AI in the Sciences and Engineering 2024: Lecture 10

Siddhartha Mishra

Seminar for Applied Mathematics (SAM), D-MATH (and),
ETH AI Center,
ETH Zürich, Switzerland.

What you learnt so far

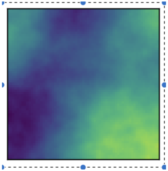
- PINNs to Solve PDEs.
- Great for some PDEs, particularly with low amount of training data.
- Have several negatives.
- We need alternatives !!
- When more data is available:
- The next several lectures: Use of Supervised Deep Learning for PDEs

What does solving a PDE mean ?

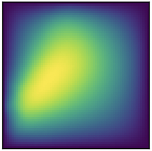
- Example 1: Consider Darcy PDEs:

$$-\mathrm{div}(a\nabla u) = f,$$

- Quantities of interest are:
  - u is temperature or pressure.
  - a is conductance or permeability.
  - f is the source.



a



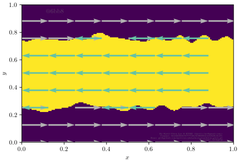
u

- Find the solution Operator $\mathcal{G} : a \mapsto \mathcal{G}a = u$.

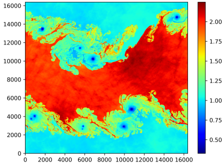
What does solving a PDE mean ?

- Example 2: Consider the Compressible Euler equations:

$$
\begin{aligned}
\rho_t + \mathrm{div}(\rho \mathrm{v}) &= 0, \\
(\rho \mathrm{v})_t + \mathrm{div}\left(\rho \mathrm{v} \otimes \mathrm{v} + p\mathrm{I}\right) &= 0, \\
E_t + \mathrm{div}\left((E + p)\mathrm{v}\right) &= 0., \\
u(x,0) = (\rho, \mathrm{v}, E)(x,0) &= a(x).
\end{aligned}
$$



Initial Condition



Solution at time T

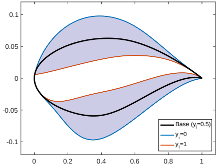
- Find the solution Operator $\mathcal{G} : a \mapsto \mathcal{G}a = u(T)$.

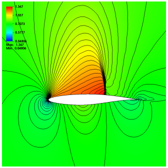
Setup

- X, Y are Banach spaces and $\mu \in \mathrm{Prob}(X)$
- Abstract PDE: $\mathcal{D}_a(u) = f$
- Solution Operator: $\mathcal{G} : X \mapsto Y$ with $\mathcal{G}(a, f) = u$
- Task: Learn Operators from data
- Core of Operator Learning
- A Problem: DNNs map finite dimensional inputs to outputs !!

Solution I: Use Parametric PDEs instead :-)

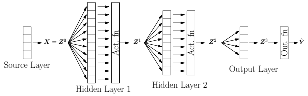
- X, Y are Banach spaces and $\mu \in \mathrm{Prob}(X)$
- Abstract PDE: $\mathcal{D}_a(u) = f$
- Solution Operator: $\mathcal{G} : X \mapsto Y$ with $\mathcal{G}(a, f) = u$
- Simplified Setting: $\dim(\mathrm{Supp}(\mu)) = d_y < \infty$
- Corresponds to Parametrized PDEs with finite parameters.
- Find Soln $u(t, x, y)$ or Observable $\mathcal{L}(y)$ for $y \in Y \subset \mathbb{R}^{d_y}$.





- Approximate Fields or observables with deep neural networks

Supervised learning of target $\mathcal{L}$ with Deep Neural networks



- $\mathcal{L}^*(z) = \sigma_o \odot C_K \odot \sigma \odot C_{K-1} \dots\dots\dots \odot \sigma \odot C_2 \odot \sigma \odot C_1(z)$.
- At the k-th Hidden layer: $z^{k+1} := \sigma(C_k z^k) = \sigma(W_k z^k + B_k)$
- Tuning Parameters: $\theta = \{W_k, B_k\} \in \Theta$,
- σ: scalar Activation function: ReLU, Tanh
- Random Training set: $\mathcal{S} = \{z_i\}_{i=1}^N \in Z$, with i.i.d z_i
- Use SGD (ADAM) to find $\mathcal{L} \approx \mathcal{L}^* = \mathcal{L}^*_{\theta^*}$

$$\theta^* := \arg\min_{\theta \in \Theta} \sum_{i=1}^{N} |\mathcal{L}(z_i) - \mathcal{L}^*_\theta(z_i)|^p,$$

