Ecuaciones diferenciales Ordinarias

Física de Fluidos - 1C 2026

$$\frac{\partial x}{\partial t} = F\left(t, x\right)$$

Ecuación diferencial odinaria

$$\frac{\partial x}{\partial t} = \alpha x$$

$$\frac{\partial x}{\partial t} = \alpha \cos(\beta x^2) + \gamma t$$

$$\frac{\partial^n x}{\partial t^n} = F\left(t, x, \ldots, \frac{\partial^{n-1} x}{\partial t^{n-1}}\right)$$

Ecuación diferencial odinaria de orden n

$$\frac{\partial^2 \theta}{\partial t^2} = -\frac{g}{\ell}\sin\theta$$

$$\begin{cases} \dfrac{\partial \theta}{\partial t} = \omega \\ \dfrac{\partial \omega}{\partial t} = -\dfrac{g}{\ell}\sin\theta \end{cases} \xrightarrow{\boldsymbol{\Omega} = (\theta, \omega)} \frac{\partial \boldsymbol{\Omega}}{\partial t} = \mathbf{F}(\boldsymbol{\Omega})$$

EDO

continuas

$t \in [0, T]$

$x(t)$

discretas

$t_0, t_1, \ldots, t_N$

$x_0, x_1, \ldots, x_N$

$t_{j+1} = t_j + j\Delta t$

continuas → **discretas**

$$t \in [0, T] \longrightarrow t_0, t_1, \ldots, t_N \qquad t_j = t_0 + j\Delta t$$

$$x(t) \longrightarrow x_0, x_1, \ldots, x_N$$

Taylor a primer orden

$$x(t_1) = x(t_0 + \Delta t) = x_0 + \frac{\partial x}{\partial t}\Big|_{(t_0, x_0)} \Delta t + O(\Delta t^2)$$

$\frac{\partial x}{\partial t}\Big|_{(t_0, x_0)} = u(t_0, x_0)$

Nos quedamos a 1er orden (**Euler**)

$$x_1 = x_0 + u(t_0, x_0)\Delta t$$

continuas → **discretas**

$t \in [0, T]$ → $t_0, t_1, \ldots, t_N$ $\quad t_{j+1} = t_j + j\Delta t$

$x(t)$ → $x_0, x_1, \ldots, x_N$

Taylor a primer orden

$$x(t_1) = x(t_0 + \Delta t) = x_0 + \frac{\partial x}{\partial t}\Big|_{(t_0, x_0)} \Delta t + O(\Delta t^2)$$

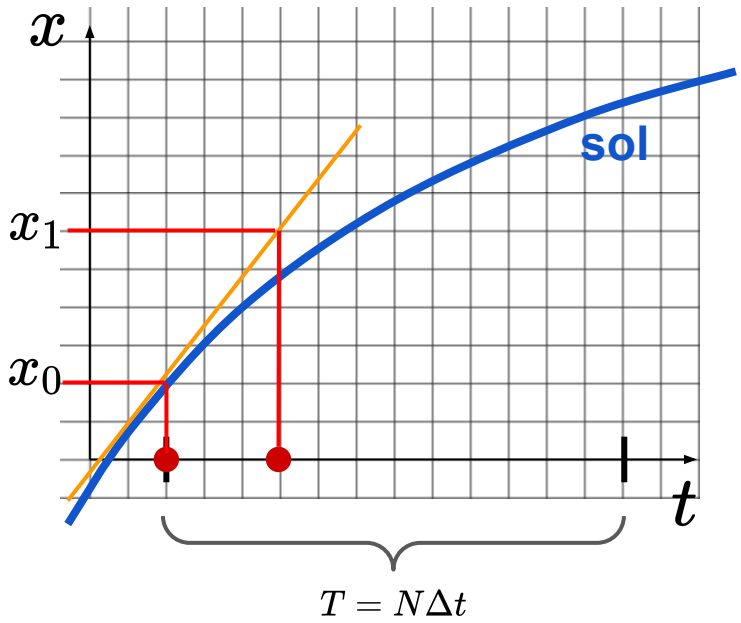
($\frac{\partial x}{\partial t}\Big|_{(t_0,x_0)} = u(t_0, x_0)$)

