# Machine Learning from a Continuous Viewpoint

Weinan E

Princeton University

Joint work with:

**Chao Ma, Lei Wu**

https://arxiv.org/pdf/1912.12777.pdf

# Examples

Standard ML problems for which we are given the dataset:

1. **Supervised learning**: Given $S = \{(\boldsymbol{x}_j, y_j = f^*(\boldsymbol{x}_j)), j \in [n]\}$, learn $f^*$.
   Objective: Minimize population risk over the "hypothesis space"

$$\mathcal{R}(f) = \mathbb{E}_{\boldsymbol{x}\sim\mu}(f(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2$$

2. **Dimension reduction**: Given $S = \{\boldsymbol{x}_j, j \in [n]\} \subset \mathbb{R}^D$ sampled from $\mu$, find mapping: $\Phi : \mathbb{R}^D \to \mathbb{R}^d, (d \ll D)$ that best preserves all important features of $\mu$.

   "Auto-encoder": Minimize reconstruction error $\boldsymbol{x} - \tilde{\boldsymbol{x}}$: $\tilde{\boldsymbol{x}} = \Psi(\boldsymbol{z}), \boldsymbol{z} = \Phi(\boldsymbol{x})$

$$\mathcal{R}(f) = \mathbb{E}_{\boldsymbol{x}\sim\mu}(\boldsymbol{x} - \Psi(\boldsymbol{z}))^2 = \mathbb{E}_{\boldsymbol{x}\sim\mu}(\boldsymbol{x} - \Psi(\Phi(\boldsymbol{x})))^2$$

Non-standard ML problem, no dataset given beforehand:

1. **Ground state of quantum many-body problem**:
   Let $H = -\frac{\hbar^2}{2m}\Delta + V$ be the Hamiltonian operator of the system

$$I(\phi) = \frac{(\phi, H\phi)}{(\phi, \phi)} = \mathbb{E}_{\boldsymbol{x}\sim\mu_\phi}\frac{\phi(\boldsymbol{x})H\phi(\boldsymbol{x})}{\phi(\boldsymbol{x})^2}, \quad \mu_\phi(d\boldsymbol{x}) = \frac{1}{(\phi,\phi)}\phi^2(\boldsymbol{x})d\boldsymbol{x}$$

   subject to the constraint imposed by Pauli exclusion principle.

2. **Stochastic control problems**:

$$s_{t+1} = s_t + b_t(s_t, a_t) + \xi_{t+1},$$

   $s_t$ = state at time $t$, $a_t$ = control at time $t$, $\xi_t$= i.i.d. noise

$$L(\{a_t\}_{t=0}^{T-1}) = \mathbb{E}_{\{\xi_t\}}\Big\{\sum_{t=0}^{T-1} c_t(s_t, a_t(s_t)) + c_T(s_T)\Big\},$$

   Look for feedback control: $a_t = F(t, s_t), t = 0, 1, \cdots, T-1$.

# Remark: High dimensionality

Benchmark: High dimensional integration

$$I(g) = \int_{X=[0,1]^d} g(\boldsymbol{x}) d\mu, \quad I_m(g) = \frac{1}{m} \sum_j g(\boldsymbol{x}_j)$$

Grid-based quadrature rules:

$$I(g) - I_m(g) \sim \frac{C(g)}{m^{\alpha/d}}$$

Appearance of $1/d$ in the exponent of $m$: **Curse of dimensionality (CoD)**!
If we want $m^{-\alpha/d} = 0.1$, then $m = 10^{d/\alpha} = 10^d$, if $\alpha = 1$.

Monte Carlo: $\{\boldsymbol{x}_j, j \in [m]\}$ is uniformly distributed in $X$.

$$\mathbb{E}(I(g) - I_m(g))^2 = \frac{\mathsf{var}(g)}{m}, \quad \mathsf{var}(g) = \int_X g^2(\boldsymbol{x}) d\boldsymbol{x} - \left( \int_X g(\boldsymbol{x}) d\boldsymbol{x} \right)^2$$

However, $\mathsf{var}(g)$ can be very large in high dimension. Variance reduction!

# Overall strategy:

Formulate a "nice" continuous problem, then discretize to get concrete models/algorithms.

- For PDEs, "nice" = well-posed.
- For calculus of variation problems, "nice" = "convex", lower semi-continuous.

For machine learning, "nice" = variational problem has simple landscape.

# How do we represent a function? An illustrative example

Traditional approach for Fourier transform:

$$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} a(\boldsymbol{\omega}) e^{i(\boldsymbol{\omega},\boldsymbol{x})} d\boldsymbol{\omega}, \quad f_m(\boldsymbol{x}) = \frac{1}{m} \sum_j a(\boldsymbol{\omega}_j) e^{i(\boldsymbol{\omega}_j,\boldsymbol{x})}$$

$\{\boldsymbol{\omega}_j\}$ is a fixed grid, e.g. uniform.

$$\|f - f_m\|_{L^2(X)} \leq C_0 m^{-\alpha/d} \|f\|_{H^\alpha(X)}$$

"New" approach: Let $\pi$ be a probability distribution and

$$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} a(\boldsymbol{\omega}) e^{i(\boldsymbol{\omega},\boldsymbol{x})} \pi(d\boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{\omega}\sim\pi} a(\boldsymbol{\omega}) e^{i(\boldsymbol{\omega},\boldsymbol{x})}$$

Let $\{\boldsymbol{\omega}_j\}$ be an i.i.d. sample of $\pi$, $f_m(\boldsymbol{x}) = \frac{1}{m} \sum_{j=1}^m a(\boldsymbol{\omega}_j) e^{i(\boldsymbol{\omega}_j,\boldsymbol{x})}$,

$$\mathbb{E}|f(\boldsymbol{x}) - f_m(\boldsymbol{x})|^2 = m^{-1} \mathsf{var}(f)$$

$f_m(\boldsymbol{x}) = \frac{1}{m} \sum_{j=1}^m a_j \sigma(\boldsymbol{\omega}_j^T \boldsymbol{x}) =$ two-layer neural network with activation function $\sigma(z) = e^{iz}$.

