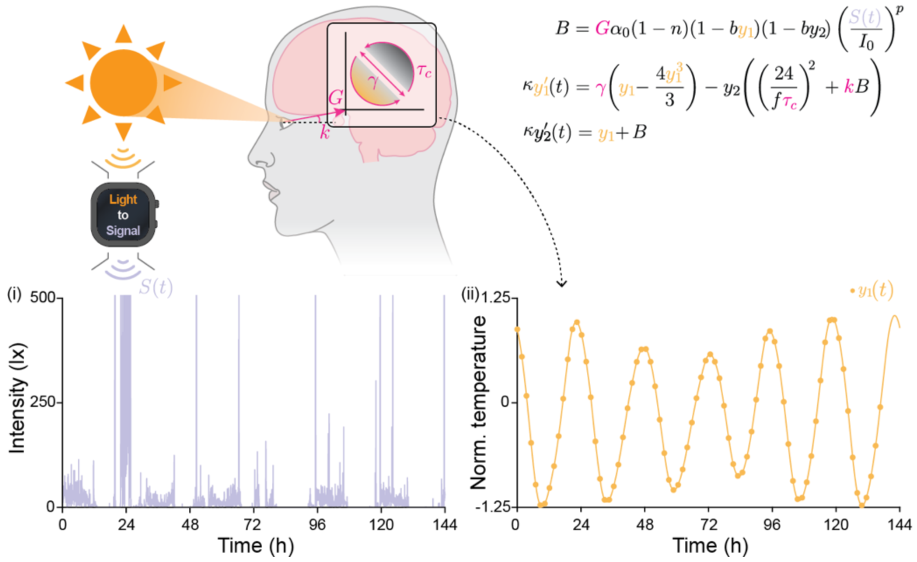
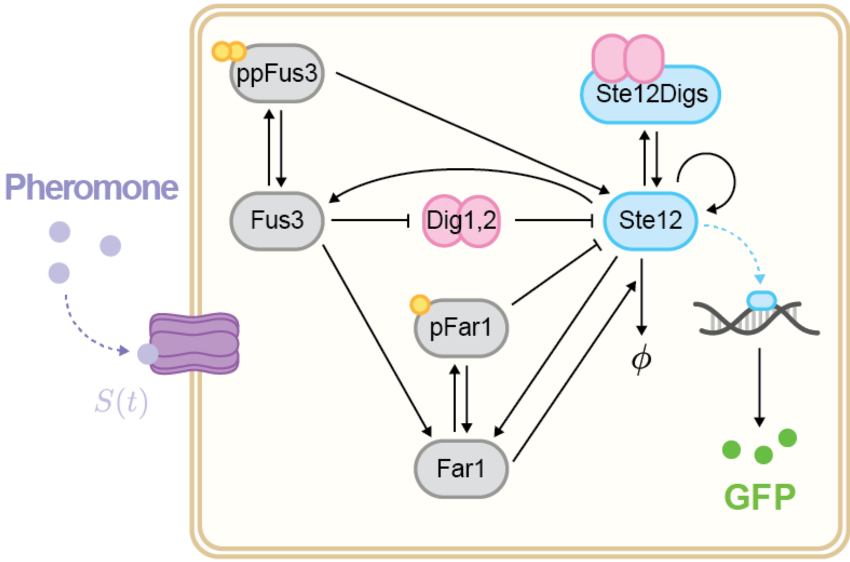
CoMaDS 2025 International Workshop on Computational and Mathematical Methods in Data Science

Physics-Informed Neural Networks (PINNs)

Fitting a Mathematical Model to Real Data





Korea U. Sejong campus

Division of applied math

Hyeontae Jo

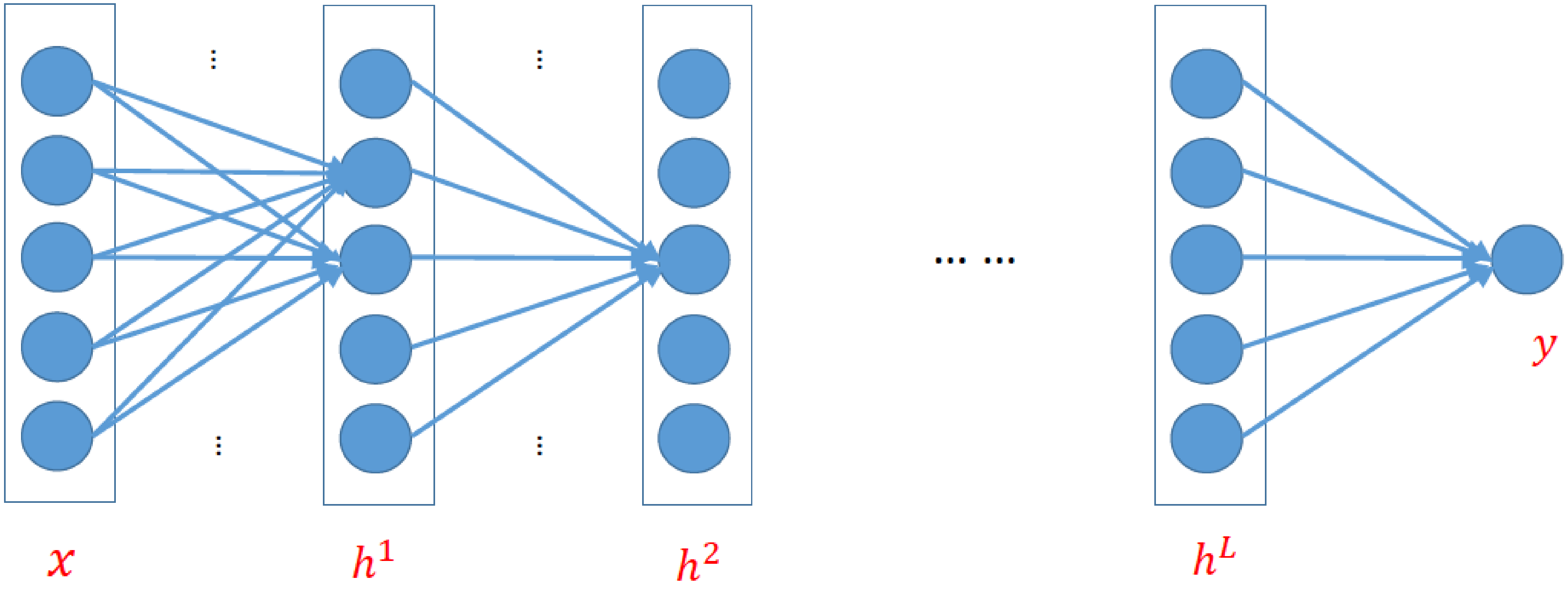
04. April. 2025.

Part 1: Introduction to Neural Networks

CSRC Computational Science Research Center

KOREA UNIVERSITY SEJONG CAMPUS

Artificial Neural Networks (ANNs) have been utilized for finding a mathematical relationship between input data x and output data y



Typical way to explain the structure of ANNs

How do we build an ANN step by step?

Mathematical description of the NN $(x \in \mathbb{R}, y \in \mathbb{R})$!

two variables are a single pair of real numbers

Linear model (transformation) $$y = w_1 x + b_1$$

With unknown parameters $\mathrm{p} = (w_1, b_1)$

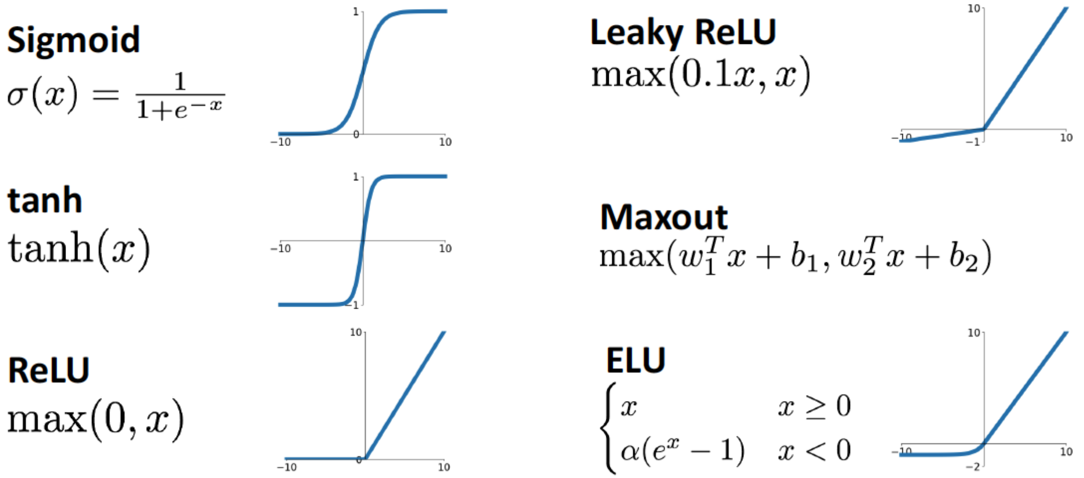
NN description (Layer/Node Representation): Each node represents a neuron, and each layer represents a transformation (usually matrix multiplication + activation)

Input layer Output layer

However, our dataset does not have to follow such linear relationship

Mathematical description of the NN $(x \in \mathbb{R}, y \in \mathbb{R})$**!**

To make nonlinearity,...



