$$

**Benefit:** The inverse (needed later) and log-determinant are computable layer-wise

$$\begin{aligned} T^{-1}(u; \theta) &= L_1^{-1} \circ L_2^{-1} \circ \dots \circ L_K^{-1}(u) \\ \log \det \nabla T(z; \theta) &= \sum_{j=1}^K \log \det \nabla L_j(y^j), \end{aligned}$$

where  $y^j = L_j \circ \cdots \circ L_1(z)$  are hidden features

# Normalizing Flows Define Invertible Transports

**Idea:** Compose  $K$  invertible maps  $L_j$  to create a *flow*

$$T(z; \theta) = L_K \circ L_{K-1} \circ \cdots \circ L_1(z)$$

**Benefit:** The inverse (needed later) and log-determinant are computable layer-wise

$$\begin{aligned} T^{-1}(u; \theta) &= L_1^{-1} \circ L_2^{-1} \circ \cdots \circ L_K^{-1}(u) \\ \log \det \nabla T(z; \theta) &= \sum_{j=1}^K \log \det \nabla L_j(y^j), \end{aligned}$$

where  $y^j = L_j \circ \cdots \circ L_1(z)$  are hidden features

**Practical remarks:**

- ▶ Each layer is chosen to be a simple transformation
- ▶ There is a trade-off in the expressiveness of each layer and the number of layers  $K$

## How do we parameterize each flow layer?

### Real non-volume preserving flow

Split input  $y \in \mathbb{R}^d$  into two blocks  $y_1$  and  $y_2$  of equal size and apply:

$$L(y) = \begin{bmatrix} y_1 \\ a(y_1) + \exp(b(y_1))y_2 \end{bmatrix}$$

where  $a$  and  $b$  are neural network modeling translation and scale

### Inverse by construction:

$$L^{-1}(w) = \begin{bmatrix} w_1 \\ (w_2 - a(w_1))/\exp(b(w_1)) \end{bmatrix}, \quad \log \det \nabla_y L(y) = b(y_1)$$

### Key Properties:

- ▶ Variable permutations are used between the layers
- ▶ Each layer can only represent conditional Gaussian distributions

**Alternative approaches:** Residual flows, masked autoregressive flows, diffusion models

# Learning Data Dependence of Transports

**Target:** Posterior distribution for any observation  $y$

$$\pi(u|y) \propto \rho(u)l(y|u)$$

**Goal:** Approximate  $\pi(u|y) \approx T(\cdot; y, \theta)_{\#}\varrho(u)$  by minimizing forward KL (flipped order):

$$\arg \min_{\theta} \mathbb{E}_y[D_{KL}(\pi^y || T(\cdot; y, \theta)_{\#}\varrho)]$$

Minimizing the forward KL is equivalent to

$$\arg \min_{\theta} \mathbb{E}_y[D_{KL}(\pi^y || T(\cdot; y, \theta)_{\#}\varrho)] = \arg \min_{\theta} \mathbb{E}_{(u,y)}[-\log T(\cdot; y, \theta)_{\#}\varrho(u)]$$

Note: Objective is likelihood and prior-free!

**Output:** After learning the map, sample any posterior  $\pi(u|y^*)$ :

$$u^{(i)} = \widehat{T}(z^{(i)}; y^*\theta), \quad z^{(i)} \sim \varrho$$

## Maximum Likelihood Training

Given target density  $\gamma(u, y) = \rho(u)\mathbb{I}(y|u)$ :

$$\arg \min_{\theta} \mathbb{E}_{(u,y) \sim \gamma} [-\log T(\cdot; y, \theta) \# \varrho(u)]$$

For standard Gaussian  $\varrho$ :

$$\arg \min_{\theta} \mathbb{E}_{(u,y) \sim \gamma} [\|T(\cdot; y, \theta)\|^2 - \log \det \nabla_u T^{-1}(u; y, \theta)]$$