# Integral transform-based representation

Let $\sigma$ be a scalar nonlinear function (activation function), e.g. $\sigma = \text{ReLU}$

Consider functions represented in the form:

$$\begin{aligned} f(\boldsymbol{x};\theta) = &\int_{\mathbb{R}^d} a(\boldsymbol{w})\sigma(\boldsymbol{w}^T\boldsymbol{x})\pi(d\boldsymbol{w}) \\ =&\mathbb{E}_{\boldsymbol{w}\sim\pi} a(\boldsymbol{w})\sigma(\boldsymbol{w}^T\boldsymbol{x}) \\ =&\mathbb{E}_{(a,\boldsymbol{w})\sim\rho} a\sigma(\boldsymbol{w}^T\boldsymbol{x}) \end{aligned}$$

$\theta =$ parameter to be optimized:

- $\theta = a(\cdot)$ corresponds to a feature-based model
- $\theta = \rho$ corresponds to a two-layer neural network-like model.
- $\theta = (a(\cdot), \pi(\cdot))$, a new model

# Discretize

- Fourier method: $\pi \sim \frac{1}{N}\sum_j \delta_{\boldsymbol{\omega}_j}$ where $\{\boldsymbol{\omega}_j\}$ lives on a uniform lattice. Optimize $a(\cdot)$:

$$f(\boldsymbol{x};\theta) \sim f_m(\boldsymbol{x}) = \frac{1}{m}\sum_j a(\boldsymbol{w}_j)\sigma(\boldsymbol{w}_j^T\boldsymbol{x})$$

- Neural network-based method: $\rho \sim \frac{1}{N}\sum_j \delta_{(a_j,\boldsymbol{\omega}_j)}$ ($\{\boldsymbol{\omega}_j\}$ is also optimized):

$$f(\boldsymbol{x};\theta) \sim f_m(\boldsymbol{x}) = \frac{1}{m}\sum_j a_j\sigma(\boldsymbol{w}_j^T\boldsymbol{x})$$

then optimize (say, using L-BFGS) — this is more in line with traditional numerical analysis (e.g. nonlinear finite element or meshless methods).

# For truly large datasets, we need to use stochastic algorithms

- objective function are all expressed as expectations:

$$\mathcal{R}(\theta) = \mathbb{E}_{\boldsymbol{x}\sim\mu}(f(\boldsymbol{x};\theta) - f^*(\boldsymbol{x}))^2$$

$$\mathcal{R}(\theta_1,\theta_2) = \mathbb{E}_{\boldsymbol{x}\sim\mu}(\boldsymbol{x} - \Psi(\Phi(\boldsymbol{x};\theta_1);\theta_2))^2$$

$$I(\theta) = \mathbb{E}_{\boldsymbol{x}\sim\mu_\theta}\frac{\phi(\boldsymbol{x};\theta)H\phi(\boldsymbol{x};\theta)}{\phi(\boldsymbol{x};\theta)^2}$$

- Gradient descent (GD) can be readily converted to stochastic gradient descent (SGD). Let $F = F(\theta) = \mathbb{E}_{\boldsymbol{x}\sim\mu}g(\theta,\boldsymbol{x})$ be the objective function:

$$\text{GD}: \quad \theta_{k+1} = \theta_k - \eta\nabla_\theta\mathbb{E}_{\boldsymbol{x}\sim\mu}g(\theta_k,\boldsymbol{x})$$

$$\text{SGD}: \quad \theta_{k+1} = \theta_k - \eta\nabla_\theta g(\theta_k,\boldsymbol{x}_k)$$

where $\{\boldsymbol{x}_k\}$ are i.i.d. random samples.

# Optimization: Defining gradient flows

"Free energy" $= \mathcal{R}(f) = \mathbb{E}_{\boldsymbol{x}\sim\mu}(f(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2$

$$f(\boldsymbol{x}) = \int a(\boldsymbol{w})\sigma(\boldsymbol{w}^T\boldsymbol{x})\pi(d\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{w}\sim\pi}a(\boldsymbol{w})\sigma(\boldsymbol{w}^T\boldsymbol{x})$$

Follow Halperin and Hohenberg (1977):

- $a =$ non-conserved, use "model A" dynamics (Allen-Cahn):

$$\frac{\partial a}{\partial t} = -\frac{\delta \mathcal{R}}{\delta a}$$

- $\pi =$ conserved (probability density), use "model B" (Cahn-Hilliard):

$$\frac{\partial \pi}{\partial t} + \nabla \cdot \mathbf{J} = 0$$

$$\mathbf{J} = \pi \boldsymbol{v},\ \boldsymbol{v} = -\nabla V,\ V = \frac{\delta \mathcal{R}}{\delta \pi}.$$