Nos quedamos a 1er orden (**Euler**)

$$x_1 = x_0 + u(t_0, x_0)\Delta t$$

Error local

$$E = |x_1 - x(t_1)| = O(\Delta t^2)$$



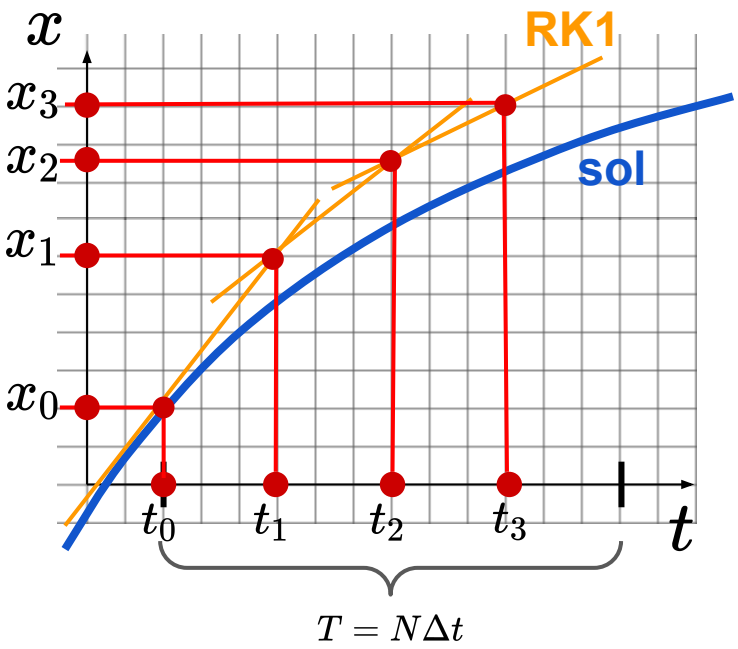
Runge Kutta Orden 1: Método de Euler

Fórmula

$$x_{j+1} = x_j + u(t_j, x_j)\Delta t$$

Error Global

$$E_g = |x_n - x(t_N)| = O(\Delta t)$$



Runge Kutta Orden 2

Taylor a orden 2

$$x(t_1) = x(t_0 + \Delta t) = x_0 + u(t_0, x_0)\Delta t + \frac{1}{2}u'(t_0, x_0)\Delta t^2 + O(\Delta t^3)$$

calcular esto puede ser complicado
y/o puede traer más errores

Runge Kutta Orden 2

Taylor a orden 2

$$x(t_1) = x(t_0 + \Delta t) = x_0 + u(t_0, x_0)\Delta t + \frac{1}{2}u'(t_0, x_0)\Delta t^2 + O(\Delta t^3)$$

calcular esto puede ser complicado
y/o puede traer más errores

RK2 error local y global del mismo orden que taylor con la ventaja de que el método requiere del cálculo de derivadas

Fórmula

$$x_{j+1} = x_j + (a_1 k_1 + a_2 k_2)\Delta t$$

$$k_1 = u(t_j, x_j)$$

$$k_2 = u(t_j + p_1 \Delta t, x_j + q_{11} k_1 \Delta t)$$

Las constantes tienen que cumplir ciertas ecuaciones

Runge Kutta Orden 2

Punto Medio

$$x_{j+1} = x_j + k_2 \Delta t$$

$$k_1 = u(t_j, x_j)$$

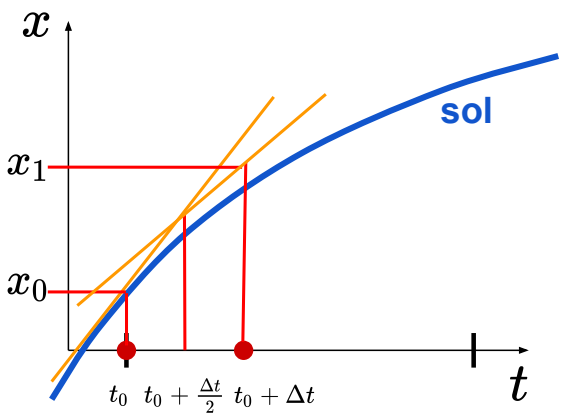
$$k_2 = u(t_j + \frac{\Delta t}{2}, x_j + k_1 \frac{\Delta t}{2})$$