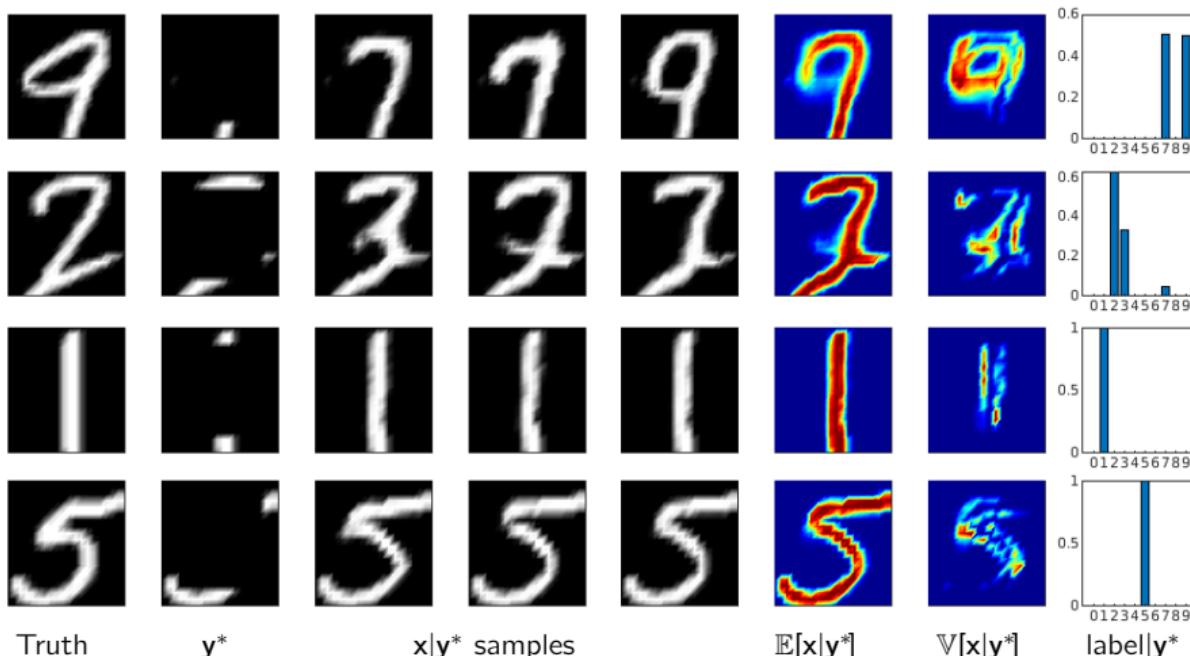
Given i.i.d. samples  $(u^{(1)}, y^{(1)}), \dots, (u^{(N)}, y^{(N)})$  from  $\gamma$ :

$$\arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N -\log \varrho \circ T^{-1}(u^{(i)}; y^{(i)}, \theta) - \log \det \nabla_u T^{-1}(u^{(i)}; y^{(i)}, \theta),$$

**Remark:** Training requires  $T^{-1}$ , but generating samples only needs to evaluate  $T$

## Another Numerical Example: Image In-Painting [B et al., 2024]

- ▶ **Goal:** Reconstruct image after removing its center section
- ▶ Use map to sample from the conditional distribution for the  $14 \times 14$  center pixels of a  $28 \times 28$  MNIST handwritten digit
- ▶ Estimate conditional mean and variance and classify digit probability



# Table of Contents

## 1 Transport for Inverse Problems

- Variational Inference
- Data Amortization

## 2 Learning Filters for Data Assimilation

- Application to 3DVar

## Data Assimilation Recap

**Setup:** Markov model for states and observations

$$\begin{aligned} v_{j+1}^\dagger &= \Psi(v_j^\dagger) + \xi_{j+1}^\dagger, & \xi_{j+1}^\dagger &\sim \mathcal{N}(0, \Sigma) \\ y_{j+1}^\dagger &= h(v_{j+1}^\dagger) + \eta_{j+1}^\dagger, & \eta_{j+1}^\dagger &\sim \mathcal{N}(0, \Gamma) \end{aligned}$$

**Goal:** Approximate the filtering distribution for state at times  $j = 1, \dots, J$ :

$$\pi_j := \mathbb{P}(v_j | y_1^\dagger, \dots, y_j^\dagger)$$

Filtering evolves according to the recursion:

- ▶ Prediction step:  $\hat{\pi}_{j+1} = P\pi_j$ , where  $\hat{\pi}_{j+1} = \mathbb{P}(v_{j+1} | y_1^\dagger, \dots, y_j^\dagger)$
- ▶ Analysis step:  $\pi_{j+1} = A(\hat{\pi}_{j+1}; y_{j+1}^\dagger)$