# Gradient flow for the feature-based model

Fix $\pi$, optimize $a$.

$$\partial_t a(\boldsymbol{w}, t) = -\frac{\delta \mathcal{R}}{\delta a}(\boldsymbol{w}, t) = -\int a(\tilde{\boldsymbol{w}}, t) K(\boldsymbol{w}, \tilde{\boldsymbol{w}}) \pi(d\tilde{\boldsymbol{w}}) + \tilde{f}(\boldsymbol{w})$$

$$K(\boldsymbol{w}, \tilde{\boldsymbol{w}}) = \mathbb{E}_{\boldsymbol{x}}[\sigma(\boldsymbol{w}^T \boldsymbol{x})\sigma(\tilde{\boldsymbol{w}}^T \boldsymbol{x})], \quad \tilde{f}(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{x}}[f^*(\boldsymbol{x})\sigma(\boldsymbol{w}^T \boldsymbol{x})]$$

This is an integral equation with a symmetric positive definite kernel.

Decay estimates due to convexity: Let $f^*(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{w}\sim\pi} a^*(\boldsymbol{w})\sigma(\boldsymbol{w}^T \boldsymbol{x})$,

$$I(t) = \frac{1}{2}\|a(\cdot, t) - a^*(\cdot)\|^2 + t(\mathcal{R}(a(t)) - \mathcal{R}(a^*))$$

Then we have

$$\frac{dI}{dt} \leq 0, \quad \mathcal{R}(a(t)) \leq \frac{C_0}{t}$$

# Conservative gradient flow

Optimize $\rho$:

$$f(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{u}\sim\rho}\phi(\boldsymbol{x}, \boldsymbol{u})$$

Example: $\boldsymbol{u} = (a, \boldsymbol{w}), \phi(\boldsymbol{x}, \boldsymbol{u}) = a\sigma(\boldsymbol{w}^T\boldsymbol{x})$

$$\partial_t \rho = \nabla(\rho \nabla V)$$

$$V(\boldsymbol{u}) = \frac{\delta \mathcal{R}}{\delta \rho}(\boldsymbol{u}) = \mathbb{E}_{\boldsymbol{x}}[(f(\boldsymbol{x}) - f^*(\boldsymbol{x}))\phi(\boldsymbol{x}, \boldsymbol{u})] = \int K(\boldsymbol{u}, \tilde{\boldsymbol{u}})\rho(d\tilde{\boldsymbol{u}}) - \tilde{f}(\boldsymbol{u})$$

- This is the mean-field equation derived by Chizat and Bach (2018), Mei, Montanari and Nguyen (2018), Rotskoff and Vanden-Eijnden (2018), Sirignano and Spiliopoulos (2018), by studying the continuum limit of two-layer neural networks.
- Does not satisfy displacement convexity.

# Mixture model

Optimize $(a, \pi)$ ($a =$ non-conservative, $\pi =$conservative):

$$\partial_t a(\boldsymbol{w}, t) = -\frac{\delta \mathcal{R}}{\delta a}(\boldsymbol{w}, t) = -\int a(\tilde{\boldsymbol{w}}, t) K(\boldsymbol{w}, \tilde{\boldsymbol{w}}) \pi(d\tilde{\boldsymbol{w}}, t) + \tilde{f}(\boldsymbol{w})$$

$$\partial_t \pi = \nabla(\pi \nabla \tilde{V}), \quad V(\boldsymbol{w}) = \frac{\delta \mathcal{R}}{\delta \pi}(\boldsymbol{w})$$

# Discretizing the gradient flows

- Discretizing the population risk (into the empirical risk) using data
- Discretizing the gradient flow
  - particle method – the dynamic version of Monte Carlo
  - smoothed particle method – analog of vortex blob method
  - spectral method – very effective in low dimensions

We will see that gradient descent algorithm (GD) for random feature and neural network models are simply the particle method discretization of the gradient flows discussed before.

# Particle method for the feature-based model

$$\partial_t a(\boldsymbol{w}, t) = -\frac{\delta \mathcal{R}}{\delta a}(\boldsymbol{w}) = -\int a(\tilde{\boldsymbol{w}}, t) K(\boldsymbol{w}, \tilde{\boldsymbol{w}}) \pi(d\tilde{\boldsymbol{w}}) + \tilde{f}(\boldsymbol{w})$$

$$\pi(d\boldsymbol{w}) \sim \frac{1}{m} \sum_j \delta_{\boldsymbol{w}_j}, a(\boldsymbol{w}_j, t) \sim a_j(t)$$

Discretized version:

$$\frac{d}{dt} a_j(t) = -\frac{1}{m} \sum_k K(\boldsymbol{w}_j, \boldsymbol{w}_k) a_k(t) + \tilde{f}(\boldsymbol{w}_j)$$

**This is exactly the GD for the random feature model.**

$$f(\boldsymbol{x}) \sim f_m(\boldsymbol{x}) = \frac{1}{m} \sum_j a_j \sigma(\boldsymbol{w}_j^T \boldsymbol{x})$$

# Particle method for the conservative flow

$$\partial_t \rho = \nabla(\rho \nabla V) \tag{1}$$

Particle method discretization:

$$\rho(t, d\boldsymbol{u}) \sim \frac{1}{m} \sum_j \delta_{\boldsymbol{u}_j(t)}$$

Define the loss function

$$I(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m) = \mathcal{R}(f_m), \quad f_m(\boldsymbol{x}) = \frac{1}{m} \sum_j \phi(\boldsymbol{x}, \boldsymbol{u}_j)$$