NN with 3-layers

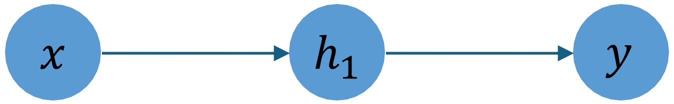
$$z_1 = w_1 x + b_1$$

$$h_1 = \sigma(z_1)$$

$$y = w_2 h_1 + b_2$$

With unknown parameters $\mathrm{p} = (w_1, b_1, w_2, b_2)$

NN description (Layer/Node Representation): Each node represents a neuron, and each layer represents a transformation (usually matrix multiplication + activation)



Input layer Hidden layer Output layer

Mathematical description of the NN $(x \in \mathbb{R}, y \in \mathbb{R})$!

$$z_1 = w_1 x + b_1$$
$$h_1 = \sigma(z_1)$$
$$z_2 = w_2 h_1 + b_2$$
$$h_2 = \sigma(z_2)$$
$$y = w_3 h_2 + b_3$$

NN with 4-layers

With unknown parameters $\mathrm{p} = (w_1, b_1, w_2, b_2, w_3, b_3)$

NN description (Layer/Node Representation): Each node represents a neuron, and each layer represents a transformation (usually matrix multiplication + activation)

Input layer 1st hidden 2nd hidden Output layer

We can also add a lot of nodes in each hidden layer!

NN with two nodes in a single hidden layer

x

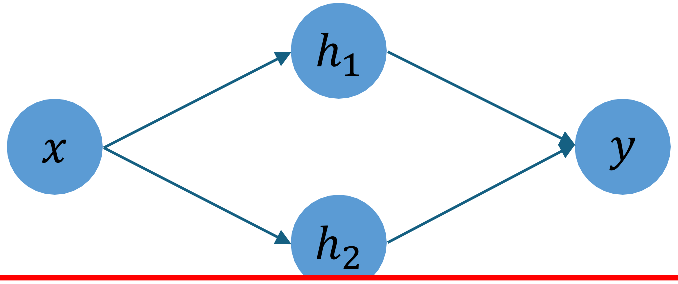
$$z_1 = w_{11}x + b_{11} \rightarrow h_1(z_1) = \sigma(z_1)$$

$$z_2 = w_{12}x + b_{12} \rightarrow h_2(z_2) = \sigma(z_2)$$

$$y = w_{21}h_1 + w_{22}h_2 + b_2$$

With unknown parameters $\mathrm{p} = (w_{11}, w_{12}, b_{11}, b_{12}, w_{21}, w_{22}, b_2)$

NN description (Layer/Node Representation): Each node represents a neuron, and each layer represents a transformation (usually matrix multiplication + activation)



In this way, we can add more nodes $(h_3, h_4, h_5 \ldots)$

How can we measure the performance of a neural network to determine whether it produces the correct output?

Once weights and biases are given, we can evaluate the L2 errors → Typical curve-fit procedure

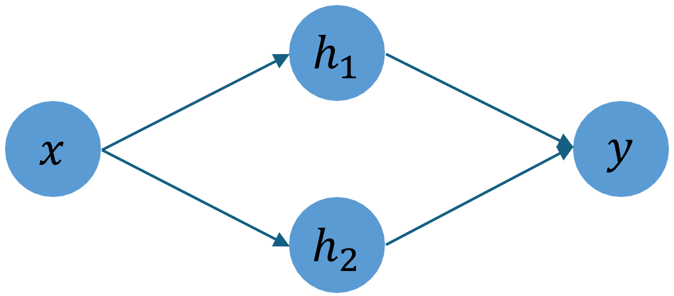
NN with two nodes in a single hidden layer

x

$z_1 = w_{11}x + b_{11}, \qquad h_1(z_1) = \sigma(z_1)$

$z_2 = w_{12}x + b_{12}, \qquad h_2(z_2) = \sigma(z_2)$

$y = w_{21}h_1 + w_{22}h_2 + b_2 = f(x;p)$



Dataset: $\{(x_i, y_i) | i = 1, \dots, n\}$

L2 error $\mathrm{E} = \sum_{i=1}^{n} |f(x_i) - y_i|^2$

: The difference between outputs of the NN and data y_i

: a function of p (i.e., $\mathrm{E} = \mathrm{E(p)}$)

Can you find the optimal p that minimizes the L2 error?

One famous way: gradient descent

p_0 is chosen randomly,

We can find a sequence of set of parameters $\{p_n\}$ via the following rule:

$$p_{n+1} = p_n - \delta \nabla_p E(p_n)$$

Calculating $\nabla_p E(p)$ would be difficult or complicate..

Let's consider forward propagation

NN with two nodes in a single hidden layer

$$\mathrm{p} = (1,1,1,1,1,1),\ \sigma(x) = x^2$$

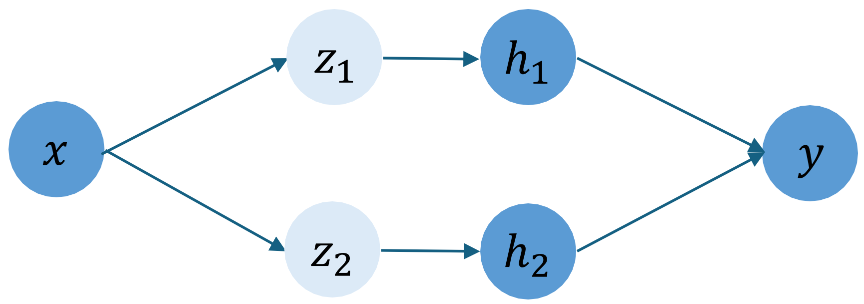
$$\{(x_i, y_i)\} = \{(1,2)\}$$

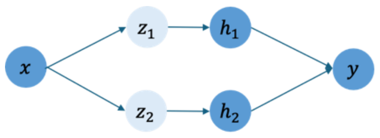
$$x$$

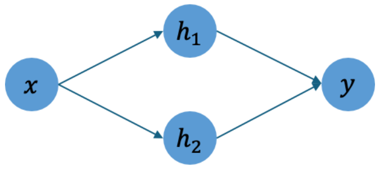
$$z_1 = w_{11}x + b_{11}, \qquad h_1(z_1) = \sigma(z_1)$$

$$z_2 = w_{12}x + b_{12}, \qquad h_2(z_2) = \sigma(z_2)$$

$$y = w_{21}h_1 + w_{22}h_2 + b_2$$







Feature	Computational Graph	Neural Network Representation
Node meaning	Variables or operations	Artificial neurons
Edge meaning	Flow of computation	Weighted connections
Purpose	Compute gradients (e.g., for training)	Define network structure and forward computation
Granularity	Fine-grained (add, multiply, etc.)	Coarse-grained (layers of neurons)
Interpretation	Math-level computation	Learning model structure

Computational graph: A graph of operations that represent how computations are done.

Let's consider forward propagation

NN with two nodes in a single hidden layer

$$p = (1,1,1,1,1,1),\ \sigma(x) = x^2$$

$$\{(x_i, y_i)\} = \{(1,2)\}$$

$$x$$

$x = 1$

$$z_1 = w_{11}x + b_{11}, \qquad h_1(z_1) = z_1^2$$

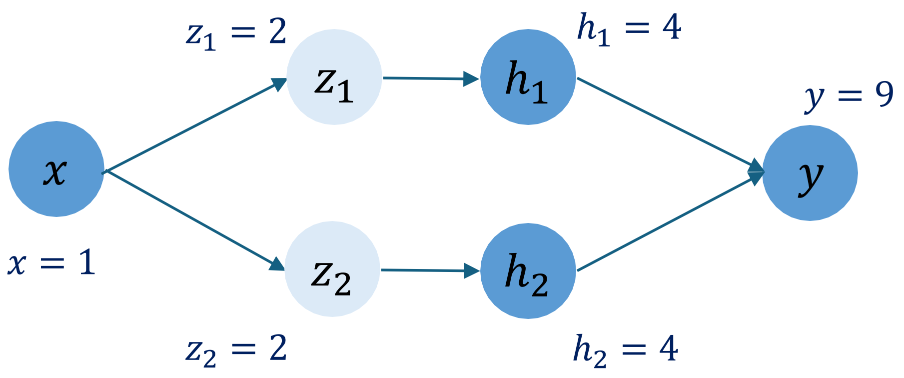
$z_1 = 1 * 1 + 1 = 2$ $\qquad h_1(2) = 4$

$$z_2 = w_{12}x + b_{12}, \qquad h_2(z_2) = z_2^2$$

$z_2 = 2$ $\qquad h_2 = 4$

$$y = w_{21}h_1 + w_{22}h_2 + b_2$$

$y = 1 * 4 + 1 * 4 + 1 = 9$



Computational graph: A graph of operations that represent how computations are done.

