Physics-Informed Machine Learning 101. Part I

Physics-Informed Machine Learning 101. Part I | Alfonso R. Reyes

Practical Physics-Informed Neural Networks for Engineers

I will be sharing soon via GitHub reproducible examples of PINNs that can be built, run, modified, and explored.

Initially, I will be sharing examples that run under Deep XDE, developed by Prof. George Karniadakis and Prof. Lu Lu. Thank you both so much for building this great Python library, and make the world wiser and machine learning "intelligent".

The amazing thing with DeepXDE is that runs with the most known Machine Learning frameworks: PyTorch and TensorFlow. Powerful JAX is another backend that DeepXDE accepts. The easier path to run PINNs is using independent environments. You may install and run the backends in their isolated environments using uv. For instance, PyTorch in its environment "pytorch-env"; TensorFlow in "tflow-env". This will get you up and running in no time. You may call it "Stage 1".

If you want to compare the results of the backends simultaneously, sort of benchmarking - to see the errors and training times of all them-, you will have to install PyTorch, TensorFlow, and JAX in the same environment, all together. If you want to go this route you will have to abandon the uv comfort zone, and create the multi-framework with the environment manager mamba. Basically is a modern "conda". The easier way to do this is by using the CPU version of the backends. We'll call this "Stage 2".

"Stage 3", will be commingling all the backends above but their corresponding GPU version. This is the hardest of the three. I have been successful but took me considerable time to get the dependencies and the CUDA version right, so they do not conflict with each other.

The example below, corresponds to the solution of the differential equation for growth and decay. It is the essential equation to get introduced to Computational Physics and SciML, along with free fall, projectile launch, and single harmonic oscillator; and of course, PINNs, which is covered in both disciplines. These are the parameters used to solve it:

Test Parameters:

Problem: dy/dx = y, y(0) = 1
Domain: [0, 2]
Boundary points: 16

Domain points: 100
Network: [1, 20, 20, 20, 1] (3 hidden layers)
Activation: tanh
Optimizer: Adam (lr=0.001)
Iterations: 1000
Success threshold: errors < 0.1

In this case, as you see in the plots, we know the analytical solution, which we use to compare the backends: PyTorch, TensorFlow, and JAX. Mind you -and this is my opinion-, that the important and harder thing in PINNs is not the neural network, or ML algorithm; it is the understanding of the physics and the construction or discovery of the differential equation.

Remember, Physics first: understand how the laws of nature work through Differential Equations; data is just an expression of nature with no constraints.

hashtags:: SciML PIML PINN Petroleum Engineering SPE Data Science notAI dontcallitAI Neural Networks