**Lemma:** Given a set of initial data $\{\boldsymbol{u}_j^0, j \in [m]\}$. The solution of (1) with initial data $\rho(0) = \frac{1}{m} \sum_{j=1}^m \delta_{\boldsymbol{u}_j^0}$ is given by

$$\rho(t) = \frac{1}{m} \sum_{j=1}^m \delta_{\boldsymbol{u}_j(t)}$$

where the particles $\{\boldsymbol{u}_j(\cdot), j \in [m]\}$ solves:

$$\frac{d\boldsymbol{u}_j}{dt} = -\nabla_{\boldsymbol{u}_j} I(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m), \quad \boldsymbol{u}_j(0) = \boldsymbol{u}_j^0, \quad j \in [m]$$

This is exactly the GD dynamics for two-layer neural networks.

# Comparison with conventional NN models

Continuous viewpoint (in this case same as mean-field): $f_m(\boldsymbol{x}) = \frac{1}{m}\sum_j a_j \sigma(\boldsymbol{w}_j^T \boldsymbol{x})$
Conventional NN models: $f_m(\boldsymbol{x}) = \sum_j a_j \sigma(\boldsymbol{w}_j^T \boldsymbol{x})$

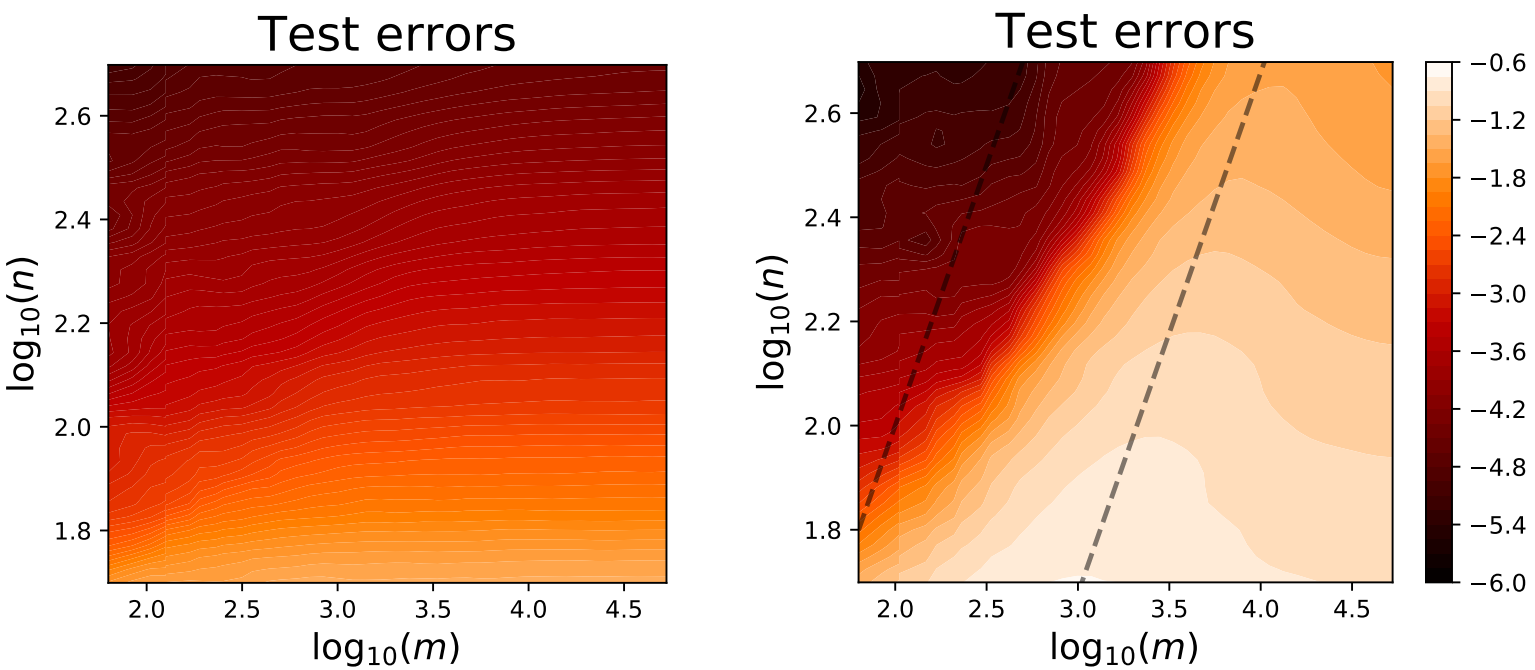



Figure: **(Left)** continuous viewpoint; **(Right)** conventional NN models. Target function is a single neuron $f^*(\boldsymbol{x}) = \sigma(\boldsymbol{e}_1^T \boldsymbol{x})$.

# Flow-based representation

Continuous dynamical system viewpoint (E (2017), Haber and Ruthotto (2017), "Neural ODEs" (Chen et al, 2018))

$$\frac{d\boldsymbol{z}}{d\tau} = \mathbf{g}(\tau, \boldsymbol{z}), \quad \boldsymbol{z}(0) = \boldsymbol{x}$$

The flow-map at time $1$: $\boldsymbol{x} \to \boldsymbol{z}(\boldsymbol{x}, 1)$.

Trial functions:

$$f = \alpha^T \boldsymbol{z}(1)$$

Will take $\alpha = \mathbf{1}$ for simplicity.

