Module 5: Learning parameters of mathematical models from data

John Lowengrub (Math, BME)
Jack Corrette (MCSB)
Caleb Hendrick (MCSB, Math)
Zirui (Ray) Zhang (UCI/WPI)

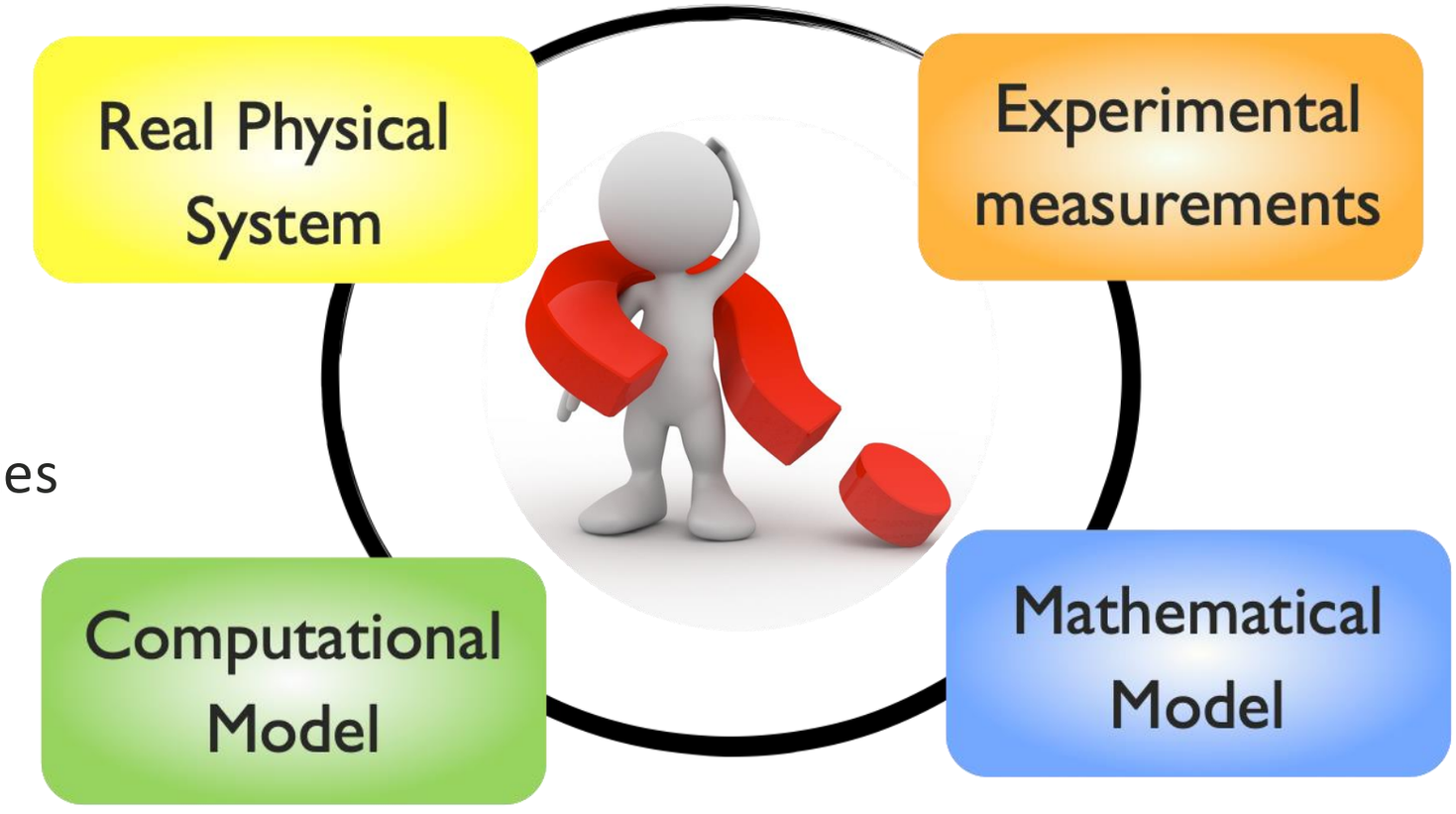
Google gemini

Mathematical and computational modeling provides a bridge between experiments and the real physical system

