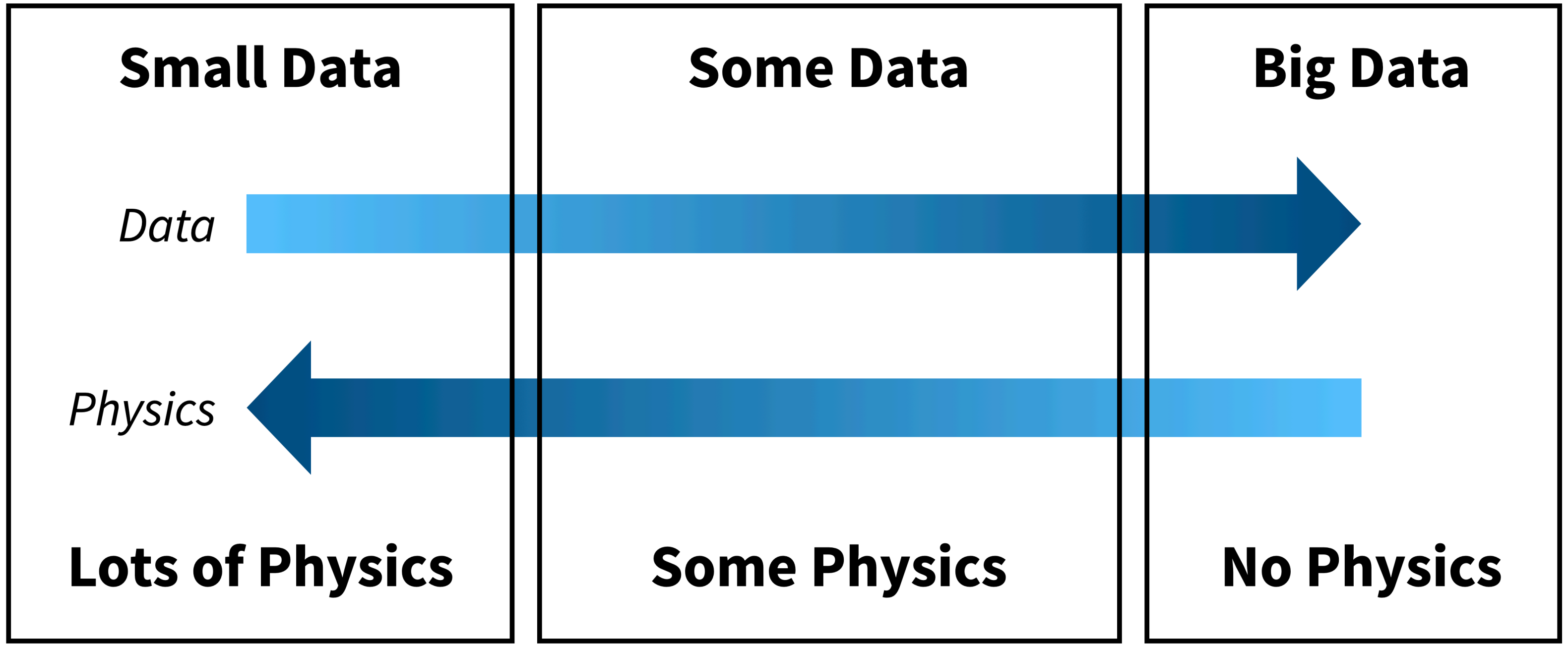
CME 216, ME 343 - Winter 2021

Eric Darve, ICME

Stanford University

Physics-informed machine learning



Physics-informed learning leverages:

- Data from experiments and/or high-fidelity computer simulations
- Physics knowledge in the form of constraints

Examples of constraints

Equality:

$$F = ma$$

$$\rho(t) = \rho(0)$$

$$E(t) = E(0)$$

Differential equations:

$$-\nabla(k \cdot \nabla u) = f$$

$$\nabla \times E = -\frac{\partial B}{\partial t}$$

$$-\nabla(k \cdot \nabla u) = f$$

Assume we are given k and f and want to compute u.

We may be given some data: $\{u(x_i)\}_i$.

Conventional ML: DNN model $u(x;\theta)$

$$\mathrm{Loss} = \sum_{i=1}^{n_{\mathrm{obs}}} \|u(x_i;\theta) - u_i\|_2^2$$

How can we leverage our PDE?

$$-\nabla(\boldsymbol{k} \cdot \nabla u) = f$$

Add a penalty term:

$$\text{Loss} = \sum_{i=1}^{n_{\text{obs}}} (u(x_i;\theta) - u_i)^2 + \lambda \sum_{j=1}^{n_{\text{phys}}} \left[f(x_j) + \nabla(k \cdot \nabla u(x_j;\theta)) \right]^2$$

A simple idea but with some interesting consequences.

If you have limited observation data u_i, the PDE can be used to impose additional constraint on the model $u(x;\theta)$.

This leads to more robust training and more accurate DNN models.

The model can easily incorporate data measured at irregular locations (x) or times (t).

Initial conditions and boundary conditions are less relevant.

With this method you can find approximate solutions of

$$-\nabla(k \cdot \nabla u) = f$$

if you are given enough observations $u_i = u(x_i)$ even **without boundary conditions.**

This is much harder to do with a traditional scheme like finite-difference where boundary conditions are expected.

How would we solve

$$-\nabla(k \cdot \nabla u) = f$$

using a convention numerical solver?

In general, numerical solvers rely on a grid or a discretization of the domain using a mesh.

Take for example:

$$-\frac{d^2u}{dx^2} = f(x), \quad u(0) = u_0, \quad u(1) = u_1$$

Approximate the 2nd order derivative using the finite-difference scheme:

$$-\frac{d^2u}{dx^2} \approx \frac{2u_i - u_{i+1} - u_{i-1}}{h^2}$$

u_i is an approximation of u at $x_i = ih$.

h is the grid size.

Then, given f_i, solve for u_i

$$\frac{2u_i - u_{i+1} - u_{i-1}}{h^2} = f_i$$

This is a linear system.

PhysML uses a different approach.

It relies on the fact the DNNs can be easily differentiated.

We will explore this idea in the next lecture video.