Supervised learning for high-d Parametric PDEs

- Can we find DNN such that $\|\mathcal{L}^* - \mathcal{L}\| \sim \mathcal{O}(\epsilon)$?
- YES: Universal Approximation Property of DNNs $\Rightarrow$:
- Given any Continuous (measurable) $\mathcal{L}$, exists a $\hat{\mathcal{L}}$:

$$\|\mathcal{L} - \hat{\mathcal{L}}\| < \epsilon$$

- If $\mathcal{L} \in W^{s,p}$, $\exists$ DNN $\hat{\mathcal{L}}$ with M parameters (Yarotsky):

$$\|\mathcal{L} - \hat{\mathcal{L}}\|_p \sim \mathcal{O}\left(M^{-\frac{s}{\bar{d}}}\right).$$

- But in Scientific Computing (often):
- $\mathcal{L}$ is not be very Regular and $\bar{d} >> 1$
- If $\mathcal{L} \in W^{1,\infty}$, $\bar{d} = 6$: 1% error, need network of size 10^{12} !!
- Curse of dimensionality: DNN Size $M \sim \epsilon^{-\frac{\bar{d}}{s}}$

Refined Error Estimates

- Error $\mathcal{E} := \|\mathcal{L} - \mathcal{L}^*\|_p$ Decomposition: $\mathcal{E} \leq \mathcal{E}_{app} + \mathcal{E}_{gen} + \mathcal{E}_{opt}$.
- Approximation error $\mathcal{E}_{app} = \|\mathcal{L} - \hat{\mathcal{L}}\|_p$,
  - $\hat{\mathcal{L}}$ is best approximation of $\mathcal{L}$ in $\mathcal{NN}(M)$.
  - One can prove that $\mathcal{E}_{app} \sim \mathcal{O}\left(\bar{d}^{\sigma} M^{-\eta}\right)$ for,
  - Linear Elliptic PDEs: (Schwab, Kutyniok et al).
  - Semi-linear Parabolic PDEs: (E, Jentzen et al).
  - Nonlinear Hyperbolic PDEs: (DeRyck, SM, 2021).
- Optimization Error $\mathcal{E}_{opt} \sim$ Computable Training error:
- Generalization Error

$$\mathcal{E}_{gen}(\theta) := \|\mathcal{L} - \mathcal{L}^*_{\theta}\|_p^p - \frac{1}{N}\sum_i |\mathcal{L}^*_{\theta}(y_i) - \mathcal{L}(y_i)|^p$$

- Using Concentration inequalities + Covering number bounds:

$$\mathcal{E}_{gen} \sim \frac{C\left(M, \log(\|W\|)\right)\log(\sqrt{N})}{\sqrt{N}}$$

Well-trained Networks

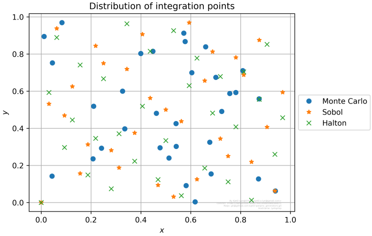
- Assume we can find DNN such that $\mathcal{E}_{app}, \mathcal{E}_T << 1$
- Still Overall Error behaves as

$$\mathcal{E} \sim \frac{C(M)}{N^{\alpha}}, \quad \alpha \leq \frac{1}{2}.$$

- If $C(M) \sim \mathcal{O}(1)$, error of 1% requires 10^4 training samples !!
- Challenge: learn maps of low regularity in a data poor regime
- Contrast with Big Data successes of machine learning.

A Trick: Lye, SM, Ray, 2020

- Use Low discrepancy sequences $\{y_i\}_{i=1}^{N} \in Y$ as Training Set



- These sequences are Equidistributed (better spread out).
- Examples: Sobol, Halton, Owen, Niederreiter ++
- Basis of Quasi-Monte Carlo (QMC) integration.

Training on Low-Discrepancy Sequences

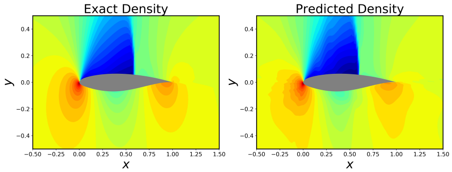
- For $\mathcal{L}$ with Bounded Hardy-Krause variation and smooth σ.
- Generalization Error for Sobol sequences: (SM, Rusch, 2020),

$$\mathcal{E} \leq \mathcal{E}_T + C\left(V_{HK}(\mathcal{L}), V_{HK}(\mathcal{L}^*)\right) \frac{(\log N)^d}{N},$$

Prediction

- Given Hicks-Henne parameter: Predict Drag, Lift, Flow
- DNN with $10^3 - 10^4$ parameters and 128 training samples :

	Run time (1 sample)	Training	Evaluation	Error
Lift	2400 s	700 s	10^{-5} s	0.78%
Drag	2400 s	840 s	10^{-5} s	1.87%
Field	2400 s	1 hr	0.2 s	1.9%

