Princeton Machine Learning Summer School

Clark Miyamoto (cm6627@nyu.edu)

Date

Abstract

Notes from the Princeton Theoretical Machine Learning Summer School

Part I
Jianfeng Lu: Neural Networks for PDEs

In these set of lectures, we'll go over how to design machine learning algorithms to solve problems arising from PDEs.

1 August 12th

Consider a PDE defined on the domain $\Omega \subset \mathbb{R}^d$. For example let's consider the driven heat equation.

$$-\Delta u = f \qquad \text{Inside } \Omega \tag{1.1}$$

$$\partial_n u = 0 \qquad \text{On } \partial\Omega \tag{1.2}$$

where $\Delta = \frac{d}{2} = -\partial^2$ is the laplacian.

We can massage this into a minimization problem, where you attempt to minimize the residual. This is called the **strong form**

$$\inf_u \int_\Omega |\Delta u + f|^2 \tag{1.3}$$

This technique has origins in mathematics long ago, however today, we ansatz $u = u_\theta$ using a neural network, and modify θ s.t. you minimize that quantity. These are known as **PINNs (Physics Informed Neural Networks)**.

Another method is to use the **variational form**

$$\inf_u \frac{1}{2} \int (|\nabla u|^2 - uf)\, dx \tag{1.4}$$

You can convince yourself that this is correct by taking the variational derivative w.r.t. u, and see the minimizers is of the form of the PDE.

Another method is to use the **weak form**, where we expect this

$$\forall v \ \int v(-\Delta u \cdot f) = 0 \iff \int_\Omega \nabla v \cdot \nabla u - vf = 0 \tag{1.5}$$

Problem 1 *To demonstrate your understanding, show the formulation of the strong, variational, and weak form in linear algebra. I.e. you want to solve $Ax = b$, reformulate this into the various forms.*

When realizing these problem on a computers, u is a high-dimensional (or infinite dimensional) quantity, obviously this becomes a problem...

1.1 Finite Dimensional Approximation (In Function Space)

Consider decomposing u into a basis $\{\varphi_i\}_i$

$$u(x) = \sum_{i=1}^{N} c_i \varphi_i(x) \tag{1.6}$$

Now it is your job to find c_i's s.t. you minimize the variational form. So let's plug it in

$$\min_{\{c_i\}} \left[\frac{1}{2} \sum_{ij} \left(c_i c_j \int \nabla \varphi_i \cdot \nabla \varphi_j \right) - \sum_i \left(c_i \int f \varphi_i dx \right) \right] \tag{1.7}$$

This is now just a quadratic minimization problem. Nice.

However, this problem suffers from the curse of dimensionality. The number of coefficients grows exponential in the dimension. To see this, notice when you descrteize the domain Ω, the number of grid points grows as $(\#\text{ points}) = (L/\epsilon)^d$.

So solving PDEs in high-dimension (in domain space) is an interesting problems. And it seems that neural networks can defeat the curse of dimensionality, so perhaps we need to combine the two!

1.2 What can Neural Networks do for PDEs?

So now that we've motived that neural networks can help us in PDEs, let's go over a few applications

1. PINN: Using neural networks ansatz for solving PDEs. We've outlined this problem above.
2. Operator learning: Your use the neural networks to map a function to another function (thus it learns an operator). I.e. this can be a time-evolution kernel. The difficulty lies in encoding an infinite dimensional space into a finite dimensional space.
3. System identification:

1.3 asdf

Let $\mathcal{X}$ be some space over functions. Your true function $u_* \in \mathcal{X}$. Now consider a finite-dimensional approximation of the function space $u_N \in \mathcal{X}_N$, where we assume $\mathcal{X}_N \subset \mathcal{X}$ (this may not hold depending on the boundary condition you set). Note that your true solution u_*, may not necessarily be in $\mathcal{X}_N$. This yields a trivial bound on the residual

$$||u_* - u_N||_x \geq \underbrace{\inf_{u \in \mathcal{X}_n} ||u_* - u||_x}_{\text{Approximation error}} \tag{1.8}$$

Going back to the forced heat equation in the variational form, we want to minimze

$$\Sigma(u) = \frac{1}{2}\int |\nabla u|^2 dx - \int uf dx \inf_{u\in\mathcal{X}} \Sigma(u) = \Sigma(u_*) \tag{1.9}$$

From the bound, we know that $\Sigma(u_N) - \Sigma(u_*) \geq 0$. Plugging this in

$$\Sigma(u_N) - \Sigma(u_*) = \frac{1}{2}\int (|\nabla u_N|^2 - |\nabla u_*|^2) dx + \int (u_N - u_*)\Delta u_* dx \tag{1.10}$$

$$= \frac{1}{2}\int |\nabla u_N - \nabla u_*|^2 dx \tag{1.11}$$

If Ω is bounded, then

$$\underbrace{\int |u_N - u_*|^2 \, dx}_{||u_* - u_N||_x^2} < C \underbrace{\int |\nabla u_N - \nabla u_*|^2 dx}_{\sim(\Sigma(u_N) - \Sigma(u_*))} \tag{1.12}$$

And if you make an additional assumption that Ω is bounded AND convex, then you know that C doesn't depend on $\dim(\Omega)$. This gives us a pretty bound

$$\Sigma(u_N) - \Sigma(u_*) \geq C||u_* - u_N||_x^2 \tag{1.13}$$

You can find another bound

$$\Sigma(u_N) - \Sigma(u_*) = \inf_{u\in\mathcal{X}_N} \Sigma(u) - \Sigma(u_*) \tag{1.14}$$

$$= \inf_{u\in\mathcal{X}_n} \frac{1}{2}\int |\nabla u - \nabla u_*|^2 \tag{1.15}$$

$$\leq \inf_{u\in\mathcal{X}_N} \frac{1}{2}||u - u_*||_x^2 \tag{1.16}$$

Putting this together, you get a very nice principled bound on the variational formulation of the problem.