Let's consider **backward propagation (backpropagation)**

NN with two nodes in a single hidden layer

$$x$$

$$z_1 = w_{11}x + b_{11}, \qquad h_1(z_1) = z_1^2$$

$$z_2 = w_{12}x + b_{12}, \qquad h_2(z_2) = z_2^2$$

$$y = w_{21}h_1 + w_{22}h_2 + b_2$$

$$\mathrm{p} = (1,1,1,1,1,1),\ \sigma(x) = x^2$$

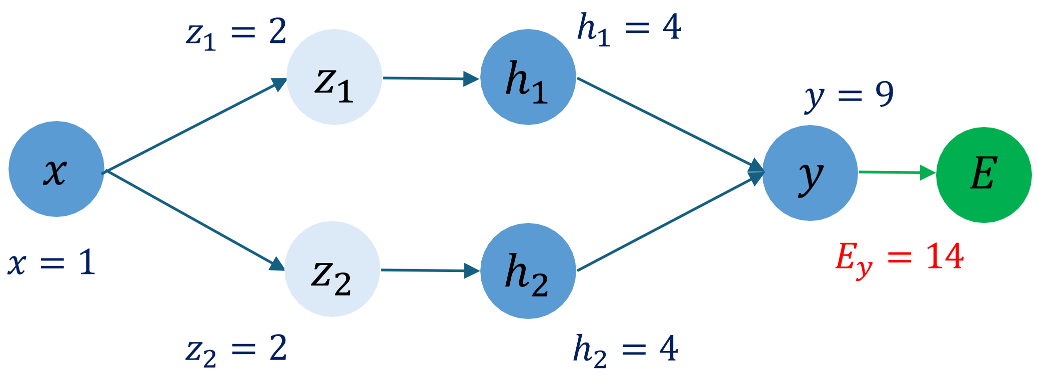
$$\{(x_i, y_i)\} = \{(1,2)\}$$

$$\mathrm{E(p)} = (f(1;p) - 2)^2$$

$$= (y - 2)^2$$

$$\partial_{w_{11}}\mathrm{E(p)} = \partial_{w_{11}}(y-2)^2 = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w_{11}}$$ Chain Rule

$$= \frac{\partial E}{\partial y} * \frac{\partial y}{\partial w_{11}} = 2(y-2) * \frac{\partial y}{\partial w_{11}} = 14 * \frac{\partial y}{\partial w_{11}}$$



We can directly calculate $\frac{\partial E}{\partial y}$ via forward propagation

Let's consider backward propagation (backpropagation)

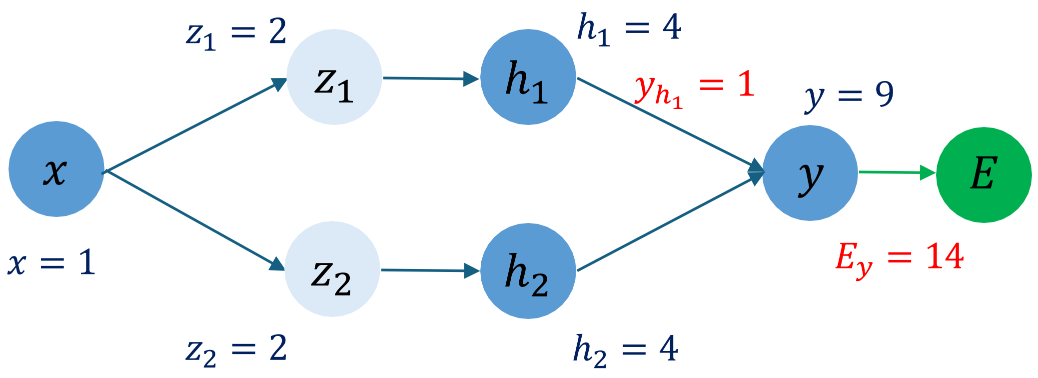
NN with two nodes in a single hidden layer

$$x$$

$$z_1 = w_{11}x + b_{11}, \qquad h_1(z_1) = z_1^2$$

$$z_2 = w_{12}x + b_{12}, \qquad h_2(z_2) = z_2^2$$

$$y = w_{21}h_1 + w_{22}h_2 + b_2$$



$$\mathrm{p} = (1,1,1,1,1,1),\ \sigma(x) = x^2$$

$$\{(x_i, y_i)\} = \{(1,2)\}$$

$$\mathrm{E(p)} = (f(1;p) - 2)^2$$

$$= (y - 2)^2$$

$$\partial_{w_{11}}\mathrm{E(p)} = \partial_{w_{11}}(y-2)^2 = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w_{11}}$$ Chain Rule

$$= \frac{\partial E}{\partial y} * \frac{\partial y}{\partial w_{11}} = 2(y-2) * \frac{\partial y}{\partial w_{11}} = 14 * \frac{\partial y}{\partial w_{11}}$$

$$\frac{\partial y}{\partial w_{11}} = \frac{\partial y}{\partial h_1}\frac{\partial h_1}{\partial w_{11}} + \underbrace{\frac{\partial y}{\partial h_2}\frac{\partial h_2}{\partial w_{11}}}_{} = w_{21} * \frac{\partial h_1}{\partial w_{11}} = 1 * \frac{\partial h_1}{\partial w_{11}}$$

= 0 (This pathway does not include w_{11})

Let's consider backward propagation (backpropagation)

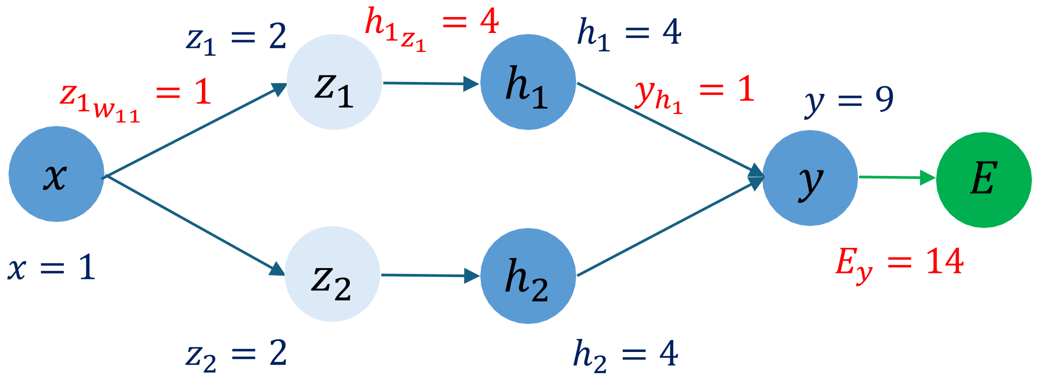
NN with two nodes in a single hidden layer

$$x$$

$$z_1 = w_{11}x + b_{11}, \qquad h_1(z_1) = z_1^2$$

$$z_2 = w_{12}x + b_{12}, \qquad h_2(z_2) = z_2^2$$

$$y = w_{21}h_1 + w_{22}h_2 + b_2$$



$$\mathrm{p} = (1,1,1,1,1,1),\ \sigma(x) = x^2$$

$$\{(x_i, y_i)\} = \{(1,2)\}$$

$$\mathrm{E(p)} = (f(1;p) - 2)^2$$

$$= (y-2)^2$$

$$\partial_{w_{11}}\mathrm{E(p)} = \partial_{w_{11}}(y-2)^2 = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w_{11}}$$ Chain Rule

$$= \frac{\partial E}{\partial y} * \frac{\partial y}{\partial w_{11}} = 2(y-2) * \frac{\partial y}{\partial w_{11}} = 14 * \frac{\partial y}{\partial w_{11}}$$

$$\frac{\partial y}{\partial w_{11}} = \frac{\partial y}{\partial h_1}\frac{\partial h_1}{\partial w_{11}} + \frac{\partial y}{\partial h_2}\frac{\partial h_2}{\partial w_{11}} = w_{21} * \frac{\partial h_1}{\partial w_{11}} = 1 * \frac{\partial h_1}{\partial w_{11}}$$

$$\frac{\partial h_1}{\partial w_{11}} = \frac{\partial h_1}{\partial z_1} * \frac{\partial z_1}{w_{11}} = 2z_1 * x_1 = 4 * 1$$

Let’s consider backward propagation (backpropagation)

