S&DS 689 Homework 2

Due on Thursday, Sept 28, 11:59 pm ET

1. (3 pts) Write TensorFlow/PyTorch/JAX code to learn the oscillatory function

$$f(x) = \begin{cases} 5 + \sum_{k=1}^{4} \sin(kx), & x \in [-\pi, 0) \\ \cos(10x), & x \in [0, \pi] \end{cases}$$

using a shallow ReLU network (i.e., one hidden layer) and 200 uniform data points in $[-\pi, \pi]$ (i.e., training dataset).

Use the networks of the widths in $\{10, 30, 100, 300, 1000\}$, and compute the L^2 relative errors of these networks on a testing dataset of at least 500 uniform data points in $[-\pi, \pi]$. You can use MSE loss, and make sure the network is well trained by choosing a proper learning rate and epochs. Run your code at least 3 times to compute the mean and standard deviation of the errors for each width. Plot the error versus the network width, and discuss what you find.

Hint: (1) Approximation error. (2) The L^2 relative error between the network $\mathcal{N}(x)$ and $f(x)$ is defined as

$$\frac{\|\mathcal{N} - f\|_2}{\|f\|_2},$$

where the L^2-norm $\|\cdot\|_2$ for a function $g(x)$ $(x \in \Omega)$ is defined as

$$\|g\|_2 = \left(\int_\Omega g^2(x) dx \right)^{\frac{1}{2}}.$$

Hence, the L^2 relative error can be approximated by

$$\frac{\|\mathcal{N} - f\|_2}{\|f\|_2} \approx \frac{\sqrt{\sum_{i=1}^{n} \left(\mathcal{N}(x_i) - f(x_i)\right)^2}}{\sqrt{\sum_{i=1}^{n} f^2(x_i)}},$$

where $\{x_i\}_{i=1}^n$ are n uniform points in Ω. n should be large enough to make the approximation accuracy.

2. (1 pts) Repeat Problem (1), but using two hidden layers.

   Compare the new results and the results in Problem (1), and discuss what you find.

   Hint: Shallow networks vs Deep networks.

3. (1 pts) Repeat Problem (1), but using 20 data points in $[-\pi, \pi]$ for training.

   Compare the new results and the results in Problem (1), and discuss what you find.

   Hint: Estimation error.

4. (0 pts) Reading:

(a) A short video on automatic differentiation

(b) (Optional) A long video on the implementation details of automatic differentiation

5. (2 pts) Write a TensorFlow/PyTorch/JAX function to compute the following Gaussian function

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right),$$

where $\sigma = 1$, $\mu = 0$, and $x \in [-5, 5]$. Both the input and output of the Python function should have the size of $(N, 1)$, where N is the batch size.

Use your function to predict $f(x)$ for 10,000 uniform points in $[-5, 5]$, i.e., $N = 10000$. Run your code on CPU, and measure the runtime, e.g., by using `timeit.default_timer()`. Also report the CPU you used, such as Intel Core i9-10900X. (Use the same hardware for all the following problems.) Note:

- Run the code at least 10 times to get the average time.
- Close other software in your computer to measure the time more accurately.

6. (3 pts) Write a TensorFlow/PyTorch/JAX function to compute the 1st-order derivative $f'(x)$.

Use your function to predict $f'(x)$ for 10,000 uniform points in $[-5, 5]$, and measure the runtime. Validate the code by comparing the results with the exact formula.

Hint: Automatic differentiation (AD).

- TensorFlow: You can use tf.GradientTape or tf.gradients, see TensorFlow documentation on AD and advanced AD.
- PyTorch: You can use Tensor.backward or torch.autograd.grad, see PyTorch documentation on AD: [1], [2], and [3]. Also see the usage of `torch.autograd.grad` in Jacobian in the library DeepXDE.
- JAX: You can use jax.grad and jax.vmap, see JAX documentation on AD: [1] and [2].