Heun's

$$x_{j+1} = x_j + (k_1 + k_2) \frac{\Delta t}{2}$$

$$k_1 = u(t_j, x_j)$$

$$k_2 = u(t_j + \Delta t, x_j + k_1 \Delta t)$$



¿qué Δt es chico?

¿Qué tiempo es largo?

¿qué escalas de longitud pretendo obtener?

T: orden de los segundos
L: orden de los metros

T: decenas de horas
L: cientos de km

$$u(t,x,y) = U_0(\cos(ky)\hat{x} + \cos(kx)\hat{y})$$

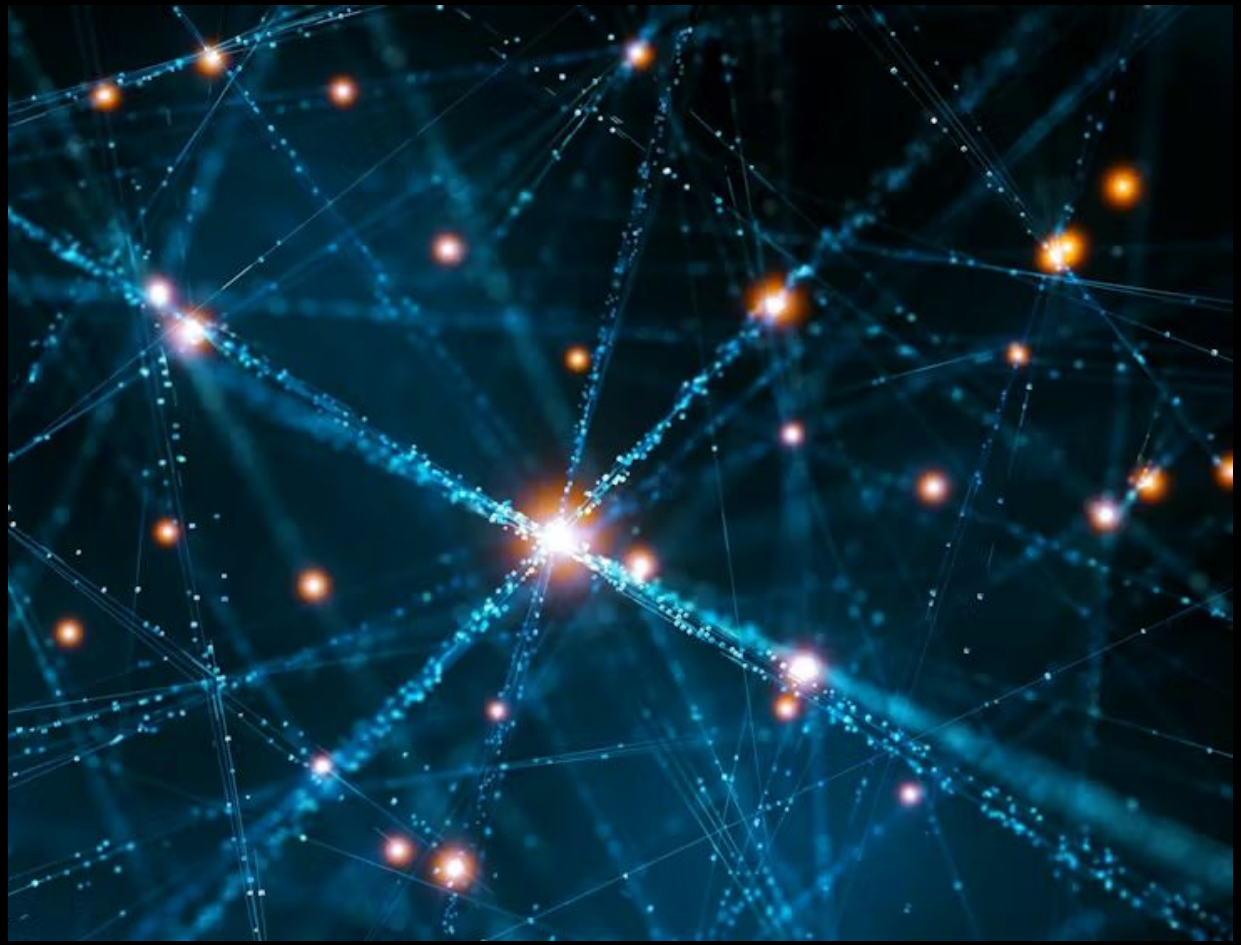
$$[U_0] = \frac{L}{T} \qquad [k] = \frac{1}{L}$$