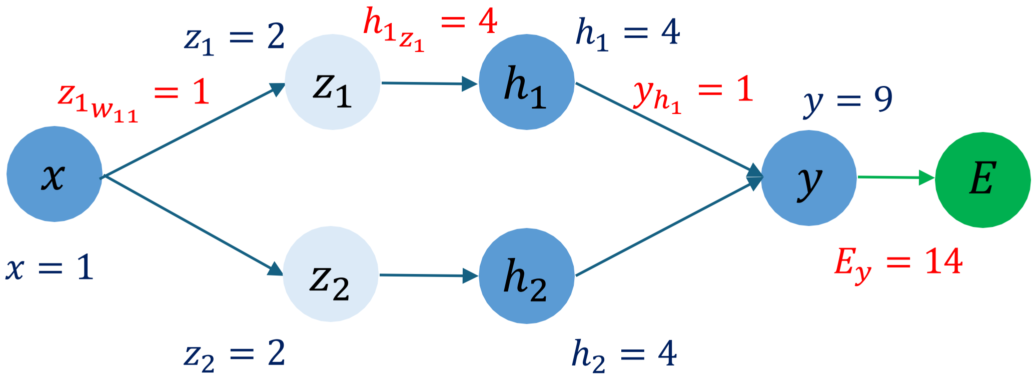
NN with two nodes in a single hidden layer

$$x$$

$$z_1 = w_{11}x + b_{11}, \qquad h_1(z_1) = z_1^2$$

$$z_2 = w_{12}x + b_{12}, \qquad h_2(z_2) = z_2^2$$

$$y = w_{21}h_1 + w_{22}h_2 + b_2$$



$$\mathrm{p} = (1,1,1,1,1,1), \ \sigma(x) = x^2$$

$$\{(x_i, y_i)\} = \{(1,2)\}$$

$$\mathrm{E(p)} = (f(1;p) - 2)^2$$

$$= (y-2)^2$$

$$\partial_{w_{11}}\mathrm{E(p)} = \partial_{w_{11}}(y-2)^2 = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w_{11}}$$ Chain Rule

$$= \frac{\partial E}{\partial y} * \frac{\partial y}{\partial w_{11}} = 2(y-2) * \frac{\partial y}{\partial w_{11}} = 14 * \frac{\partial y}{\partial w_{11}}$$

$$\frac{\partial y}{\partial w_{11}} = \frac{\partial y}{\partial h_1}\frac{\partial h_1}{\partial w_{11}} + \frac{\partial y}{\partial h_2}\frac{\partial h_2}{\partial w_{11}} = w_{21} * \frac{\partial h_1}{\partial w_{11}} = 1 * \frac{\partial h_1}{\partial w_{11}}$$

$$\frac{\partial h_1}{\partial w_{11}} = \frac{\partial h_1}{\partial z_1} * \frac{\partial z_1}{w_{11}} = 2z_1 * x_1 = 4 * 1$$

$$\partial_{w_{11}}\mathrm{E(p)} = \frac{\partial E}{\partial y} * \frac{\partial f(1;p)}{\partial h_1} * \frac{\partial h_1}{\partial z_1} * \frac{\partial z_1}{w_{11}} = 14 * 1 * 4 * 1 = 56$$

Backpropagation: Separating one heavy task to several number of simple tasks via chain rule

Let's consider backward propagation (backpropagation)

NN with two nodes in a single hidden layer

$$x$$

$$z_1 = w_{11}x + b_{11}, \qquad h_1(z_1) = z_1^2$$

$$z_2 = w_{12}x + b_{12}, \qquad h_2(z_2) = z_2^2$$

$$y = w_{21}h_1 + w_{22}h_2 + b_2$$

$$\mathrm{p} = (1,1,1,1,1,1),\ \sigma(x) = x^2$$

$$\{(x_i, y_i)\} = \{(1,2)\}$$

$$\mathrm{E(p)} = (f(1;p) - 2)^2$$

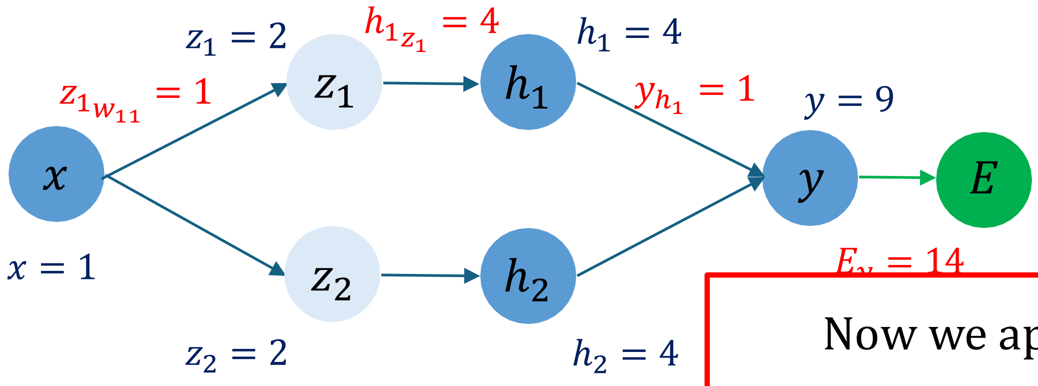
$$= (y - 2)^2$$

$$\partial_{w_{11}}\mathrm{E(p)} = \partial_{w_{11}}(y-2)^2 = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w_{11}}$$ Chain Rule

$$= \frac{\partial E}{\partial y} * \frac{\partial y}{\partial w_{11}} = 2(y-2) * \frac{\partial y}{\partial w_{11}} = 14 * \frac{\partial y}{\partial w_{11}}$$

$$\frac{\partial y}{\partial w_{11}} = \frac{\partial y}{\partial h_1}\frac{\partial h_1}{\partial w_{11}} + \frac{\partial y}{\partial h_2}\frac{\partial h_2}{\partial w_{11}} = w_{21} * \frac{\partial h_1}{\partial w_{11}} = 1 * \frac{\partial h_1}{\partial w_{11}}$$

$$\frac{\partial h_1}{\ } = \frac{\partial h_1}{\ } * \frac{\partial z_1}{\ } = 2z_1 * x_1 = 4 * 1$$



Now we apply gradient descent …

$$new\ w_{11} = old\ w_{11} - \delta * 56$$

$$* 1 * 4 * 1 = 56$$

Backpropagation: Separating one heavy task to several number of simple tasks via chain rule

Later, we will design NN using python! → Colab

```python
import numpy as np
from numpy.random import randn

N, Din, H, Dout = 64, 1000, 100, 10
x, y = randn(N, Din), randn(N, Dout)
w1, w2 = randn(Din, H), randn(H, Dout)
for t in range(10000):
  h = 1.0 / (1.0 + np.exp(-x.dot(w1)))
  y_pred = h.dot(w2)
  loss = np.square(y_pred - y).sum()
  dy_pred = 2.0 * (y_pred - y)
  dw2 = h.T.dot(dy_pred)
  dh = dy_pred.dot(w2.T)
  dw1 = x.T.dot(dh * h * (1 - h))
  w1 -= 1e-4 * dw1
  w2 -= 1e-4 * dw2
```

Initialize weights and bias

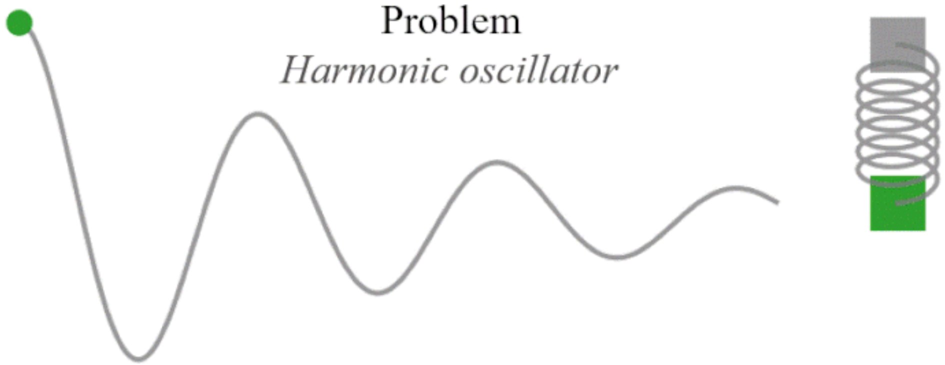
Compute L2 loss $E = \sum_{i=1}^{n} |f(x_i) - y_i|^2$

Compute gradients $\frac{\partial}{\partial w} L$

Gradient descent $w \leftarrow w - \frac{\partial}{\partial w} L$

Part 2: Application of NNs to Differential Eqns

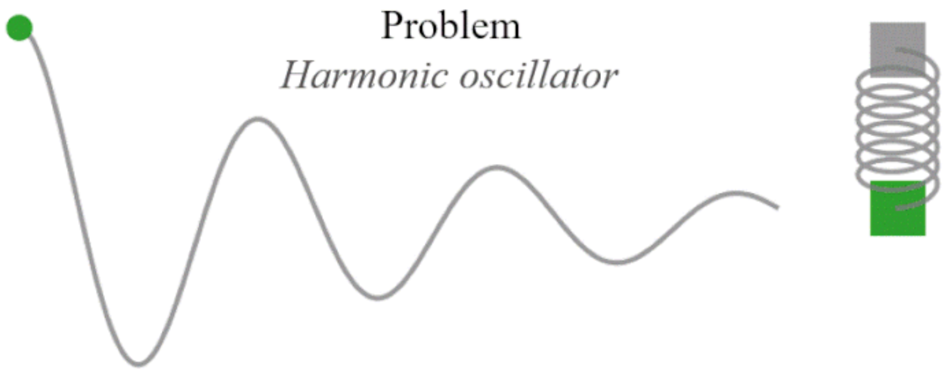
Why PINN?



