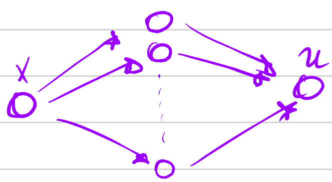
Solving PDE/ODE using neural networks by converting the differential equations into an optimization problem. Let's take a simple differential equation

$$\frac{d^2u}{dx^2} + a\frac{du}{dx} = b \qquad \text{B.C.} \begin{cases} u(0) = u_0 \\ u(1) = u_1 \end{cases}$$

<u>Assumption</u>: the function $u(x)$ is a neural network $NN(x)$, which takes $x$ as input and gives $u$ as output

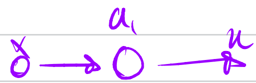



$$\underset{\text{output}}{u} = \underset{\text{input}}{NN(x)}$$

<u>Universal approximation theorem</u>:

Regardless of the PDE/ODE we can always approximate the solution $u(x)$ by a neural network.

Let's consider a simple case, one neuron and a linear activation function




$$a_1 = \sigma(w_1 x)$$

$$\Rightarrow u = w_2 \sigma(w_1 x)$$

$$\frac{du}{dx} = w_2 \sigma'(w_1 x) w_1$$

$$\frac{d^2u}{dx^2} = w_2 \sigma''(w_1 x) w_1^2$$

$$\frac{d^j u}{dx^j} = \dots$$

It is possible to compute all derivatives of $u$ with respect to $x$. If you have more neurons and more hidden layers, the same principle holds. This is known as back propagation.

If $\hat{u} = NN(x)$ we can find $u'(x)$, $u''(x)$ .. $u^{j}(x)$.

From here, I proposed the problem:

$$\text{Minimize} \left\{ \left[ \frac{d^2\hat{u}}{dx^2} + a\frac{d\hat{u}}{dx} - b \right]^2 \right\}$$

How can include the boundary condition?

$$\text{Minimize} \left\{ \left[ \frac{d^2\hat{u}}{dx^2} + a\frac{d\hat{u}}{dx} - b \right]^2 + \left[ \hat{u}(0) - u_0 \right]^2 + \left[ \hat{u}(1) - u_1 \right]^2 \right\}$$

this is known as the loss function. The loss function = cost function + BC / IC.

$$\text{cost function} \rightarrow \text{PDE/ODE}$$

and then, the loss function is minimized.

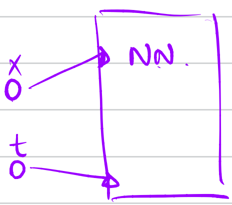
**Example:**

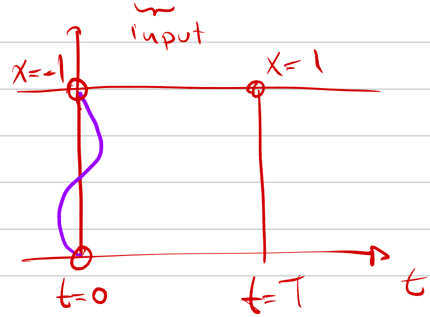
paper → Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations

Burger's equation → $u(x,t)$

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$
$$u(0, x) = -\sin(\pi x), \quad \Rightarrow \text{I.C.}$$
$$u(t, -1) = u(t, 1) = 0. \quad \Rightarrow \text{B.C.}$$

The postulate $\underbrace{u(x,t)}_{\text{output}} = NN(\underbrace{x,t}_{\text{input}})$

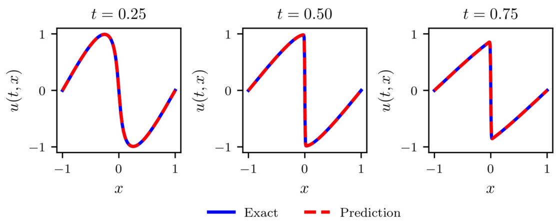



Now, we can define the loss function

$$\hat{u} = NN(x,t)$$

Loss function : $\left\{ \left[\hat{u}_t + \hat{u}_{xx} - \nu \hat{u}_x\right]^2 + \left[\hat{u}(0,x) + \sin(\pi x)\right]^2 + \left[\hat{u}(t,-1)\right]^2 + \left[\hat{u}(t,1)\right]^2 \right\}$




The solution works better than finite methods.