$$\boxed{C||u_* - u_N||_x^2 \leq \Sigma(u_N) - \Sigma(u_*) \leq \inf_{u\in\mathcal{X}_N} \frac{1}{2}||u_* - u||_x^2} \tag{1.17}$$

Part II
Yury Polyanskiy: Quantization

2 Motivation

Information theory was motivated by attempting to study how much "information" is lost when we move from the continuous time signals to bits. In LLM, you end up just computing matrix multiplication. However keeping the entries of the matrix at full precision (i.e. float32), this is expensive (both in storage, and in transfer time between GPU and CPU), so what happens when we move from float32 $\to$ float4? This is **quantization**.

In a modern context, you have a transformer which converts text to tokens (integers), pictures to soft-tokens (Polyanskiy says this is a vector in $\mathbb{R}^{d_{model}}$).

At the end of these lectures we'll prove a theorem on the optimal compression rate for quantizing the multiplication of two Gaussian iid matrices.

$$\mathbb{E}||\hat{Y} - Y||_F^2$$

where $Y = AB$, and $\hat{Y}$ is the same product where A, B have been quantized.

3 Scalar Quantization (Information Theory)

3.1 Uniform Quantization

Consider a quantization $x \in \mathbb{R}$, and I want to construct a map $x \mapsto \epsilon \in [x/\epsilon]$, where $[n] = [1, ..., n]$. This is called **uniform quantization (UQ)** (in machine learning this is called **round to nearest**). We can then ask, what is the error of UQ?

Consider $x \sim p_x$ (which is $U[-1/2, 1/2]$). This means as you take $\epsilon \to 0$ you find

$$x|[x/\epsilon] = m] \approx \text{Uniform}[-\epsilon/2, \epsilon/2] \tag{3.1}$$

Then you can compute

$$\mathbb{E}[(x - q(x))^2] = \frac{\epsilon^2}{12}(1 + \mathcal{O}(1)) = \frac{2^{-2R}}{12} \tag{3.2}$$

3.2 Dithering

Is there a rigorous way to write $\mathbb{E}[(x - q_\epsilon(x))^2] = \epsilon^2/12$?

To do this, we introduce this cool math trick. Let $U \sim \text{Uniform}[-\epsilon/2, \epsilon/2] \perp X$. Now consider $Y = X + U$. Now quantize Y, which is $q(Y) = q(X + U) = \epsilon[(x + u)/2]$. Now you subtract off the U, so your quantized estimate of X is

$$\hat{X} = q(X + U) - U \tag{3.3}$$

Proposition 1 (Crypto Lemma) $(\hat{X}, X) =^{(d)} (X + \tilde{U}, X)$, *where* $X \perp \tilde{U} \sim$ *Uniform*$[-\epsilon/2, \epsilon, 2]$

Proof: Notice that

$$\hat{X} - X = q(X + U) - (X + U) = \textit{Uniform}[-\epsilon/2, \epsilon/2] \perp X \tag{3.4}$$

Note that this holds for any discretization / lattice scheme.

What's really nice is that this makes any resulting calculation of $f(X)$ very simple. I.e. consider you're wanting to see $Y = X^2$

$$\mathbb{E}[(Y - \hat{Y})^2] = \mathbb{E}[(X^2 - \hat{X}^2)^2] = \mathbb{E}[(X^2 - (X + U)^2)^2] = \frac{\epsilon^2}{3}\mathbb{E}[X^2] + \text{constant} \tag{3.5}$$

3.3 What about non-uniform q?

OpenAI's newest's open source model GPT-OSS uses MX-FP4 quantization. Your quantization domain becomes $\{0, \pm 1/2, \pm 1, \pm 3/2, \pm 2, \pm 4, \pm 6\}$. They got these numbers because it's useful for hardware.

Definition 1 *For a given p_x. Defined the best quantization D_{scalar} as*

$$D_{scalar}(R, P_x) = \inf_{|image\ q| \leq 2^R} \mathbb{E}[|X - q(X)|^2] \tag{3.6}$$

$$= \inf_{p_{x,y}} \{\mathbb{E}[|X - Y|^2] : |\sup p_Y| \leq 2^R\} \tag{3.7}$$

So this is secretly the Wasserstein-2 projection. And it's very non-convex :(

Fix $\text{im}(q) = \{c_1 < ... < c_N\} = G$. The optimal quantization of q is the nearest n-bin $C = \{B_j\}_j$.

Now that you're given the bins $\{B_j\}_{j=1}^N$, the optimal quantization $q(x_= \mathbb{E}[X|X \in B_j\}$.

3.4 What is the best non-uniform quantization?

Theorem 1 (Panter-Diter '51) *Under mild assumptions*

$$D_{scalar}(R; p_x) = \frac{2^{-2R}}{12}\left(\int p_x(x)^{1/2} dxi\right)^3 \tag{3.8}$$

Proof: Let $R \to \infty$ and $N \to \infty$. So this means the number of points in interval $I := N \int_I \lambda(x)dx$ So now you need to ask

$$\mathbb{E}[(x - q(x))^2] = \sum \mathbb{E}[(x - c_j)^2]\mathbb{1} \tag{3.9}$$

Part III
Jianfeng Lu

Part IV
Jianfeng Lu

Part V
Jianfeng Lu