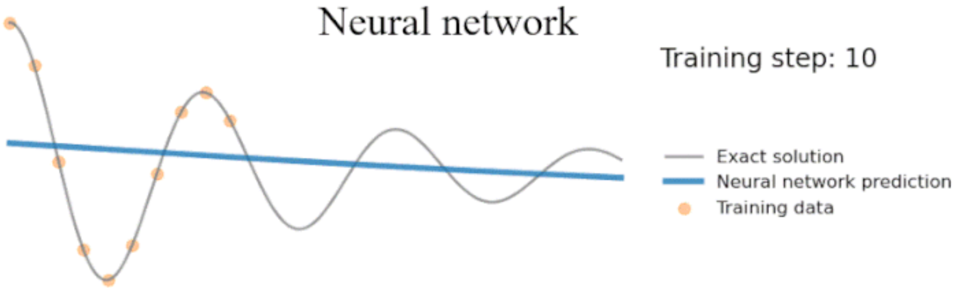
$$m\frac{d^2u}{dx^2} + \mu\frac{du}{dx} + ku = 0$$

Data obtained from Scientific phenomena can be described by a physics law (~Nonlinear)

Conventional neural networks can effectively fit observed data



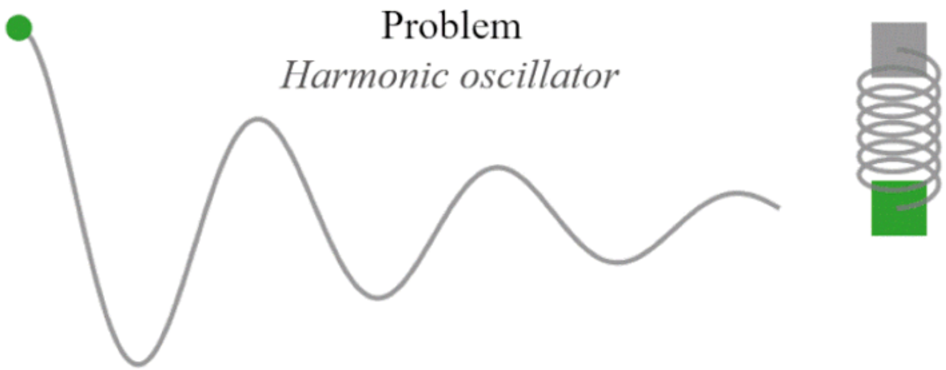
$$m\frac{d^2u}{dx^2} + \mu\frac{du}{dx} + ku = 0$$

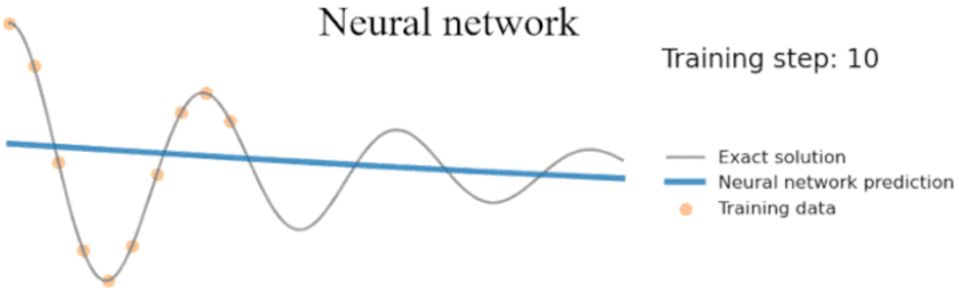


But they are unable to capture the physics of unobservable regions.

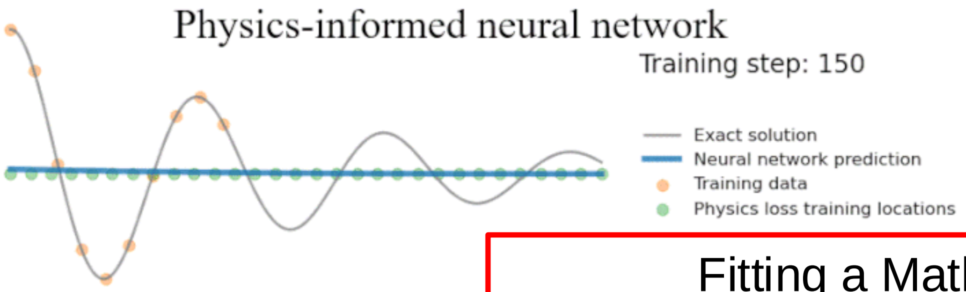
$$\min \frac{1}{N}\sum_{i}^{N}(u_{\mathrm{NN}}(x_i;\theta) - u_{\mathrm{true}}(x_i))^2$$

Physics-informed neural networks can be used to data-driven science Scientific discovery





Conventional neural networks can effectively fit observable data, but they are unable to capture the physics of unobservable regions.



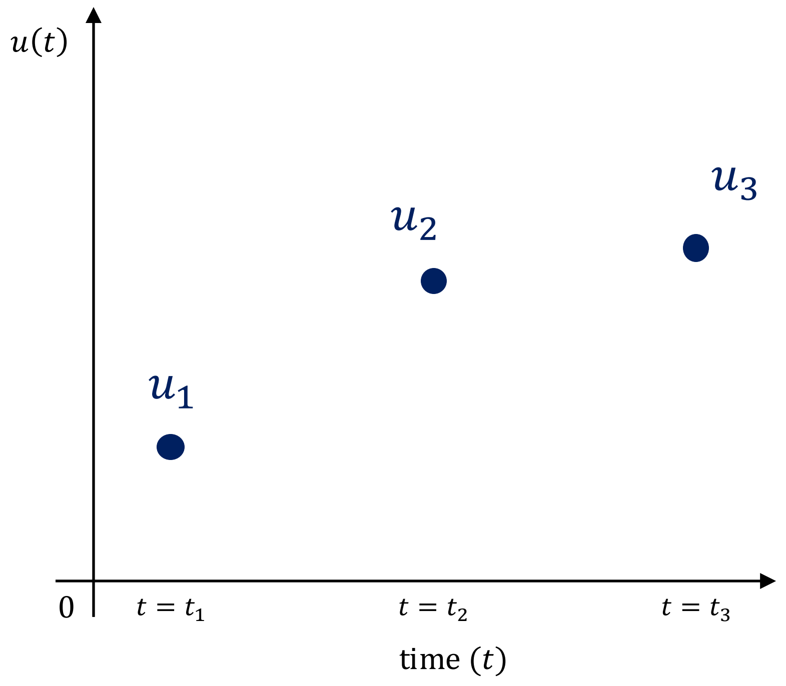
PINNs integrate physics knowledge into the neural network structures, enabling the extension to unobservable regions.

Fitting a Mathematical Model to Real Data:
From Classical Optimization to the Application of PINNs

Raissi, Maziar, et al, *Journal of Computational physics* 378 (2019): 686-707.

Artificial neural networks u_{NN} can fit the given data by minimizing data loss

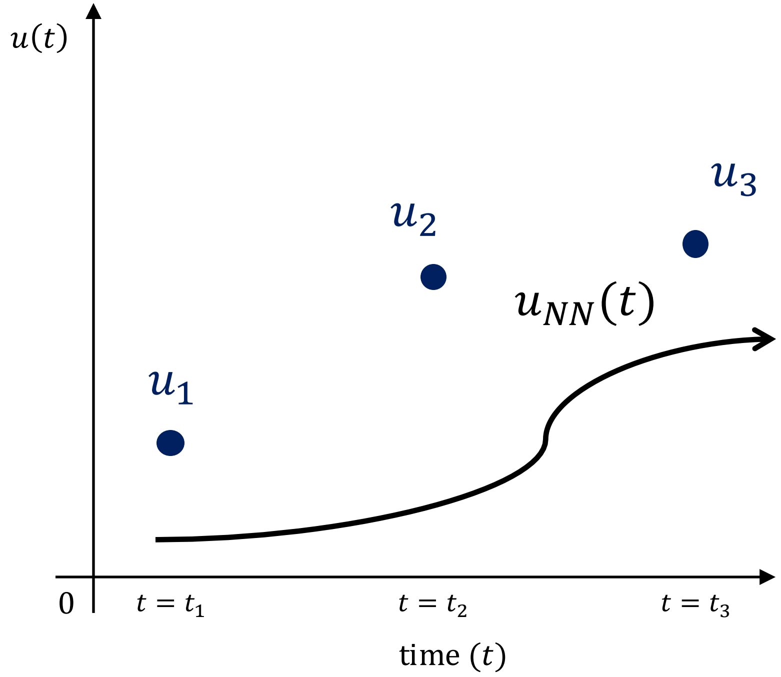
Given data $\{t_i, u_i\}_{i=1}^{3}$ and differential equation $\boldsymbol{u'(t) = au(t) + b}$