# How do we choose the form of $g$ ?

The correct form of $\mathbf{g}$ is given by (E, Ma and Wu, 2019):

$$\mathbf{g}(\tau, \boldsymbol{z}) = \mathbb{E}_{\boldsymbol{w}\sim\pi_\tau} \boldsymbol{a}(\boldsymbol{w}, \tau)\sigma(\boldsymbol{w}^T \boldsymbol{z})$$

where $\{\pi_\tau\}$ is a family of probability distributions.

$$\begin{aligned}\frac{d\boldsymbol{z}}{d\tau} =& \mathbb{E}_{\boldsymbol{w}\sim\pi_\tau} \boldsymbol{a}(\boldsymbol{w}, \tau)\sigma(\boldsymbol{w}^T \boldsymbol{z}) \\ =& \mathbb{E}_{(\boldsymbol{a},\boldsymbol{w})\sim\rho_\tau} \boldsymbol{a}\sigma(\boldsymbol{w}^T \boldsymbol{z})\end{aligned}$$

Discretize: We obtain the "residual neural network" model:

$$\boldsymbol{z}_{l+1} = \boldsymbol{z}_l + \frac{1}{LM}\sum_{j=1}^{M} \boldsymbol{a}_{j,l}\sigma(\boldsymbol{z}_l^T \boldsymbol{w}_{j,l}), l = 1, 2, \cdots, L-1, \quad \boldsymbol{z}_0 = V\tilde{\boldsymbol{x}}$$

$$f_L(\boldsymbol{x}) = \mathbf{1}^T \boldsymbol{z}_L$$

# Compositional law of large numbers

Consider the following compositional scheme:

$$\begin{aligned} \boldsymbol{z}_{0,L}(\boldsymbol{x}) &= \boldsymbol{x}, \\ \boldsymbol{z}_{l+1,L}(\boldsymbol{x}) &= z_{l,L}(\boldsymbol{x}) + \frac{1}{LM}\sum_{k=1}^{M} \boldsymbol{a}_{l,k}\sigma(\boldsymbol{w}_{l,k}^T z_{l,L}(\boldsymbol{x})), \end{aligned}$$

$(\boldsymbol{a}_{l,k}, \boldsymbol{w}_{l,k})$ are pairs of vectors i.i.d. sampled from a distribution $\rho$.

**Theorem (E, Ma and Wu 2019)**

*Assume that*

$$\mathbb{E}_\rho |||\boldsymbol{a}||\boldsymbol{w}^T|||_F^2 < \infty$$

*where for a matrix or vector $\boldsymbol{A}$, $|\boldsymbol{A}|$ means taking element-wise absolute value for $\boldsymbol{A}$. Define $\boldsymbol{z}$ by*

$$\begin{aligned} \boldsymbol{z}(\boldsymbol{x}, 0) &= \boldsymbol{x}, \\ \frac{d}{d\tau}\boldsymbol{z}(\boldsymbol{x}, t) &= \mathbb{E}_{(\boldsymbol{a},\boldsymbol{w})\sim\rho}\boldsymbol{a}\sigma(\boldsymbol{w}^T\boldsymbol{z}(\boldsymbol{x}, \tau)). \end{aligned}$$

*Then we have*

$$\boldsymbol{z}_{L,L}(\boldsymbol{x}) \to \boldsymbol{z}(\boldsymbol{x}, 1)$$

*almost surely as $L \to +\infty$.*

# The optimal control problem

In a slightly more general form

$$\frac{d\boldsymbol{z}}{d\tau} = \mathbb{E}_{\boldsymbol{u}\sim\rho_\tau}\boldsymbol{\phi}(\boldsymbol{z},\boldsymbol{u}), \quad \boldsymbol{z}(\boldsymbol{x},0) = \boldsymbol{x}$$

$\boldsymbol{z}$ =state, $\rho_\tau$ = control at time $\tau$.

The objective : Minimize $\mathcal{R}$ over $\{\rho_\tau\}$

$$\mathcal{R}(\{\rho_\tau\}) = \mathbb{E}_{\boldsymbol{x}\sim\mu}(f(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2 = \int_{\mathbb{R}^d}(f(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2 d\mu$$

where

$$f(\boldsymbol{x}) = \mathbf{1}^T\boldsymbol{z}(\boldsymbol{x},1)$$

# Pontryagin's maximum principle

Define the Hamiltonian $H : \mathbb{R}^d \times \mathbb{R}^d \times \mathcal{P}_2(\Omega) :\mapsto \mathbb{R}$ as

$$H(\boldsymbol{z}, \boldsymbol{p}, \mu) = \mathbb{E}_{\boldsymbol{u}\sim\mu}[\boldsymbol{p}^T \boldsymbol{\phi}(\boldsymbol{z}, \boldsymbol{u})].$$