Escalas características

$$\tau = \frac{k}{U_0} \qquad \mathcal{L} = \frac{1}{k}$$

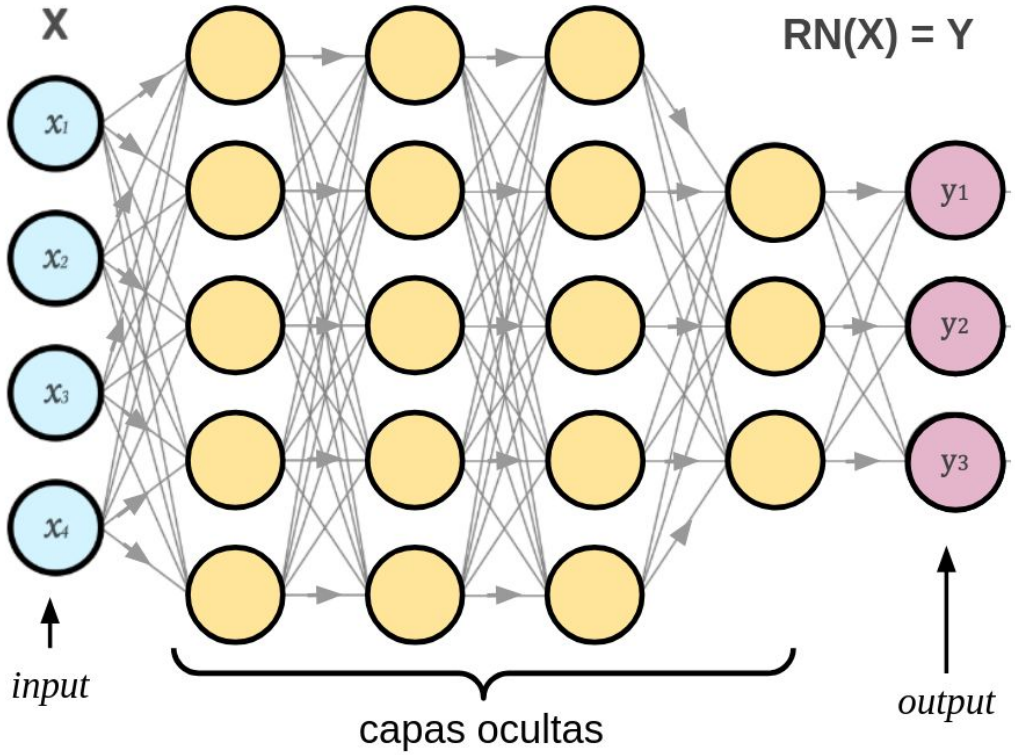
Adimensionalización

$$x = x'/k$$
$$y = y'/k \longrightarrow \frac{\partial \mathbf{X}'}{\partial t'} = \cos(y')\hat{x} + \cos(x')\hat{y}$$
$$t = t'\tau$$