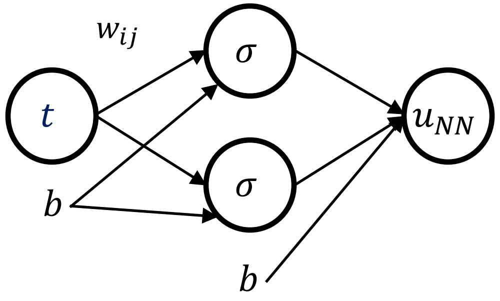
NN: A function with linear transform and nonlinear activation functions

Artificial neural networks u_{NN} can fit the given data by minimizing data loss

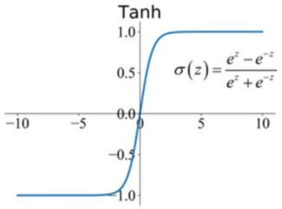
Given data $\{t_i, u_i\}_{i=1}^{3}$ and differential equation $\boldsymbol{u'(t) = au(t) + b}$



Step 1: Construct an artificial Neural Network (NN), $u_{NN}(t; \theta_u)$

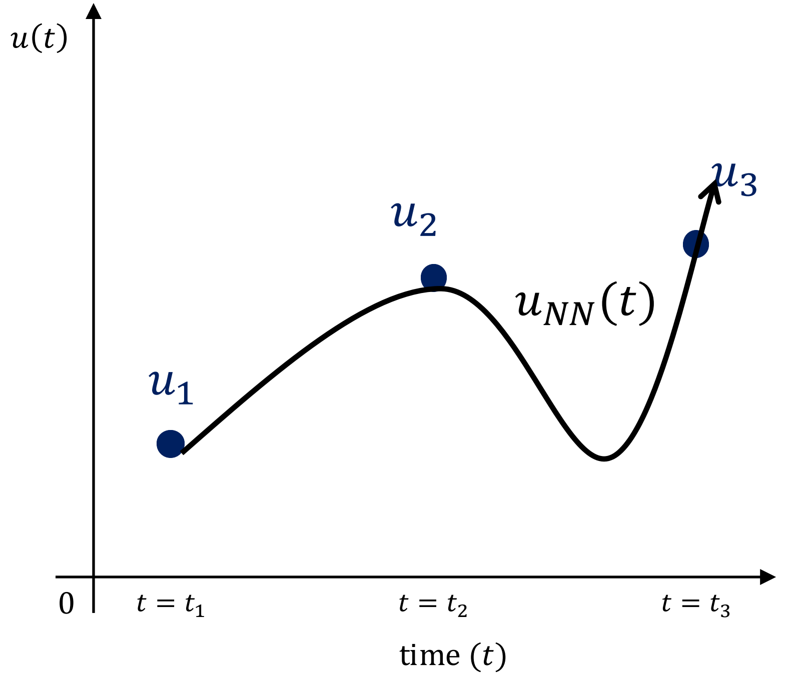


$$u_{NN}(t;\ \ldots w_{ij}, b_{ij}, \ldots) := u_{NN}(t; \theta_u)$$

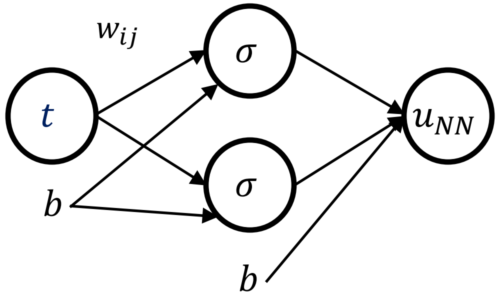


Artificial neural networks u_{NN} can fit the given data by minimizing data loss

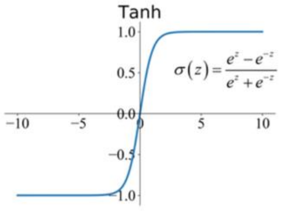
Given data $\{t_i, u_i\}_{i=1}^{3}$ and differential equation $\boldsymbol{u'(t) = au(t) + b}$



Step 1: Construct an artificial Neural Network (NN), $u_{NN}(t; \theta_u)$



$$u_{NN}\left(t;\ \ldots w_{ij}, b_{ij}, \ldots\right) \coloneqq u_{NN}(t; \theta_u)$$

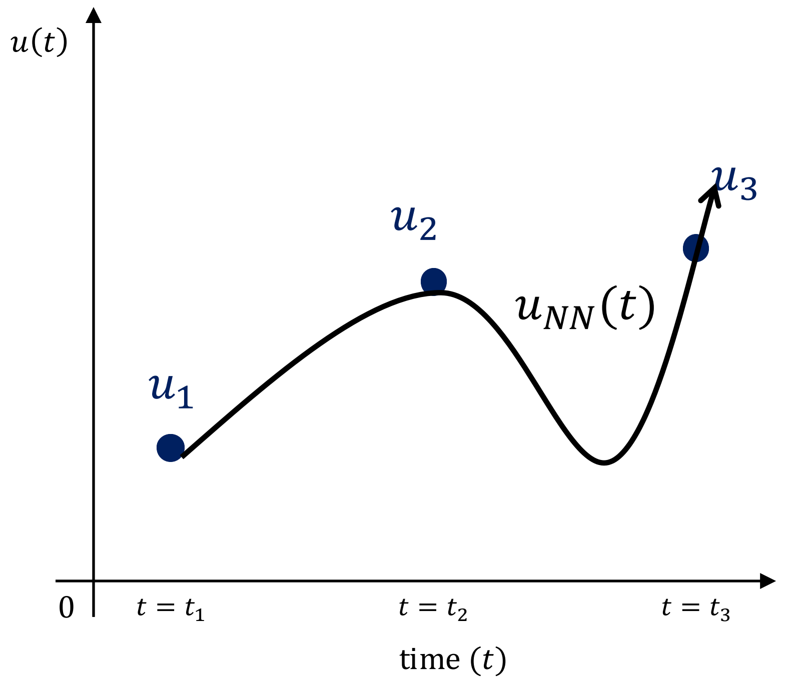


Step 2: Fit the NN that passes through all observation points

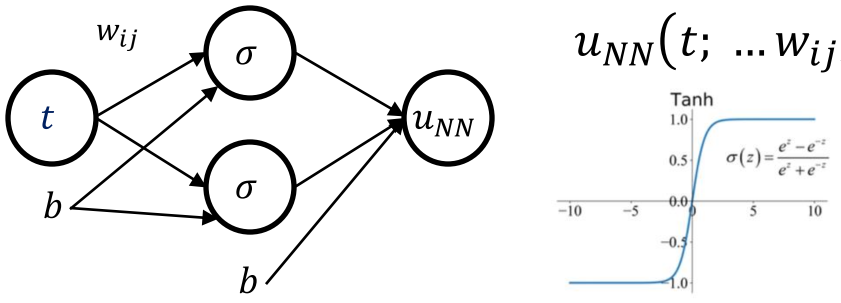
(E2 Loss) $$L_{data}(\theta_u) \coloneqq \sum_{i=1}^{3} |u_i - u_{NN}(t_i; \theta_u)|^2 \to 0$$

Artificial neural networks u_{NN} can fit the given data by minimizing data loss

Given data $\{t_i, u_i\}_{i=1}^{3}$ and differential equation $\boldsymbol{u'(t) = au(t) + b}$



Step 1: Construct an artificial Neural Network (NN), $u_{NN}(t;\theta_u)$



$$u_{NN}(t;\ \ldots w_{ij}, b_{ij}, \ldots) \coloneqq u_{NN}(t;\theta_u)$$

Step 2: Fit the NN that passes through all observation points

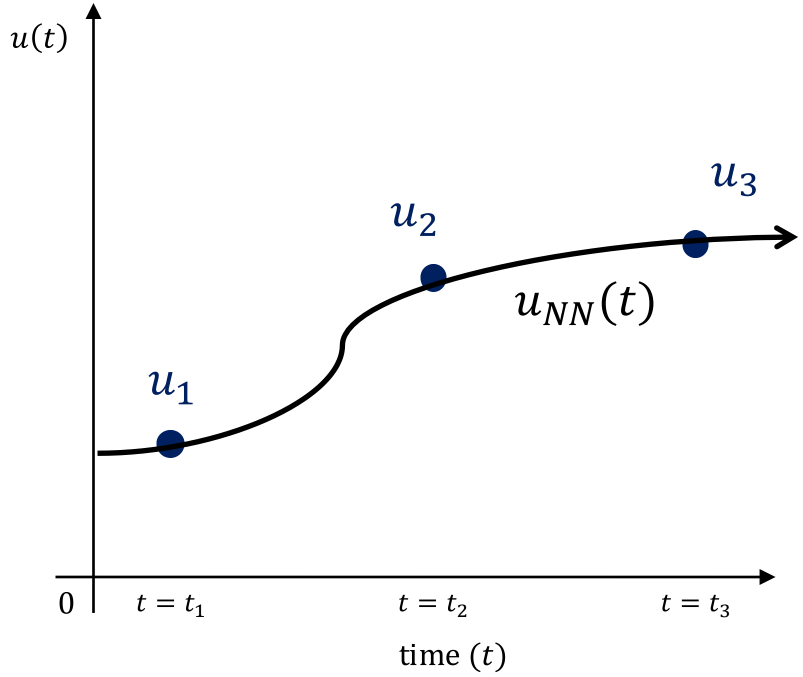
(E2 Loss) $$L_{data}(\theta_u) \coloneqq \sum_{i=1}^{3} |u_i - u_{NN}(t_i;\theta_u)|^2 \to 0$$

However,

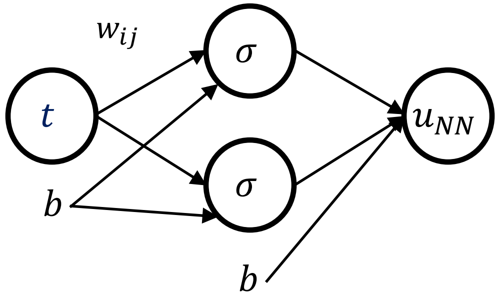
$$u'_{NN}(t) \neq a u_{NN}(t) + b$$

We add constraints to u_{NN} to fit both data points and the model simultaneously.

