Homework 7
APPM 4720/5720 Scientific Machine Learning, Fall 2024

Due date: Monday, Oct. 14 '24, before midnight, via Gradescope

Instructor: Prof. Becker
Revision date: 10/11/2024

Theme: Baby PINNs

Instructions Collaboration with your fellow students is OK and in fact recommended, although direct copying is not allowed. The internet **is allowed for basic tasks** (e.g., looking up definitions on wikipedia, looking at documentation, looking at basic tutorials) but it is not permissible to search for solutions to the exact problem or to *post* requests for help on forums such as `http://math.stackexchange.com/`.

Setup The simplest differential equation is the ODE for $u : [0, T] \to \mathbb{R}$

$$u'(t) = f(t) \tag{1}$$

for a given function f (more generally, we look at $f(u, t)$ but here we're assuming it's only a function of t). Then via the fundamental theorem of calculus, we know the solution via integration:

$$u(t) = \int_a^t f(s)ds + c \tag{2}$$

for an arbitrary point a in the domain, and some constant c. By specifying an *initial condition* $u(0) = u_0$, we can determine the constant c. We'll work with such a trivial differential equation for this homework in order to introduce the PINN idea and so that it's each to check your answer.

Problem 1: Program a PINN to solve (1) for $f(t) = \sin(\pi t)/(\pi t)$, the normalized Sinc function, on the domain $[0, 10]$, with initial condition $u(0) = 0$. You can check your answer by integrating it exactly (it doesn't have a closed form, but it's so common that it has its own function, "Si", available in `scipy.special`). You can choose any neural net architecture you'd like. $\boxed{\text{Turn in your code}}$ as well as a $\boxed{\text{plot}}$ showing the true solution as well as the PINN solution. Enforce the initial condition by adding it as a penalty to the loss function.

Suggestions: you can use SGD to train, or L-BFGS or similar. The lab 3 solutions for our class show an example of how to use L-BFGS. Your network could be a SIREN or a standard ReLU MLP or whatever you wish. To get the derivatives working properly, I suggest following the setup of the SIREN ipynb notebook where they introduce a `gradient` function, and their neural net returns both the output and a (differentiable copy) of the input "coordinates".

Problem 2: Repeat the above but for the function

$$f(t) = \begin{cases} \frac{\cos(t) - \sin(t)/t}{t} & t \neq 0 \\ 0 & t = 0 \end{cases}$$

which is the derivative of (unnormalized) sinc. Do this on the domain $[0, 20]$ with initial condition $u(0) = 0$. Turn in a $\boxed{\text{plot}}$ as before, and $\boxed{\text{comment on the training}}$: was this easier or harder to train compared to the previous problem?

Problem 3: **Students in 5720 only** Repeat problem 1, but this time enforce the initial condition by parameterizing $u(t) = u_0 + t \cdot \tilde{u}(t)$ where $\tilde{u}$ is the neural network, so that the initial conditions are automatically enforced. Turn in $\boxed{\text{code}}$ and a similar $\boxed{\text{plot}}$, and $\boxed{\text{comment on the training}}$: was it easier or harder to train compared to the first method?