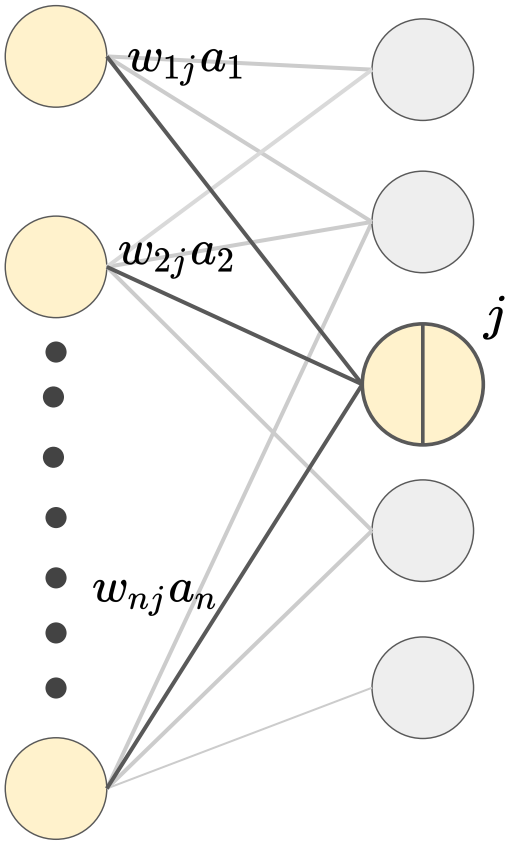


PINNs

Red Neuronal

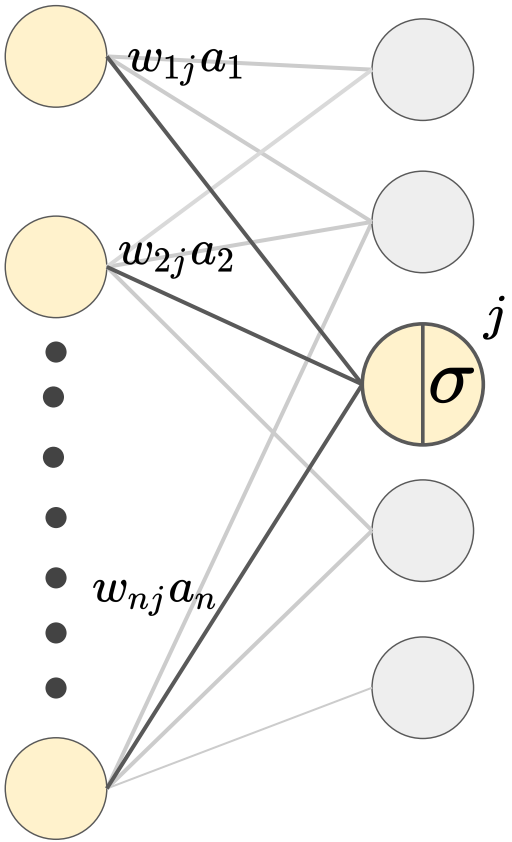


La neurona



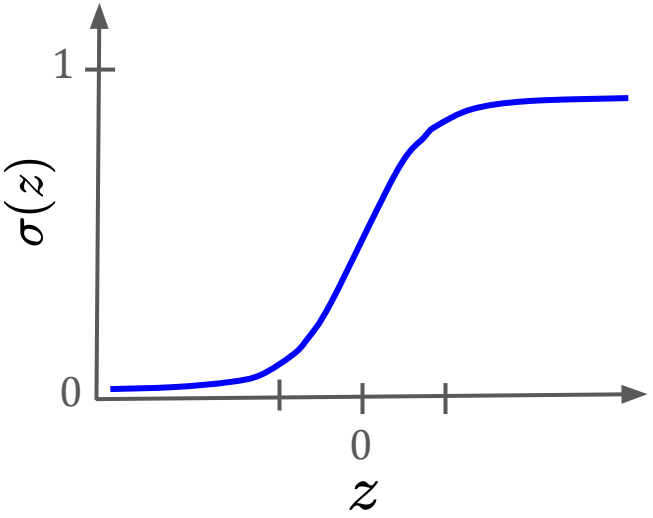
$$\sum_{i=1}^{n} w_{ij}a_i$$

La neurona

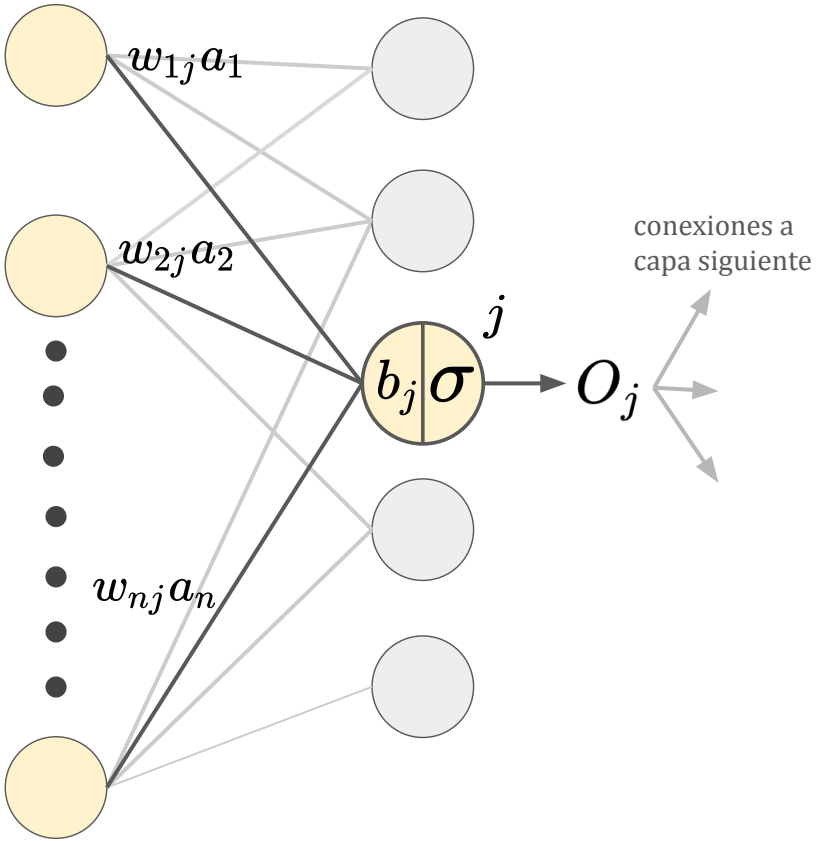


$$\sigma\left(\sum_{i=1}^{n} w_{ij}a_i\right)$$