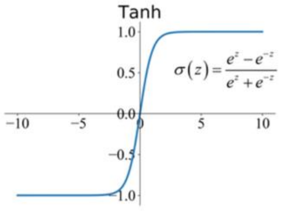
Given data $\{t_i, u_i\}_{i=1}^{3}$ and differential equation $\boldsymbol{u'(t) = au(t) + b}$



Step 1: Construct an artificial Neural Network (NN), $u_{NN}(t; \theta_u)$



$$u_{NN}(t;\ \dots w_{ij}, b_{ij}, \dots) := u_{NN}(t; \theta_u)$$



Step 2: Fit the NN that passes through all observation points

(E2 Loss) $$L_{data}(\theta_u) := \sum_{i=1}^{3} |u_i - u_{NN}(t_i; \theta_u)|^2 \to 0$$

M number of collocation time points $t_j \in [0, T]$

Sample from [0,T] with equal distance

$(dt = t_j - t_{j-1})$

or,

$t_j \sim Unif(0, T)$

Step 3: Input the NN to the equation

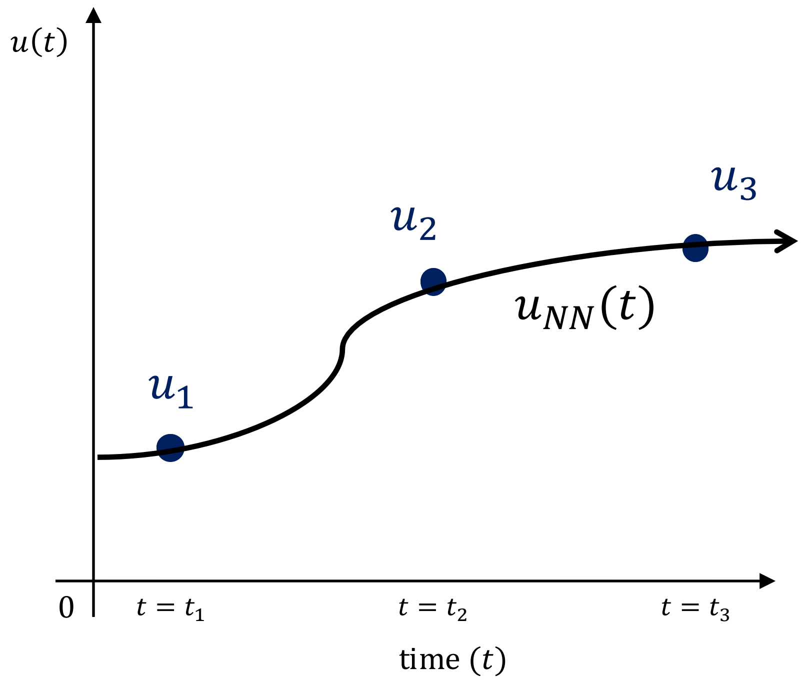
(Regularization) $$L_{reg}(\theta_u) := \sum_{j=1}^{M} \left| u'_{NN}(t_j; \theta_u) - \left(a u_{NN}(t_j; \theta_u) + b\right) \right|^2 \to 0$$

u_{NN} is 'differentiable' with respect to t

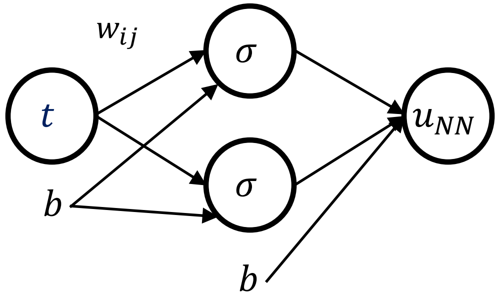
$$\boxed{u'_{NN}(t) \approx a u_{NN}(t) + b}$$

We add constraints to u_{NN} to fit both data points and the model simultaneously.

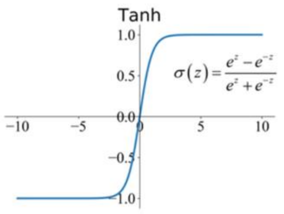
Given data $\{t_i, u_i\}_{i=1}^{3}$ and differential equation $\boldsymbol{u'(t) = au(t) + b}$



Step 1: Construct an artificial Neural Network (NN), $u_{NN}(t; \theta_u)$



$$u_{NN}(t;\ \ldots w_{ij}, b_{ij}, \ldots) \coloneqq u_{NN}(t; \theta_u)$$



Step 2: Fit the NN that passes through all observation points

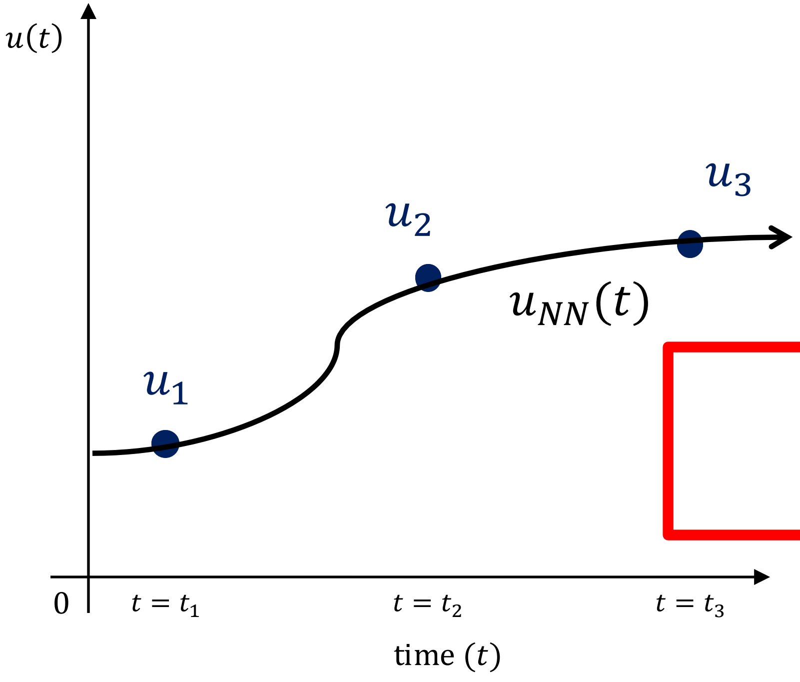
(E2 Loss) $$L_{data}(\theta_u) \coloneqq \sum_{i=1}^{3} |u_i - u_{NN}(t_i; \theta_u)|^2 \to 0$$

Step 3: Input the NN to the equation with initial guesses $\tilde{a}$ and $\tilde{b}$

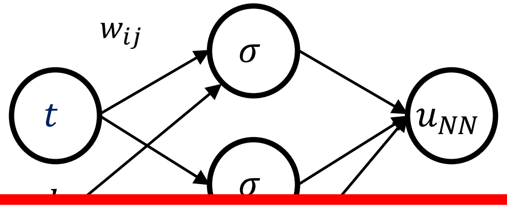
(Regularization) $$L_{reg}(\theta_u; \tilde{a}, \tilde{b}) \coloneqq \sum_{j=1}^{M} \left| u'_{NN}(t_j; \theta_u) - \left(\tilde{a} u_{NN}(t_j; \theta_u) + \tilde{b}\right) \right|^2 \to 0$$

We add constraints to u_{NN} to fit both data points and the model simultaneously.

