Neural Networks and Partial Differential Equations

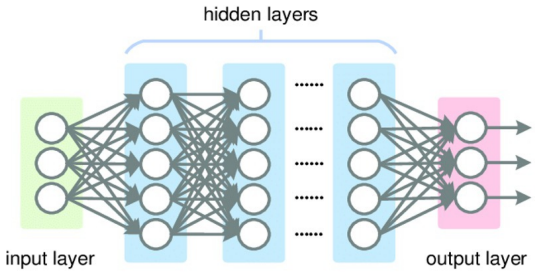
Lexing Ying
Department of Mathematics
Stanford University

FAU DCN-AvH Seminar
Dec 8, 2021

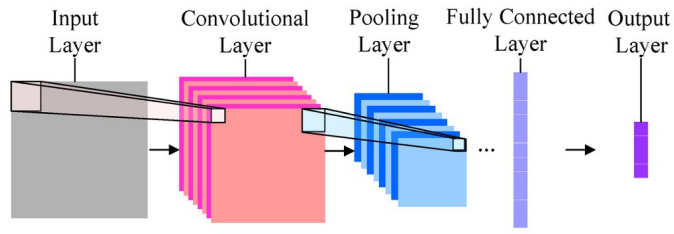
Introduction

Neural networks (NNs)

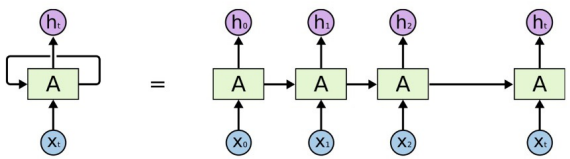
- flexible representations of high-dimensional functions, maps, and distributions



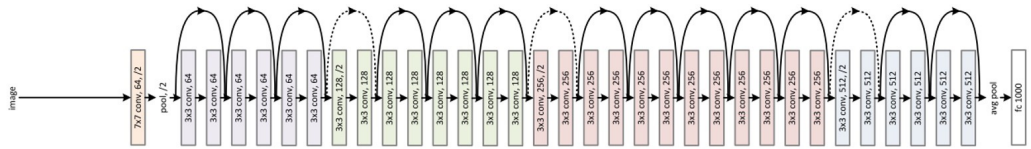
FCN



CNN



RNN



ResNet

Outline

- How NNs help solve PDEs?
- How PDEs help explain and develop NNs?

1. NN for PDEs

Main reason: NNs can represent of high-dimensional functions and maps

- High-dim PDEs
- Low-dim PDEs
- High-dim to low-dim reduction

1.1. NN for high-dim PDEs

Use NN to represent the solution

Consider several examples

- Many-body quantum mechanics (QM)
- Molecular dynamics (MD)
- Control problems

1.1.1 Many-body quantum mechanics

Schroedinger equation: Ground state energy

$$\Phi_* = \mathrm{argmin}_\Phi \frac{\langle \Phi | H | \Phi \rangle}{\langle \Phi | \Phi \rangle}$$

where H is the N-body Hamiltonian and $\Phi(x)$ is an antisym fn on $\mathbb{R}^{3N}$

NN ansatz $\Phi(x) = \Phi_\theta(x)$

$$\theta_* = \mathrm{argmin}_\theta \frac{\langle \Phi_\theta | H | \Phi_\theta \rangle}{\langle \Phi_\theta | \Phi_\theta \rangle}$$

Use for example the variational Monte Carlo (VMC) method

$$\mathrm{argmin}_\theta \frac{\langle \Phi_\theta | H | \Phi_\theta \rangle}{\langle \Phi_\theta | \Phi_\theta \rangle} = \mathrm{argmin}_\theta \frac{\int |\Phi_\theta(x)|^2 \frac{(H\Phi_\theta)(x)}{\Phi_\theta(x)} dx}{\int |\Phi_\theta(x)|^2 dx}$$

NN architecture (for fermionic systems)

- Jastrow factor $\Phi_\theta(x) = \exp([\text{sym}]_\theta) \cdot [\text{anti-sym-Slater-determinant}]_\theta$
- Backflow: θ-parameterized diffeomorphism in x

1.1.2 Molecular dynamics

(Overdamped) Langevin equation

$$dX = -\nabla V(X)dt + \sqrt{\frac{2}{\beta}}dB$$

where $X \in \mathbb{R}^{3N}$: coordinate of a molecular system
Quantities of interests: reaction rate, transition states, transition paths

Example: committor function $u(x) = \mathbb{P}_x(\tau_A < \tau_B)$ for metastable states A, B

$$-\frac{1}{\beta}\Delta u + \nabla V \cdot \nabla u = 0, \quad u|_{\partial A} = 1, \quad u|_{\partial B} = 0.$$