- **Mathematical**
  - Model of real system, typically simplified
  - lack of knowledge about real system (uncertainty about model)
- **Experimental data**
  - measurements
  - Experimental errors, noise (uncertainty in measurements)
- **Parametric**
  - Need to determine parameter values
  - limited knowledge of model parameter values (uncertainty in parameter values)
- **Computational**
  - numerical approximations,

(e.g. discretization over finite number of grid points)

  - rounding errors (uncertainty about numerical accuracy)



A model is only as good as its parameters

3 main types of parameter estimation from data:

- Maximum likelihood estimation (MLE)

  - Point estimates for parameters that are the best in explaining the data

- Maximum a posteriori (MAP)

  - Point estimates for parameters that are the best in explaining the data taking into account prior knowledge. (e.g., parameter ranges)

- Bayesian parameter inference

  - Estimate probability distribution of parameters, given the data.

Outline of module 5

Focus on Maximum likelihood estimation (MLE)

- Brief intro to MLE

- Application of MLE to determine parameters of logistic growth model (demonstration in Matlab)

- Neural network approaches. (Physics-Informed Neural Networks)

  -Test algorithms
  -Vary network parameters
  -Inference using synthetic data
  -Inference using bacteria growth data

Maximum likelihood estimation (MLE) I

- Let $p(D|\rho)$ denote the probability that the data D can be described by a model with parameters ρ. Then, we want to choose ρ to *maximize* $p(D|\rho)$.

- Typically, this is done by minimizing $-\log(p(D|\rho))$. That is,

$$\rho_{MLE} = \text{argmin}_{\rho}\left(-\log(p(D|\rho))\right)$$

This is usually motivated by assuming that D contains a set of independent observations, e.g., $p(D|\rho) = \Pi_{i=1}^{n} p(D_i|\rho)$

- If one defines a cost function by

$$c(D|\rho) = -\log(p(D|\rho))$$

Then, we can write

$$p(D|\rho) = e^{c(D,\rho)}$$

Maximum likelihood estimation (MLE) II

Let's see how to apply this to a typical problem.

Consider the model of logistic growth:

$$\frac{dx}{dt} = \lambda\, x(t)(1 - (x(t)/\theta)^{\alpha})\,, \quad x(0) = p_0$$

where

- $x(t)$ is the population at time t,
- λ is the intrinsic *reproduction rate*,
- θ is the *carrying capacity*, and
- α is an *impedance* (shape) parameter

Maximum likelihood estimation (MLE) III

Given the parameters $(\lambda, \theta, \alpha, p_0)$

we can write the solution as: $x(t; \lambda, \theta, \alpha, p_0)$

and take the cost function to be the sum of squared errors:

$$c(\lambda, \theta, \alpha, p_0) = SSE(\lambda, \theta, \alpha, p_0) = \sum_i (x(t_i, \lambda, \theta, \alpha, p_0) - x_i)^2$$

where x_i are the observations at t_i

Thus, we want to solve the inverse problem:

$$(\lambda^*, \theta^*, \alpha^*, p_0^*) = \arg\min_{(\lambda,\theta,\alpha,\mathrm{p_0})} \mathrm{SSE}(\lambda, \theta, \alpha, \mathrm{p_0})$$

Maximum likelihood estimation (MLE) IV

In classical optimization theory, generate a sequence:

$$\boldsymbol{\rho}_n = (\lambda_n, \theta_n, \alpha_n, p_{0,n}) \xrightarrow{n \to \infty} \boldsymbol{\rho}^* = (\lambda^*, \theta^*, \alpha^*, p_0^*)$$

The classic approach to generate the sequence is gradient descent:

$$\boldsymbol{\rho}_{n+1} = \boldsymbol{\rho}_n - \delta \boldsymbol{\nabla}_{\boldsymbol{\rho}} SSE(\rho_n)$$

Here, $\boldsymbol{\nabla}_{\boldsymbol{\rho}}$ denotes the gradient with respect to the parameters.