Función de activación $\sigma(z)$

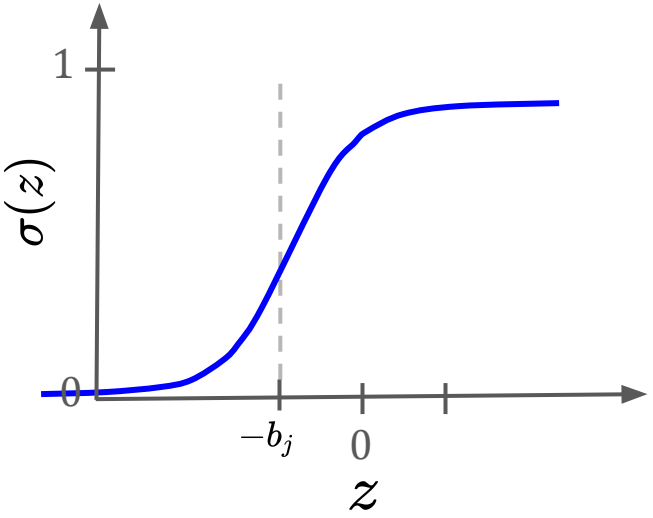


La neurona



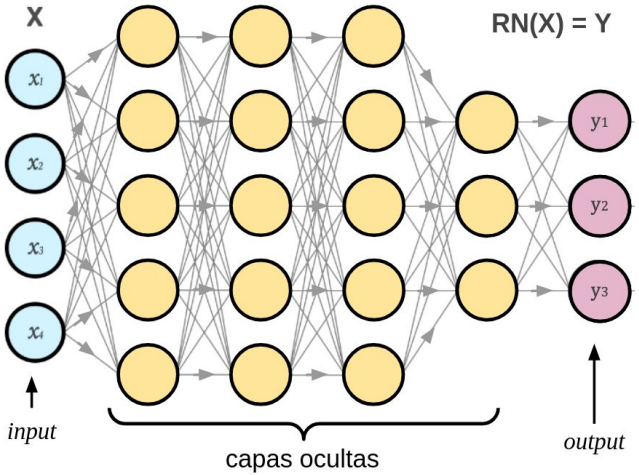
$$O_j = \sigma(\sum_{i=1}^{n} w_{ij}a_i + b_j)$$

Función de activación $\sigma(z)$
y bias



x ⟶ **f(x) = y**