Variational form

$$u_* = \mathrm{argmin}_u \frac{1}{2}\int |\nabla_x u(x)|^2 e^{-\beta V(x)} dx$$

NN ansatz $u(x) = u_\theta(x)$ and optimize with stochastic gradient descent

$$\theta_* = \mathrm{argmin}_\theta \frac{1}{2}\int |\nabla_x u_\theta(x)|^2 e^{-\beta V(x)} dx$$

1.1.3 Control theory

Hamilton-Jacobi-Bellman equation

$$\partial_t v(x,t) + \max_{a \in A} [r(x,a) + (L_a v)(x,t)] = 0$$
$$-\lambda v(x) + \max_{a \in A} [r(x,a) + (L_a v)(x)] = 0$$

where $x \in \mathbb{R}^N$ is the state, a is the action, $v(\cdot)$ is the value fn

- Dynamical programming
- Also called Markov decision process and reinforcement learning
- Linear programming formulations: dual, primal, primal dual (convex/concave minimax problem)

NN ansatz $v(x,t) = v_\theta(x,t)$

- Value network
- Deep Q-network
- Convergence guarantee is hard for most NN algorithms

1.2 NN for low-dim PDEs

Parametric PDEs

- $m(x)$ is a parameter field

$$L_m u = f$$

- Quantity of interest d (depends on the operator L_m)

Example: $x \in \Omega$

- $L_m = -\boldsymbol{\nabla} \cdot (m(x)\boldsymbol{\nabla})$
- $d = \text{DtN}(L_m)$, Dirichlet-to-Neumann map (related to L_m^{-1})

Consider high-dim maps

$$\text{fwd problem} : m \to d, \quad \text{inv problem} : d \to m.$$

Use NN to represent the inv and fwd maps.

High-dim maps

$$\text{fwd problem} : m \to d, \quad \text{inv problem} : d \to m.$$

NN architectures:

- Analytic formulas (e.g. backprojection formula)
  Parameterize the formula with θ
- Unrolling the iterations of an optimization algo
  Parameterize the iterations with θ

Given data $\{(d_i, m_i)\}$, this is a supervised learning problem for θ

A lot of applications (joint work with Yuwei Fan)

- Farfield imaging
- Seismic imaging
- Electrical impedance tomography
- Traveltime tomography
- Optical tomography

1.3 High-dim to low-dim reduction

Model reduction

- Many-body QM to molecular dynamics
- Molecular dynamics to Boltzmann
- Boltzmann to fluid (Euler/Navier-Stokes)

The closure problem: close the reduced model (make it self-contained)

- Find a small number of "moments"
- Find the dynamical equations of the moments

NN ansatz

- Find "moments" using auto-encoder, e.g.

$$f(x,v) \to (\rho(x), u(x), E(x)) \to f(x,v).$$

- Learn the dynamical eqn for $(\rho(x), u(x), E(x))$

$$(\rho_t(x), u_t(x), E_t(x)) \Rightarrow (\rho_{t+dt}(x), u_{t+dt}(x), E_{t+dt}(x))$$

  using short/cheap simulations of $f(x,v)$ and exploration (RL)

2. PDEs for NNs

Outline

- 1. NN architecture
- 2. NN training (supervised learning)
- 3. NN generative modeling (unsupervised learning)
- 4. NN generative adversarial network (minimax)