Note that this requires, $\nabla_{\boldsymbol{\rho}} x(t, \boldsymbol{\rho})$ which must be calculated.

An alternative, is to use the *Nelder-Mead algorithm* (fminsearch), which does not require derivatives.

MATLAB

Maximum likelihood estimation (MLE) V

We will demonstrate how to fit the logistic growth model to synthetic noisy data. The codes can be found in `https://github.com/CalebHend/MCSB-Bootcamp-Data-Fitting/tree/main/fminsearch-MATLAB-Demo`. Note that this approach still requires the solution of Eq. (1) to be computed at each stage of the iteration.

Physically-Informed Neural Networks I

- Introduction to Neural Network
- How to Train a Neural Network
- What is a Physics-Informed Neural Network (PINN)
- Solving an Inverse Problem using PINNs

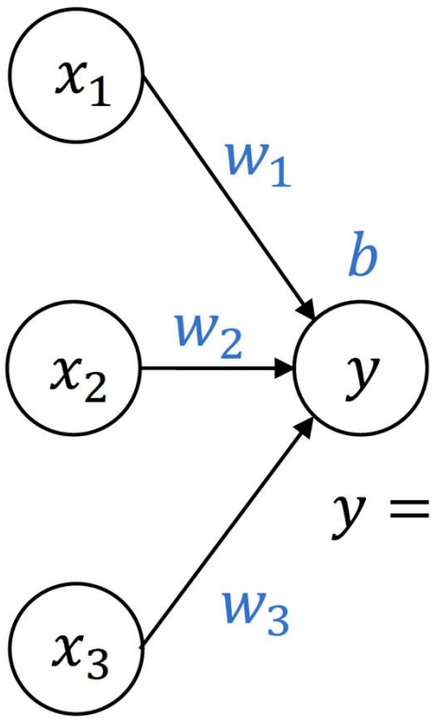
Advantages:

-Solve the forward and backward problems simultaneously

-Use sophisticated algorithms previously developed for neural networks.

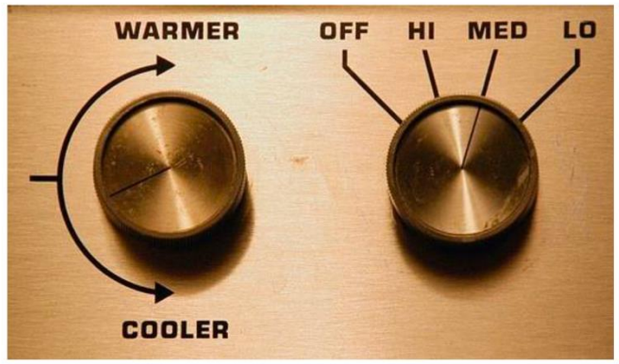
-Very powerful, e.g., can infer unknown functions with “little” imposed structure,…

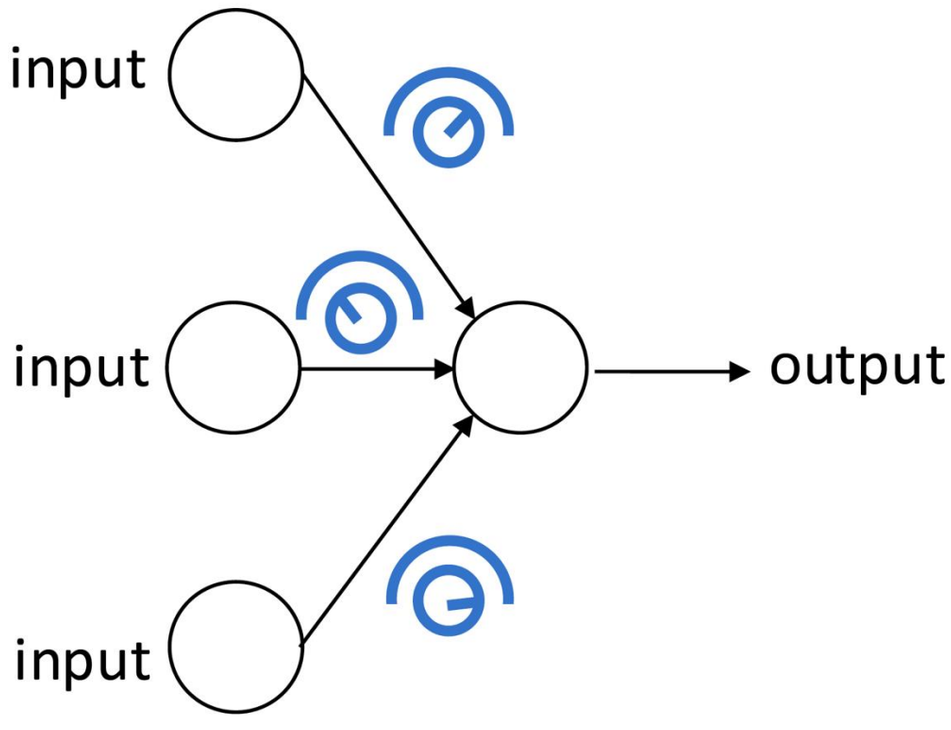
Physically-Informed Neural Networks II: Neuron