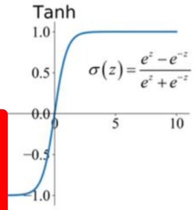
Given data $\{t_i, u_i\}_{i=1}^{3}$ and differential equation $\boldsymbol{u'(t) = au(t) + b}$



Step 1: Construct an artificial Neural Network (NN), $u_{NN}(t; \theta_u)$



$$u_{NN}(t;\ \dots w_{ij}, b_{ij}, \dots) \coloneqq u_{NN}(t; \theta_u)$$



$$\boxed{\begin{gathered} u'_{NN}(t) \approx \tilde{a} u_{NN}(t) + \tilde{b} \\ \tilde{a} \to a, \tilde{b} \to b \end{gathered}}$$

through all observation points

(E2 Loss) $$L_{data}(\theta_u) \coloneqq \sum_{i=1}^{3} |u_i - u_{NN}(t_i; \theta_u)|^2 \to 0$$

Step 3: Input the NN to the equation with initial guesses $\tilde{a}$ and $\tilde{b}$

(Regularization) $$L_{reg}(\theta_u; \tilde{a}, \tilde{b}) \coloneqq \sum_{j=1}^{M} \left| u'_{NN}(t_j; \theta_u) - \left(\tilde{a} u_{NN}(t_j; \theta_u) + \tilde{b}\right) \right|^2 \to 0$$

Thank you!

Let's jump into colab implementation

Korea_htj@korea.ac.kr

Appendix: Application of PINNs...

Computational Science Research Center

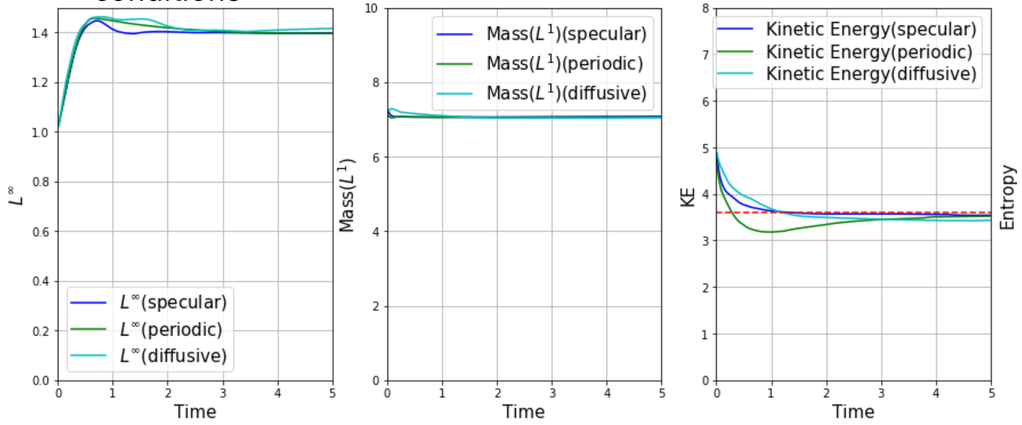
KOREA UNIVERSITY SEJONG CAMPUS

Real-world examples 1: Solving DEs

Solution of Fokker-Planck equation

$$\partial_t u + v\partial_x u = \partial_v(\partial_v u + vu)$$

Physical quantities from u_{NN} with different boundary conditions

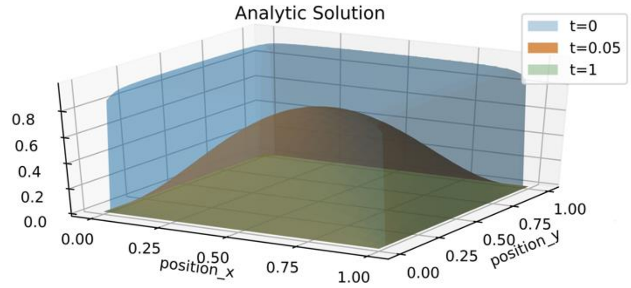


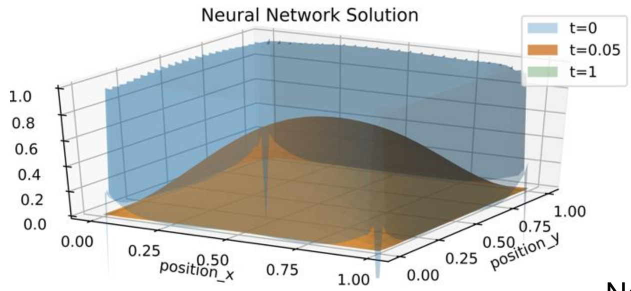
Jin Woo Jang
POSTECH

Jae Young Lee
Joongang
University

Solutions & Parameters of PDE

$$\partial_t u - {\color{blue}D}(\partial_{xx} u - \partial_{yy} u) = 0$$





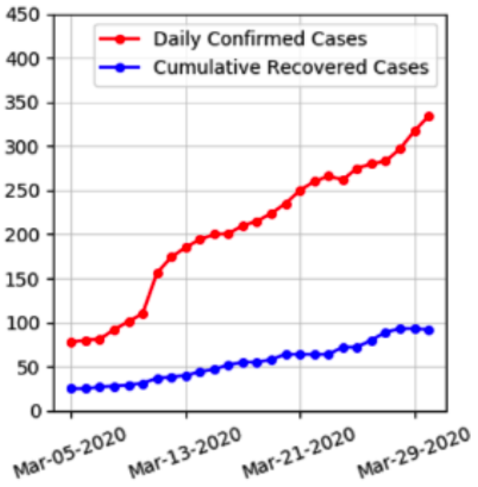
Hwijae Son
Konkuk
University

Eunheui Kim
California State University Long Beach

Fokker-Planck equation via the neural network approach (J. Comput. Phys.)

Neural Network Approach to Forward-Inverse Problems (Netw. Heterog. Media)

Real-world examples 2: Time-varying parameters in infectious diseases model.



COVID19 report

$I(t)$: The number of daily confirmed cases

$R(t)$: The cumulative number of recovered or deceased cases

(from Korea Disease Control and Prevention Agency)

$$\frac{dS}{dt} = -\beta(\mathrm{t})SI$$

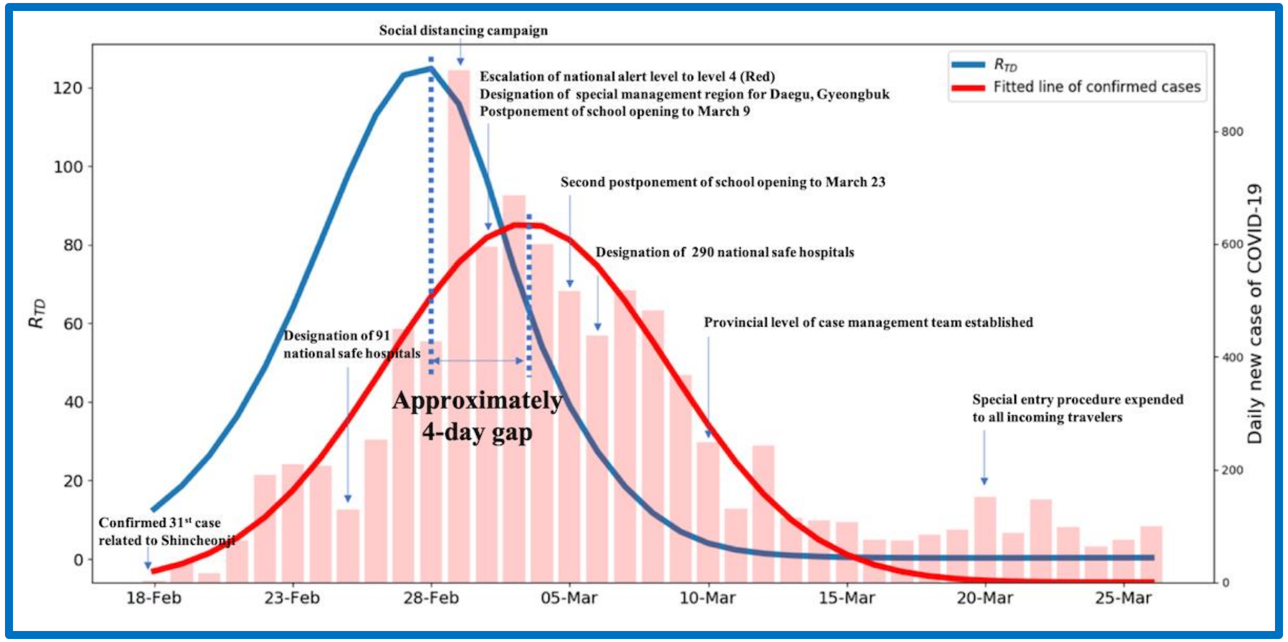
$$\frac{dI}{dt} = \beta(\mathrm{t})SI - \gamma I$$

$$\frac{dR}{dt} = \gamma(\mathrm{t})I$$

$\beta(\mathrm{t}) \leftarrow$ NN $\beta_{NN}(t)$

$\gamma(\mathrm{t}) \leftarrow$ NN $\gamma_{NN}(t)$

$$R_{TD}(t) \approx \frac{\beta_{NN}(t)}{\gamma_{NN}(t)}$$



Hyung Ju Hwang
POSTECH

Se Young Jung
Seoul National University Bundang Hospita

Hwijae Son
Konkuk University

Real-World Implications of a Rapidly Responsive COVID-19 Spread Model (J. Medical Internet Res.)

Real-world examples 3: Density-PINN to infer the cell signaling dynamics