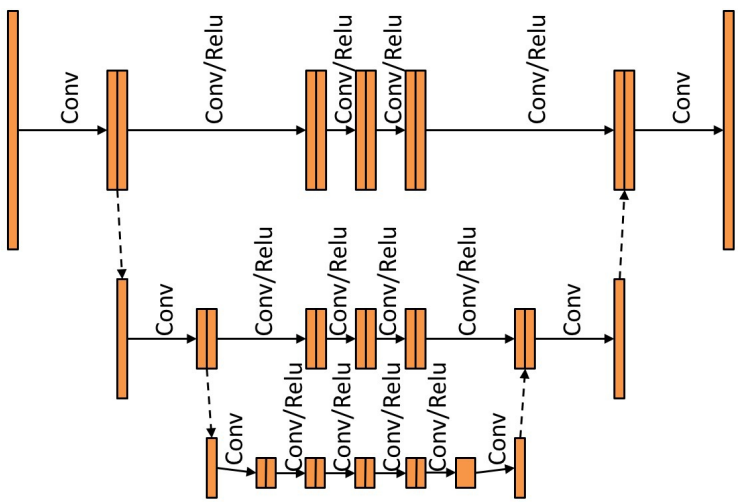
2.1. NN architecture

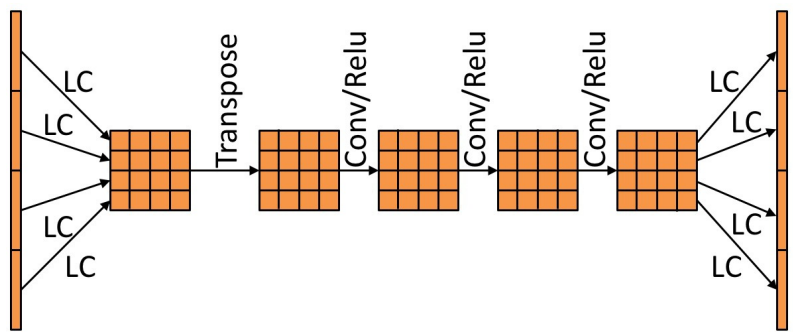
Popular NNs depend on mathematical structures

- Translation invariance: CNN
- ODE and semi-group: ResNet
- Tensor product: Attention mechanism $Y = \sigma(UXV^{\mathsf{T}})$.

Many other PDE operators

- Pseudo-differential ops (multiscale, fast multipole, wavelets): Multiscale networks
- Fourier integral operators for wave propagation: Switch-Nets





2.2. PDE for NN training

- Mean-field approach
- SGD, optimization vs. sampling, ensemble-methods

2.2.1. Mean-field

$x \in \mathbb{R}^d$: input, $y(x)$: response. Two-layer network (NN model fn and loss fn),

$$f_{[\theta_i]}(x) = \frac{1}{m}\sum_{i=1}^{m}\sigma(x\cdot\theta_i), \quad L([\theta_i]) = \int \frac{1}{2}\left|f_{[\theta_i]}(x) - y(x)\right|^2 \mu(x)dx.$$

Lift $\{\theta_i\}$ to a measure $p(\theta) \equiv \frac{1}{m}\sum_i \delta_{\theta_i}(\theta)$. Model and loss functions become

$$f_{[\theta_i]}(x) \Rightarrow \int \sigma(\theta\cdot x)p(\theta)d\theta, \quad L(p) = \int \frac{1}{2}\left|\int \sigma(\theta\cdot x)p(\theta)d\theta - y(x)\right|^2 \mu(x)dx.$$

Mean-field description of GD for $p : \mathbb{R}^d \to \mathbb{R}^+$

$$\partial_t p = \boldsymbol{\nabla}\cdot\left(p\boldsymbol{\nabla}\frac{\delta L}{\delta p}\right)$$

Fokker-Planck-type equation (McKean-Vlasov process) [Chizat-Bach, Mei-Montanari-Nguyen, Rotskoff-Vanden-Eijnden, Sirignano-Spiliopoulos]
Challenges

- This PDE is hard to analyze (the loss not displacement-convex)
- Extension to multilevel networks, other architectures

2.2.2. SGD, optimization vs. sampling, ensemble-methods

Let θ be the vector of all NN parameters, e.g. in a 2-layer NN

$$\theta = [\theta_1, \ldots, \theta_m]^\mathsf{T}$$

SGD for $\min_\theta L(\theta)$ can be modeled for example by

$$\dot{\theta} = -\boldsymbol{\nabla}_\theta L(\theta) + \text{noise}$$

or for $p : \mathbb{R}^{md} \to \mathbb{R}^+$

$$\partial_t p(\theta) = \boldsymbol{\nabla} \cdot (p(\theta) \boldsymbol{\nabla} L(\theta)) + \frac{1}{\beta} \Delta p(\theta).$$

Introduce

$$E(p) \equiv \frac{1}{\beta} \int p(\theta) \ln p(\theta) d\theta + \int L(\theta) p(\theta) d\theta$$

The equation and limiting distribution are

$$\partial_t p = \boldsymbol{\nabla} \cdot \left(p \boldsymbol{\nabla} \frac{\delta E(p)}{\delta p} \right), \quad p_* = \operatorname{argmin}_p E(p) \sim e^{-\beta L(\cdot)}$$

$$\partial_t p = \nabla \cdot \left(p \nabla \frac{\delta E(p)}{\delta p} \right), \quad p_* = \operatorname{argmin}_p E(p) \sim e^{-\beta L(\cdot)}$$

1. Gives a sampling perspective of SGD
   - Convergence rate
   - Flat minimum
   - Generalization error

2. Ensemble methods, e.g. for an SPD convolution operator K

$$\partial_t p = \nabla \cdot \left(pKp \nabla \frac{\delta E(p)}{\delta p} \right)$$