The solutions of the control problem must satisfy:

$$\rho_\tau = \text{argmax}_\rho \, \mathbb{E}_{\boldsymbol{x}}[H\left(\boldsymbol{z}_\tau^{t,\boldsymbol{x}}, \boldsymbol{p}_\tau^{t,\boldsymbol{x}}, \rho\right)], \;\; \forall \tau \in [0, 1],$$

and for each $\boldsymbol{x}$, $(\boldsymbol{z}_\tau^{t,\boldsymbol{x}}, \boldsymbol{p}_\tau^{t,\boldsymbol{x}})$ are defined by the forward/backward equations:

$$\frac{d\boldsymbol{z}_\tau^{t,\boldsymbol{x}}}{d\tau} = \nabla_{\boldsymbol{p}} H = \mathbb{E}_{\boldsymbol{u}\sim\rho_\tau(\cdot;t)}[\boldsymbol{\phi}(\boldsymbol{z}_\tau^{t,\boldsymbol{x}}, \boldsymbol{u})]$$
$$\frac{d\boldsymbol{p}_\tau^{t,\boldsymbol{x}}}{d\tau} = -\nabla_{\boldsymbol{z}} H = \mathbb{E}_{\boldsymbol{u}\sim\rho_\tau(\cdot;t)}[\nabla_{\boldsymbol{z}}^T \boldsymbol{\phi}(\boldsymbol{z}_\tau^{t,\boldsymbol{x}}, \boldsymbol{u}) \boldsymbol{p}_\tau^{t,\boldsymbol{x}}].$$

$$f(\boldsymbol{x}) = \mathbf{1}^T \boldsymbol{z}(\boldsymbol{x}, 1)$$

with the boundary conditions:

$$\boldsymbol{z}_0^{t,\boldsymbol{x}} = \boldsymbol{x}$$
$$\boldsymbol{p}_1^{t,\boldsymbol{x}} = 2(f(\boldsymbol{x}; \rho(\cdot; t)) - f^*(\boldsymbol{x}))\mathbf{1}.$$

# Gradient flow for flow-based models

Define the Hamiltonian $H : \mathbb{R}^d \times \mathbb{R}^d \times \mathcal{P}_2(\Omega) :\mapsto \mathbb{R}$ as

$$H(\boldsymbol{z}, \boldsymbol{p}, \mu) = \mathbb{E}_{\boldsymbol{u}\sim\mu}[\boldsymbol{p}^T \boldsymbol{\phi}(\boldsymbol{z}, \boldsymbol{u})].$$

The gradient flow for $\{\rho_\tau\}$ is given by

$$\partial_t \rho_\tau(\boldsymbol{u}, t) = \nabla \cdot (\rho_\tau(\boldsymbol{u}, t)\nabla V(\boldsymbol{u}; \rho)), \ \forall \tau \in [0, 1],$$

where

$$V(\boldsymbol{u}; \rho) = \mathbb{E}_{\boldsymbol{x}}[\frac{\delta H}{\delta \rho}\left(\boldsymbol{z}_\tau^{t,\boldsymbol{x}}, \boldsymbol{p}_\tau^{t,\boldsymbol{x}}, \rho_\tau(\cdot; t)\right)],$$

and for each $\boldsymbol{x}$, $(\boldsymbol{z}_\tau^{t,\boldsymbol{x}}, \boldsymbol{p}_\tau^{t,\boldsymbol{x}})$ are defined by the forward/backward equations:

$$\frac{d\boldsymbol{z}_\tau^{t,\boldsymbol{x}}}{d\tau} = \nabla_{\boldsymbol{p}} H = \mathbb{E}_{\boldsymbol{u}\sim\rho_\tau(\cdot;t)}[\boldsymbol{\phi}(\boldsymbol{z}_\tau^{t,\boldsymbol{x}}, \boldsymbol{u})]$$
$$\frac{d\boldsymbol{p}_\tau^{t,\boldsymbol{x}}}{d\tau} = -\nabla_{\boldsymbol{z}} H = \mathbb{E}_{\boldsymbol{u}\sim\rho_\tau(\cdot;t)}[\nabla_{\boldsymbol{z}}^T \boldsymbol{\phi}(\boldsymbol{z}_\tau^{t,\boldsymbol{x}}, \boldsymbol{u})\boldsymbol{p}_\tau^{t,\boldsymbol{x}}].$$

with the boundary conditions:

$$\boldsymbol{z}_0^{t,\boldsymbol{x}} = \boldsymbol{x}$$
$$\boldsymbol{p}_1^{t,\boldsymbol{x}} = 2(f(\boldsymbol{x}; \rho(\cdot; t)) - f^*(\boldsymbol{x}))\mathbf{1}.$$

# Discretize the gradient flow

- forward Euler for the flow in $\tau$ variable, step size $1/L$.
- particle method for the GD dynamics, $M$ samples in each layer

$$\boldsymbol{z}_{l+1}^{t,\boldsymbol{x}} = \boldsymbol{z}_l^{t,\boldsymbol{x}} + \frac{1}{LM}\sum_{j=1}^{M} \boldsymbol{\phi}(\boldsymbol{z}_l^{t,\boldsymbol{x}}, \boldsymbol{u}_l^j(t)), \quad l = 0, \dots, L-1$$

$$\boldsymbol{p}_l^{t,\boldsymbol{x}} = \boldsymbol{p}_{l+1}^{t,\boldsymbol{x}} + \frac{1}{LM}\sum_{j=1}^{M} \nabla_{\boldsymbol{z}}\boldsymbol{\phi}(\boldsymbol{z}_{l+1}^{t,\boldsymbol{x}}, \boldsymbol{u}_{l+1}^j(t))\boldsymbol{p}_{l+1}^{t,\boldsymbol{x}}, \quad l = 0, \dots, L-1$$

$$\frac{d\boldsymbol{u}_l^j(t)}{dt} = -\mathbb{E}_{\boldsymbol{x}}[\nabla_{\boldsymbol{w}}^T \boldsymbol{\phi}(\boldsymbol{z}_l^{t,\boldsymbol{x}}, \boldsymbol{u}_l^j(t))\boldsymbol{p}_l^{t,\boldsymbol{x}}].$$