Note:  $A(\cdot; y) = B(\cdot; y) \circ Q$  in the earlier lectures

# Model Problem: Lorenz-96 System

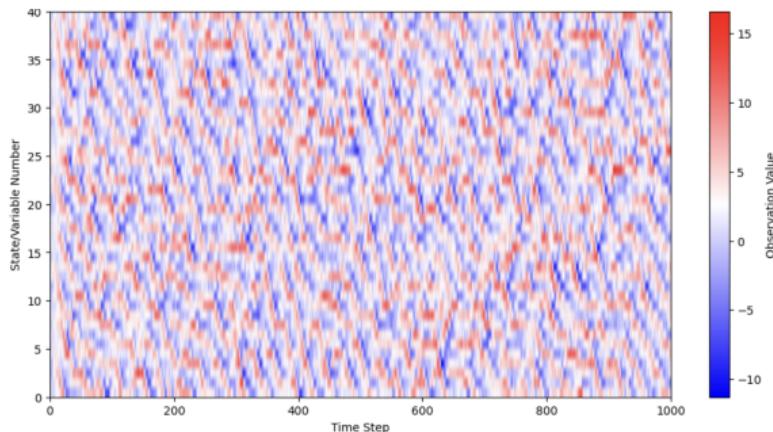
**Continuous-Time System:** Coupled ODE with  $n = 40$  states:

$$\frac{dv_i}{dt} = (v_{i+1} - v_{i-2})v_{i-1} - v_i + F, \quad i = 1, \dots, n$$

starting from initial condition  $v(0) \sim \mathcal{N}(0, I_n)$

**Discretized System:** Discretize ODE using RK4 and evolve state  $v_j$  at discrete time-steps

**Direct Observations:**  $y_j = v_j + \eta_j, \quad \eta_j \sim \mathcal{N}(0, \Gamma)$



# Learning Filters for Data Assimilation

**Goal:** Approximate  $\pi_{j+1} \approx q_{j+1}(\theta)$  using variational inference

Recall:

$$\begin{aligned} & \arg \min_{\theta} D_{KL}(q_{j+1}(\theta) || \pi_{j+1}) \\ &= \arg \min_{\theta} D_{KL}(q_{j+1}(\theta) || \hat{\pi}_{j+1}) - \mathbb{E}_{u \sim q_{j+1}(\theta)} [\log l(y_{j+1}^\dagger; u)] \end{aligned}$$

Approximate  $\pi_{j+1}$  for all steps by minimizing the sum of KL divergences:

$$\min_{\theta} \frac{1}{J} \sum_{j=0}^{J-1} D_{KL}(q_{j+1}(\theta) || \hat{\pi}_{j+1}) - \mathbb{E}_{u \sim q_{j+1}(\theta)} [\log l(y_{j+1}^\dagger; u)]$$

## Approximate Distributions

Consider Gaussian distributions  $q_j$  defined recursively:

- ▶ Prediction step

$$\hat{m}_{j+1} = \Psi(m_j)$$

$$\hat{C}_{j+1} = J_j C_j J_j^T + \Sigma, \quad J_j = \text{Jacobian of } \Psi$$

- ▶ Analysis step

$$m_{j+1} = \hat{m}_{j+1} + K(y_{j+1}^\dagger - Hm_{j+1})$$

$$C_{j+1} = (I - KH)\hat{C}_{j+1}(I - KH)^T + K\Gamma K^T$$

Define:  $\pi_{j+1} = \mathcal{N}(\hat{m}_{j+1}, \hat{C}_{j+1})$  and  $q_{j+1}(K) = \mathcal{N}(m_{j+1}(K), C_{j+1}(K))$

**Goal:** Learn  $K$  that yields close filters for all time  $j$

## 1 Transport for Inverse Problems

- Variational Inference
- Data Amortization

## 2 Learning Filters for Data Assimilation

- Application to 3DVar

Code: [www.github.com/baptistar/MLforIPDA](https://www.github.com/baptistar/MLforIPDA)

**Thank You to attendees and organizers**

astuart@caltech.edu, r.baptista@utoronto.ca