$$y = \sigma(w_1 x_1 + w_2 x_2 + w_3 x_2 + b)$$

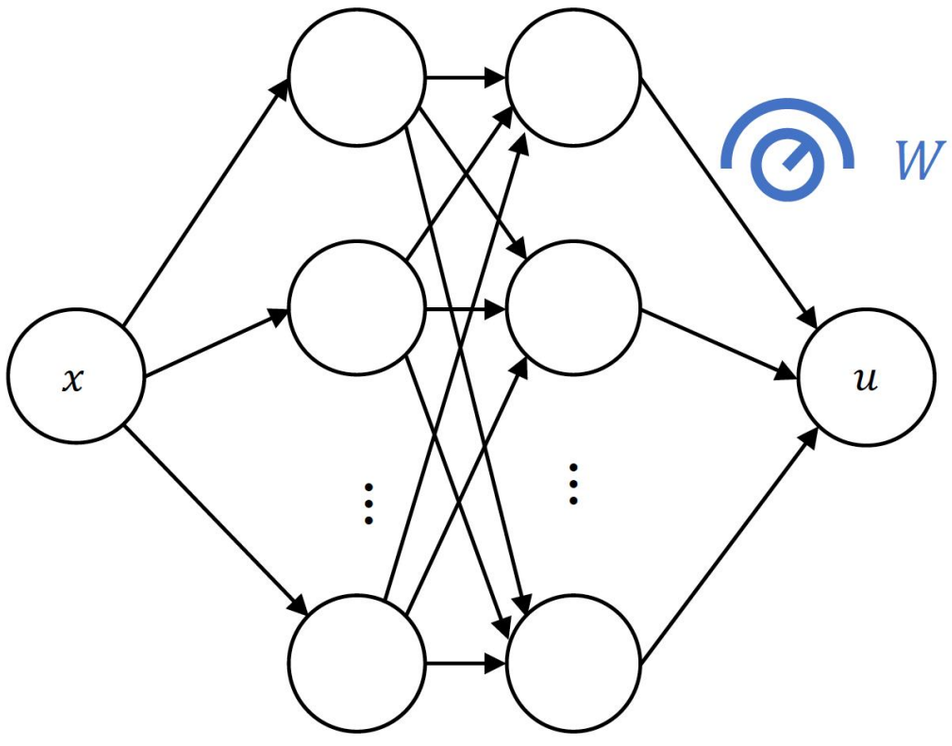
Example activation $\sigma(z) = \tanh(z)$