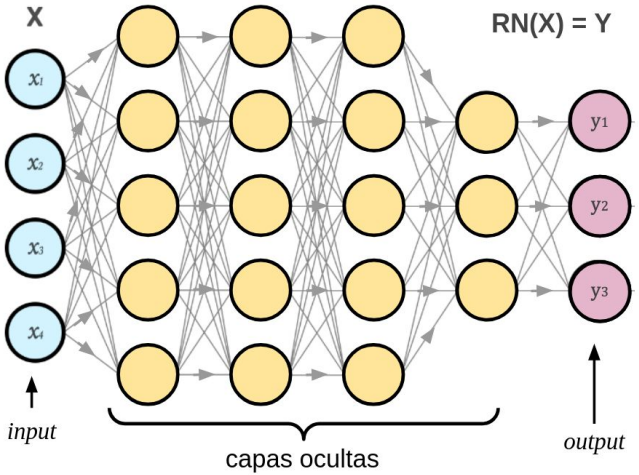
queremos aproximar **f** por una **Red Neuronal (RN)**



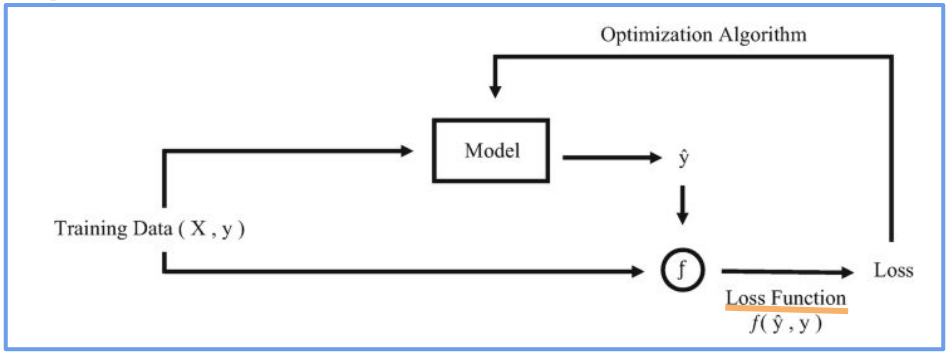
Supervised Learning

$x \longrightarrow f(x) = y$

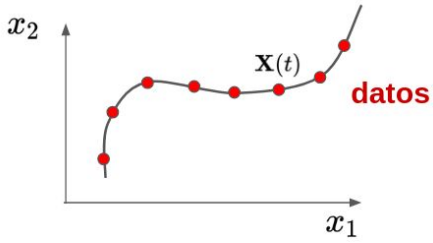
queremos aproximar **f** por una **Red Neuronal (RN)**



