- Foundation for PINN

Neural Network approach

  - *y = f*(x)*
    - x = input,
    - y = output
    - f = Function
  - $y = f * (x, \theta)$
    - $\theta = w + b$ 1st Key Ingredient for PINN
      - w = weights
      - b = *bias*

Loss Function: Is a measure of deviation or inaccuracy, or cost we want to minimize.

➔ L(θ) 2nd Key Ingredient for PINN

Learning Rate: It refers to gradient descent techniques used to adjust weights to minimize L(θ).

➔ $\theta' = \theta - \eta \Delta\theta(\theta)$ 3rd Key Ingredient for PINN

Physics in Neural Network

Differentia Equations:

2nd Partial Differential Equation -> $\frac{\partial T}{\partial t} = \frac{k\partial^2 T}{\rho c \partial x^2}$

where,

T = Temperature t = time

Heat Diffusivity = $k/\rho c$ x = distance

→ Example Differential Equation

$$F(x) = a\frac{d^2y}{dx^2} + b\frac{dy}{dx} + cy = 0 \quad , \quad y(0)=0 \ \& \ y(1)=1$$

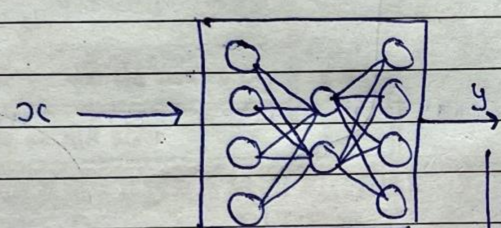
→ Approach = NN, by 2019, Raissi et. al Journal of Computatical Physics.

→ Idea

Prereq = Function ✓ (Universal Approximation)

= derivatives ✓ (Chain rule, Back prop)

= optimizer ✓



loss Function

$L_y(\theta)$ + $L_y(\theta)$ + $L_{y_b}(\theta)$ + $L_{y_0}(\theta)$

data Fit | Differential Equation Residual | Boundry Condition | Initial Condition

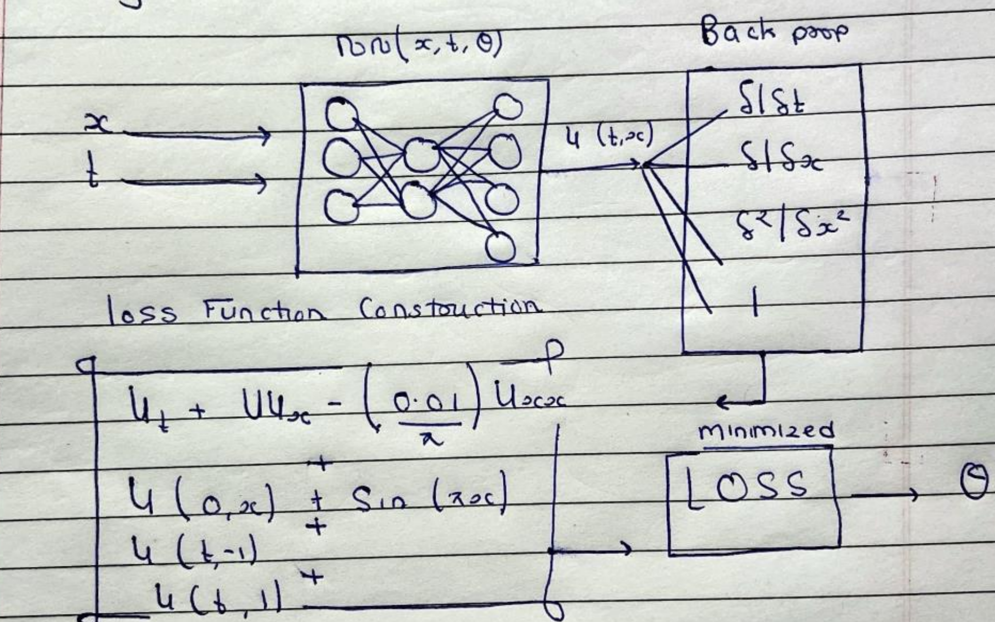
Initial Condition → If, Given

Differential Equation Residual → $\min \left[F(x) - 0 \right]$

Boundry Condition → $\min \left[[y(0) - y0]^2 + [y(1) - y1]^2 \right]$

Burger's Equation , M. Raissi J. Comp. Phy 2019

- A Fundamental partial differential Equation in various Fields.
- 1-D, U is Function of t and x
- $u_t + uu_x - \left(\frac{0.01}{\pi}\right) u_{xx} = 0$, $x \in [-1,1]$, $t \in [0,1]$
- Initial Condition = $u(0,x) = -\sin(\pi x)$.
- Boundry Condition = $u(t,-1) = u(t,1) = 0$



TensorFlow

```
def NN
def calculate(t,x):
    u = neural_net(tf.concat([t,x],1), w, b)
    return u
```

```
def PINN
def f(t,x):
    u = calculate(t,x)
    u_t = tf.gradients(u,t)[0]
    u_x = tf.gradients(u,x)[0]
    u_xx = tf.gradient(u_x,x)[0]
    f = u_t + u*u_x - (0.01/tf.pi)*u_xx
    return f
```