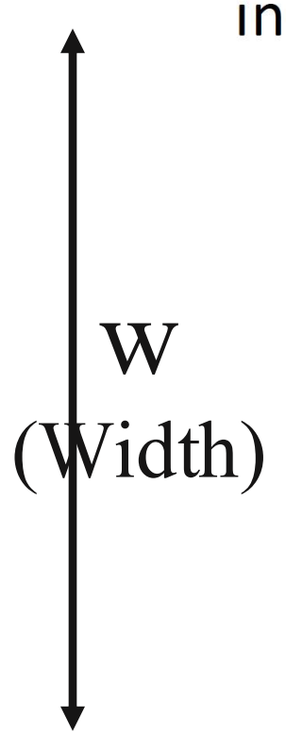


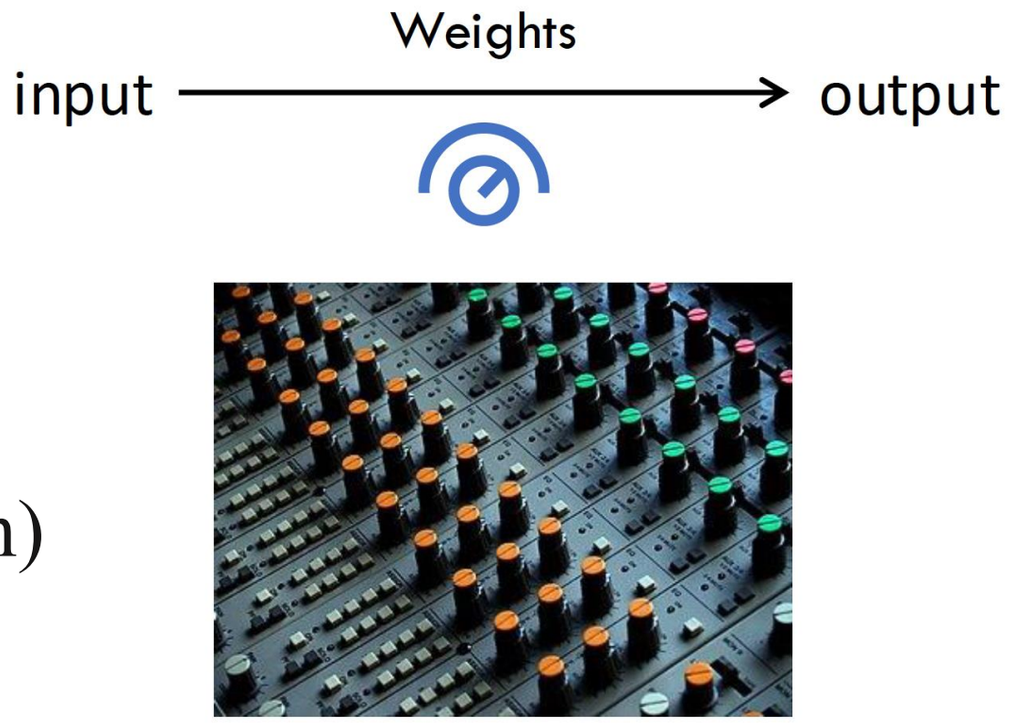


Physically-Informed Neural Networks II: Neural network

Neural network is a function
that depends on the weights $u_W(x)$







Physically-Informed Neural Networks III: How to train a Neural network

Given data $(t_1, y_1), \ldots (t_N, y_N)$,

$$\min_{W} \qquad L_{data} = \sum_{i=1}^{N} (u_W(t_i) - y_i)^2$$

min
weights

error between the neural network prediction and data

Turn the “knobs” to decrease the loss

Google Colab

Physically-Informed Neural Networks IV: Google Colab

The necessary files can be found at:

`https://drive.google.com/drive/folders/10YSAciWFgAvxpZKgGNzzt9tpcDoNeF4W?usp=sharing`

Have one member of your group **make a copy** of the folder in their Drive and **share and edit the copy** with the group. Work only in the copied folder.

Google Colab

References

- Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.

- Zhang et al., Medical Image Analysis, 103423 (2025)

- Zhang et al., BiLO: Bilevel Local Operator Learning for PDE inverse problems. Part 1: PDE constrained optimization. Arxiv.2404.17789v4 (2025). J. Comput. Phys. in review.

- Zhang et al., BiLO: Bilevel Local Operator Learning for PDE inverse problems. Part 2: Efficient uncertainty quantification with low-rank adaptation. ArXiv (2025). J. SCI. Comput. in review.