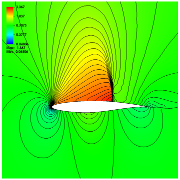


- Errors with Random Training pts: Lift 8.2%, Drag: 23.4%

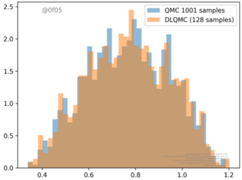
Forward UQ

- DL-UQ algorithm of Lye,SM,Ray, 2020 is

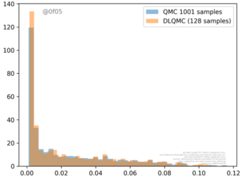
$$\mathcal{L}\#\mu \approx \frac{1}{M}\sum_{i=1}^{M}\delta_{\mathcal{L}(y_i)} \approx \frac{1}{M}\sum_{i=1}^{M}\delta_{\mathcal{L}^*(y_i)}$$



Sample



Lift PDF



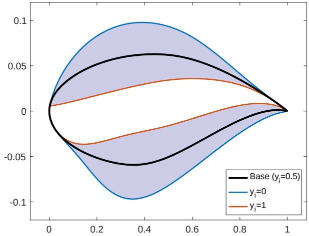
Drag PDF

Observable	Speedup (MC)	Speedup (QMC)
Lift	246.02	6.64
Drag	179.54	8.56

Application: PDE constrained Optimization

- Example: Shape optimization of airfoils
- Parametrize airfoil shape with Hicks-Henne basis functions:

$$S = S_{ref} + \sum_{i=1}^{d} \phi_i, \quad \phi_i = \phi(y_i), \; y = \{y_i\} \in Y \subset \mathbb{R}^{\bar{d}}.$$



- Change airfoil shape to Minimize Drag for constant Lift.

Airfoil shape optimization

- Solve the minimization problem: Find $y^* = \arg\min_{y \in Y} J(y)$,

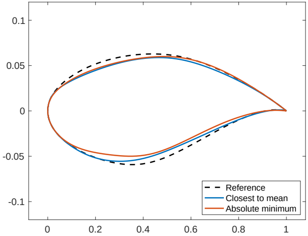
$$J(y) = C_D(y) + P_1(C_L(y) - C_L^{ref}) + P_2(G(y) - G^{ref}(y)).$$

- With penalization parameters $P_1, P_2 >> 1$
- Standard Shape optimization algorithms require Gradients $\nabla_y J(y)$ at each iteration.
- Multiple calls to PDE (and Adjoint) solvers.
- Can be very expensive, even infeasible for optimization under uncertainty.

A DNN based surrogate optimization algorithm

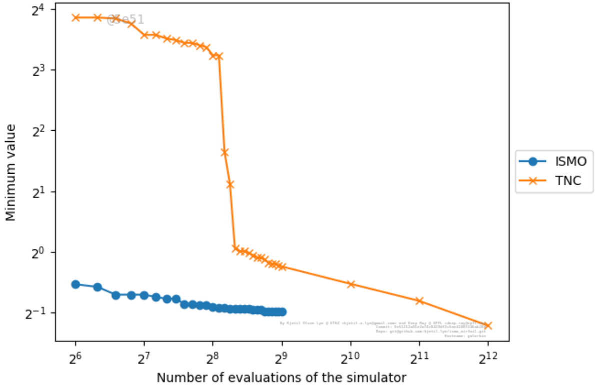
- Choose Training Set $\mathcal{S} \subset Y$ (Random, Sobol,...)
- Train Neural Networks to obtain $C_D^* \approx C_D$, $C_L^* \approx C_L$
- For each step of optimization algorithm:
  - Evaluate objective function as $J^*(y) = J(C_D^*(y), C_L^*(y))$.
  - Evaluate Gradients $\nabla_y J^*$, Hessians $\nabla_y^2 J^*$.
- Run optimization algorithm till minimum is found.
- Significantly faster as DNNs are cheap to evaluate !
- Issue: Training points may not represent Extrema well.
- Addressed in ISMO algorithm of Lye et. al, 2020.

Shape Optimization of airfoils: summary

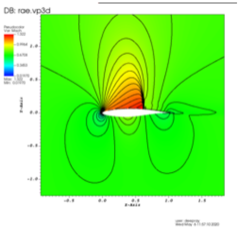


- Reference Drag: 0.0115, Mean optimized Drag: 0.0058
- Reference Lift: 0.876, Mean optimized Lift: 0.887
- Almost 50% Drag reduction on average.

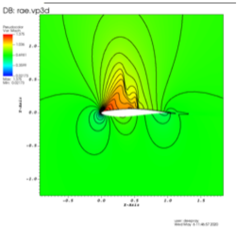
Comparison with Standard Optimizer



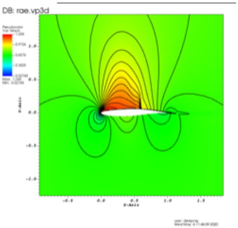
Flow around optimized shapes



Reference



Mean



Best