This recovers the gradient descent algorithm (with back-propagation) for the ResNet:

$$\boldsymbol{z}_{l+1} = \boldsymbol{z}_l + \frac{1}{LM}\sum_{j=1}^{M} \boldsymbol{\phi}(\boldsymbol{z}_l, \boldsymbol{u}_l).$$

# Max principle-based training method

Qianxiao Li, Long Chen, Cheng Tai and Weinan E (2017):

**Basic "method of successive approximation" (MSA)**:

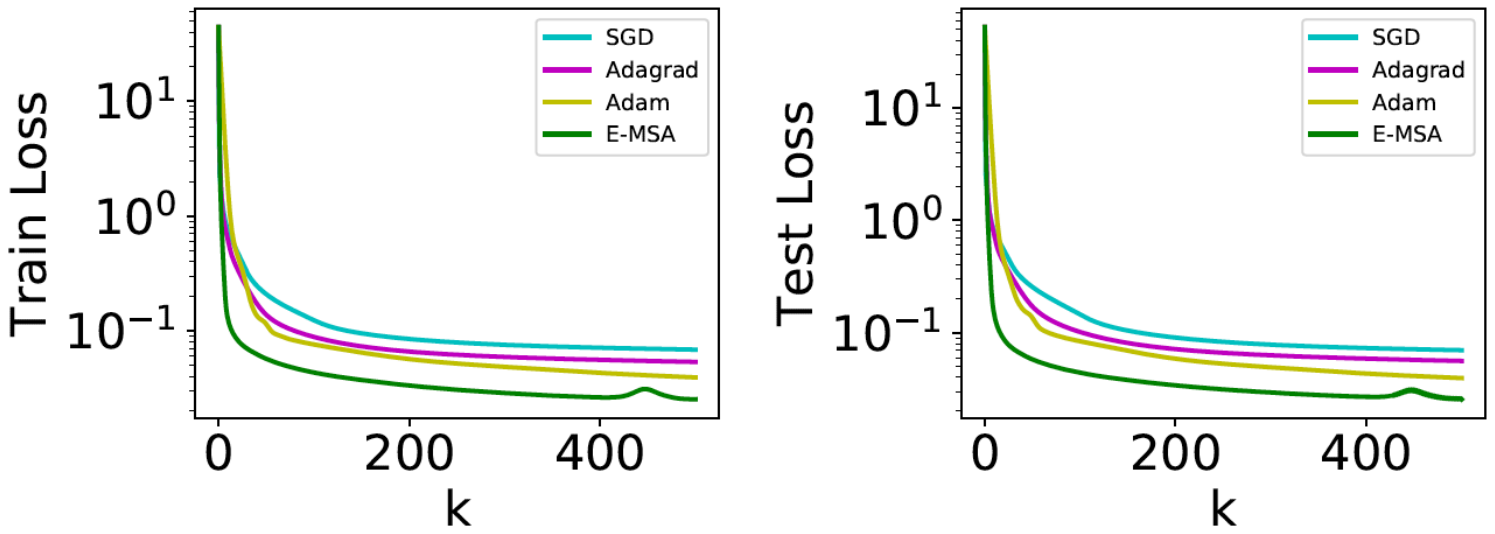
Initialize: $\theta^0 \in \mathcal{U}$

For $k = 0, 1, 2, \cdots$:

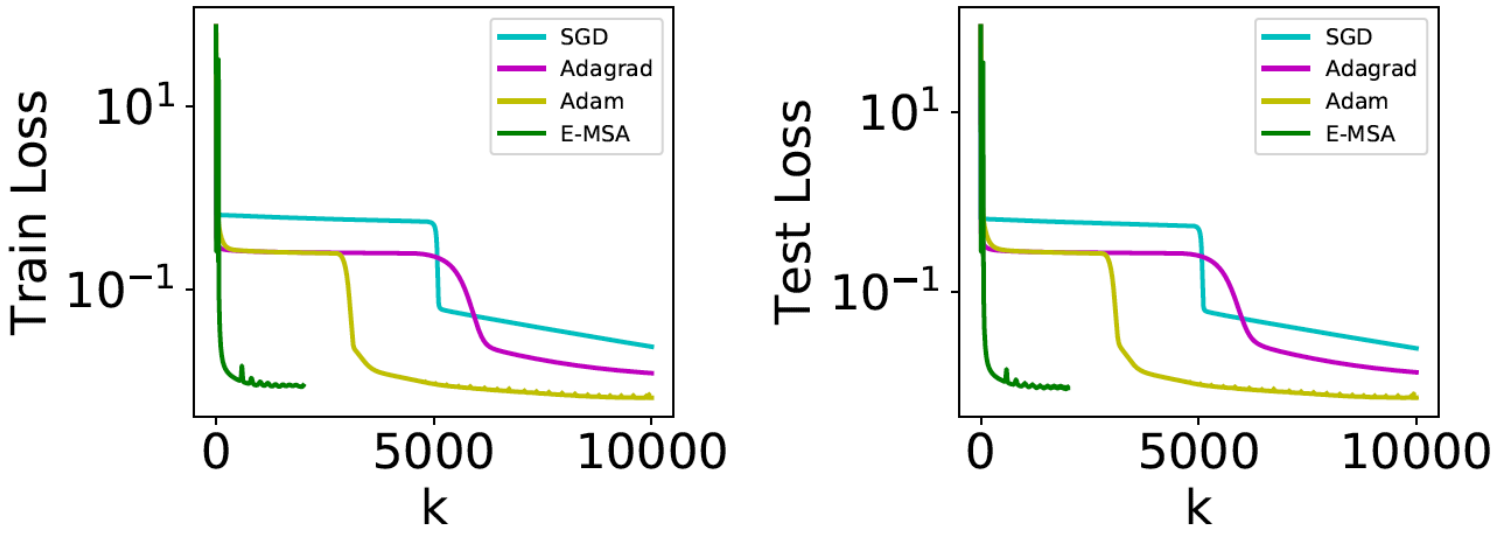
- Solve
$$\frac{d\boldsymbol{z}_\tau^k}{d\tau} = \nabla_{\boldsymbol{p}} H(\boldsymbol{z}_\tau^k, \boldsymbol{p}_\tau^k, \theta_\tau^k), \quad \boldsymbol{z}_0^k = V\boldsymbol{x}$$
- Solve
$$\frac{d\boldsymbol{p}_\tau^k}{d\tau} = -\nabla_{\boldsymbol{z}} H(\boldsymbol{z}_\tau^k, \boldsymbol{p}_\tau^k, \theta_\tau^k), \quad \boldsymbol{p}_1^k = 2(f(\boldsymbol{x}; \theta^k) - f^*(\boldsymbol{x}))\mathbf{1}$$
- Set $\theta_\tau^{k+1} = \text{argmax}_{\theta \in \Theta} H(\boldsymbol{z}_\tau^k, \boldsymbol{p}_\tau^k, \theta)$, for each $\tau \in [0, 1]$