epoch



X - Input data
y - Expected output
ŷ - Predicted output
f - Loss function



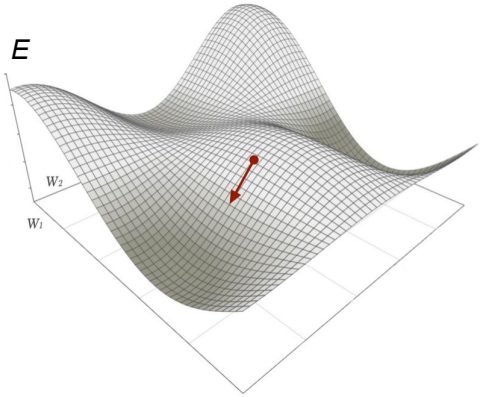
$$E = \frac{1}{N_d} \sum_{i=1}^{N_d} |\mathbf{f}(\mathbf{x}_i^d) - \mathbf{RN}(\mathbf{x}_i^d, \mathbf{W})|^2$$

función de costo

Descenso por el gradiente

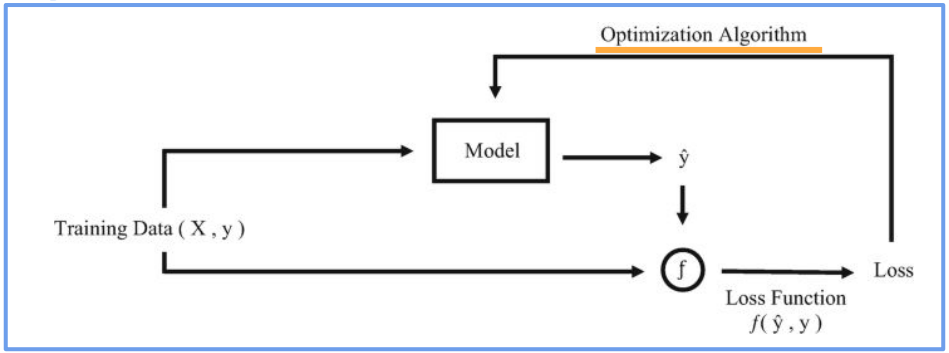
learning rate

$$\mathbf{W}_{\text{nuevo}} = \mathbf{W} - \eta \nabla_{\mathbf{w}} E$$



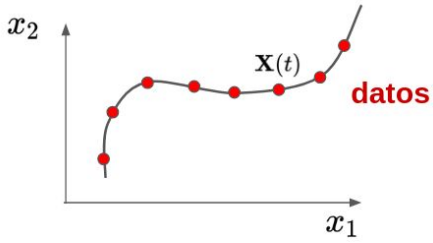
Backpropagation optimizar esto usando **propiedades** de las **funciones de activación y regla de la cadena**

epoch



X - Input data
y - Expected output

ŷ - Predicted output
f - Loss function



$$E = \frac{1}{N_d} \sum_{i=1}^{N_d} |\mathbf{f}(\mathbf{x}_i^d) - \mathbf{RN}(\mathbf{x}_i^d, \mathbf{W})|^2$$

función de costo

Physical Informed Neural Networks (PINNs)

$$\frac{\partial \mathbf{X}}{\partial t} - \mathbf{u}(\mathbf{X}, t) = 0$$

Error de los datos

$$\mathrm{MSE}_d = \frac{1}{N_d} \sum_{i=1}^{N_d} \left| \mathbf{X}(t_i^d) - \mathbf{RN}(t_i^d, \mathbf{W}) \right|^2$$

Error de la física

$$\mathrm{MSE}_f = \frac{1}{N_f} \sum_{j=1}^{N_f} \left| \frac{\partial \mathbf{RN}}{\partial t}\left(t_j^f, \mathbf{W}\right) - \mathbf{u}\left(\mathbf{RN}(t_j^f, \mathbf{W}), t_j^f\right) \right|^2$$

$$E = \mathrm{MSE}_d + \lambda \mathrm{MSE}_f$$

función de costo

PyTorch

$$x = (x_0, x_1, \ldots, x_N)$$
$$y = (y_0, y_1, \ldots, y_N)$$

$$\mathbf{X} = [\,[x1, y1],\; [x2, y2],\; [x3, y3],\; \ldots\; [xn, yn]\,]$$