   - This is the Stein variational gradient descent (SVGD) [Liu+Wang], [Lu-Lu-Nolen]
   - Particle realization $\{\theta_1, \ldots, \theta_n\}$

$$\dot{\theta}_i = -\frac{1}{\beta n} \sum_j \nabla K(\theta_i - \theta_j) - \frac{1}{n} \sum_j K(\theta_i - \theta_j) \nabla L(\theta_j)$$

2.3. PDE for generative modeling

Use a push-forward map from a simple to a complex distribution

$$\Phi : \text{Gaussian } z \to x, \quad \gamma(z) \to p(x).$$

NN as an ansatz to represent $\Phi = \Phi_\theta : \gamma(z) \to p_\theta(x)$
Goal: match a target distribution $q(x)$

$$\min_\theta D_{\mathrm{KL}}(p_\theta || p^*) \quad \text{or} \quad \min_\theta D_{\mathrm{KL}}(p^* || p_\theta)$$

A PDE approach [E et al]: potential fn $\phi(x) : X \to \mathbb{R}$

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \boldsymbol{\nabla}\phi(x(t)), \quad t \in [0,1], \quad x(0) = z$$
$$\frac{\mathrm{d}\ln p(x(t))}{\mathrm{d}t} = -\Delta\phi(x(t))), \quad \left(\frac{\mathrm{d}p(x(t))}{\mathrm{d}t} = -p(x(t))\boldsymbol{\nabla}\cdot\boldsymbol{\nabla}\phi(x(t))\right)$$

NN ansatz

- Parameterize scl fn $\phi(x) = \phi_\theta(x)$ instead of map Φ
- Discretizing the ODE gives a ResNet with shared parameter θ
- Automatic differentiation takes care of the derivatives

2.4. PDE for generative adversarial network

$$\min_{\theta} D_{\mathrm{KL}}(p_\theta || p^*) \quad \text{or} \quad \min_{\theta} D_{\mathrm{KL}}(p^* || p_\theta)$$

might not be the right objective for many ML problems (e.g mode collapse)

Replace the objective with an inner maximization

$$\min_{\theta} \left(\max_{w} E_{p^*}(\theta, w) \right)$$

where

- θ: generator network parameter
- w: discriminator network parameter
- A two-player zero-sum game with pure strategies. Hard to analyze

Consider instead the mixed-strategies

$$\min_{p(\theta)} \max_{q(w)} \iint E(\theta, w) p(\theta) q(w) d\theta dw + \frac{1}{\beta} \int p(\theta) \ln p(\theta) d\theta - \frac{1}{\beta} \int q(w) \ln q(w) dw$$

$$\min_{p(\theta)} \max_{q(w)} \iint E(\theta, w) p(\theta) q(w) d\theta dw + \frac{1}{\beta} \int p(\theta) \ln p(\theta) d\theta - \frac{1}{\beta} \int q(w) \ln q(w) dw$$

Gradient ascent/descent dynamics in Wasserstein metric

$$\partial_t p(\theta) = \boldsymbol{\nabla} \cdot \left(p(\theta) \boldsymbol{\nabla} \left(\frac{1}{\beta} \ln p(\theta) + (Eq)(\theta) \right) \right),$$
$$\partial_t q(w) = \boldsymbol{\nabla} \cdot \left(q(w) \boldsymbol{\nabla} \left(\frac{1}{\beta} \ln q(w) - (E^{\mathsf{T}} p)(w) \right) \right).$$

Q: Does this system converge for large β? Hard PDE problem

[Domingo-Enrich et al] uses the birth-death process

$$\partial_t p = \boldsymbol{\nabla} \cdot (p \boldsymbol{\nabla} (+Eq)) - \alpha p (Eq + c),$$
$$\partial_t q = \boldsymbol{\nabla} \cdot \left(q \boldsymbol{\nabla} \left(-E^{\mathsf{T}} p \right) \right) + \alpha q (E^{\mathsf{T}} p + c).$$

An alternative

Consider nested optimization

$$\min_{p(\theta)} \frac{1}{\beta} \int p \ln p d\theta + \left(\max_{q(w)} \iint E(\theta, w) p(\theta) q(w) d\theta dw - \frac{1}{\beta} \int q \ln q dw \right)$$

Quasi-static dynamics (joint work with Chao Ma)

- For each fixed p_t, solve inner max problem using

$$\partial_t q_s = \nabla \cdot \left(q \nabla \left(\frac{1}{\beta} \ln q - E^{\mathsf{T}} p_t \right) \right), \quad q[p_t](w) \propto \exp\Big((E^T p_t)(w) \Big).$$

- Solve the outer min problem (using stationary condition for q

$$\partial_t p_t = \nabla \cdot \left(p_t \nabla \left(\frac{1}{\beta} \ln p + E q[p_t] \right) \right)$$

- Both steps can be implemented using an ensemble dynamics

Thank you

Research supported by NSF