**Extended MSA**:

$$\tilde{H}(\boldsymbol{z}, \boldsymbol{p}, \theta, \boldsymbol{v}, \boldsymbol{q}) := H(\boldsymbol{z}, \boldsymbol{p}, \theta) - \frac{1}{2}\rho\|\boldsymbol{v} - f(\boldsymbol{z}, \theta)\|^2 - \frac{1}{2}\rho\|\boldsymbol{q} + \nabla_{\boldsymbol{z}} H(\boldsymbol{z}, \boldsymbol{p}, \theta)\|^2.$$


Train Loss
$10^1$
$10^0$
$10^{-1}$
0
200
400
k
SGD
Adagrad
Adam
E-MSA
Test Loss
$10^1$
$10^0$
$10^{-1}$
0
200
400
k
SGD
Adagrad
Adam
E-MSA

(a)


Train Loss
$10^1$
$10^{-1}$
0
5000
10000
k
SGD
Adagrad
Adam
E-MSA
Test Loss
$10^1$
$10^{-1}$
0
5000
10000
k
SGD
Adagrad
Adam
E-MSA

(b)

# Comparison between GD and maximum principle

Maximum principle:

$$\rho_\tau = \text{argmax}_\rho \, \mathbb{E}_{\boldsymbol{x}}[H\left(\boldsymbol{z}_\tau^{t,\boldsymbol{x}}, \boldsymbol{p}_\tau^{t,\boldsymbol{x}}, \rho\right)], \ \forall \tau \in [0,1],$$

GD:

$$\partial_t \rho_\tau(\boldsymbol{u}, t) = \nabla \cdot \left( \rho_\tau(\boldsymbol{u}, t) \nabla \mathbb{E}_{\boldsymbol{x}}[\frac{\delta H}{\delta \rho}\left(\boldsymbol{z}_\tau^{t,\boldsymbol{x}}, \boldsymbol{p}_\tau^{t,\boldsymbol{x}}, \rho_\tau(\boldsymbol{u}; t)\right)] \right), \ \forall \tau \in [0,1],$$

Hybrid:

Introducing a different time scale for optimization step: One time forward/backward propagation every $k$ steps of optimization.

- $k = 1$, usual GD or SGD
- $k = \infty$, maximum principle

# "Mean-field" vs. "continuous"

- They sometimes give rise to the same models (e.g. two-layer NNs).

They are DIFFERENT viewpoints.

- mean field: discrete $\to$ continuous by taking the limit (more like interacting particles in stat phys)
- continuous formulation: continuous $\to$ discrete by discretization (more like the usual numerical analysis situation)

"Continuous" formulation tries to formulate the "first principles" of ML.
It allows us to think "outside the box" about ML.

- alternative discretization (spectral)
- alternative model

# Flow-based random feature model

$$\frac{d\boldsymbol{z}}{d\tau} = \mathbb{E}_{\boldsymbol{w}\sim\pi_\tau}\boldsymbol{a}(\boldsymbol{w},\tau)\sigma(\boldsymbol{w}^T\boldsymbol{z})$$

Random feature model: Fix $\{\pi_\tau\}$, optimize over $\{\boldsymbol{a}(\cdot,\tau)\}$.

# Summary

- Continuous formulation tries to formulate the "first principles" of ML.
- It gives rise to new variational and PDE-like problems.
- The performance of these models/algorithms are more stable to the choice of hyper-parameters (no "phase transition").
- It gives rise to new models and new algorithms, more suited to the specific application.

It allows us to think "outside the box" about ML.

The crucial point is the representation of functions as some form of expectations.

- integral transform-based
- flow-based

Other representations?

In a way, neural networks are very natural.