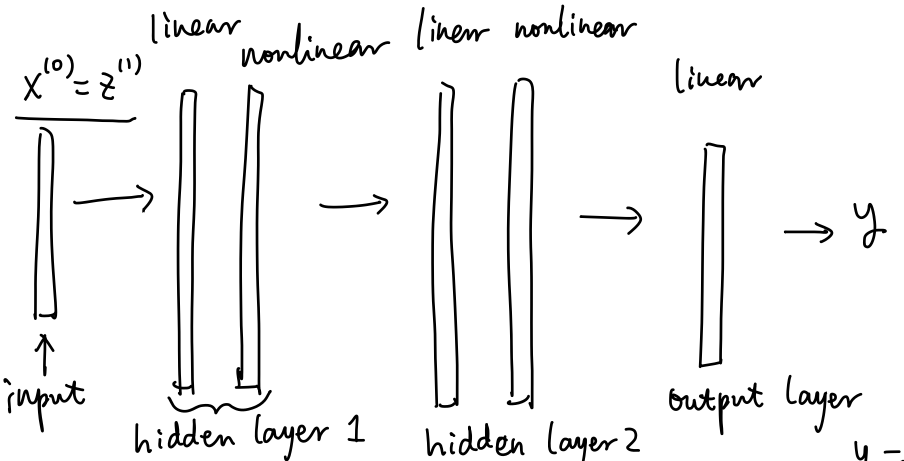
Review:

Feedforward neural network (FNN):



$$z^{(2)} = \sigma\left(W^{(1)} z^{(1)} + b^{(1)}\right) \qquad z^{(3)} = \sigma\left(W^{(2)} z^{(2)} + b^{(2)}\right) \qquad y = W^{(3)} z^{(3)} + b^{(3)}$$

example: take FNN with 1 hidden layer. activation function

$$\sigma(x) = \mathrm{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \le 0 \end{cases} = \max\{x, 0\}$$

$$W^{(1)} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}. \qquad b^{(1)} = \begin{pmatrix} -2 \\ 0 \end{pmatrix}$$

output layer: $W^{(2)} = (1, 1)$. $b^{(2)} = 0$

input: $x^{(0)} = z^{(1)}$

hidden layer:
$$\begin{aligned} z^{(2)} &= \sigma\left(W^{(1)} z^{(1)} + b^{(1)}\right) \\ &= \sigma\left(\begin{pmatrix} 2 \\ 1 \end{pmatrix} z^{(1)} + \begin{pmatrix} -2 \\ 0 \end{pmatrix}\right) \\ &= \sigma\left(\begin{pmatrix} 2z^{(1)} - 2 \\ z^{(1)} \end{pmatrix}\right) \\ &= \mathrm{ReLU}\left(\begin{pmatrix} 2z^{(1)} - 2 \\ z^{(1)} \end{pmatrix}\right) = \begin{pmatrix} \mathrm{ReLU}(2z^{(1)} - 2) \\ \mathrm{ReLU}(z^{(1)}) \end{pmatrix} \end{aligned}$$

output layer: $y = W^{(2)} z^{(2)} + b^{(2)}$

$$= (1,1)\, z^{(2)} + 0$$

$$= (1,1) \begin{pmatrix} \mathrm{ReLU}(2z^{(1)}-2) \\ \mathrm{ReLU}(z^{(1)}) \end{pmatrix} + 0$$

$$= \mathrm{ReLU}(2z^{(1)}-2) + \mathrm{ReLU}(z^{(1)})$$

FNN: $y = \mathrm{ReLU}(2x^{(0)}-2) + \mathrm{ReLU}(x^{(0)})$

$$\underline{y = \max\{2x^{(0)}-2,\ 0\} + \max\{x^{(0)},\ 0\}}$$

(1) if $x<0$.

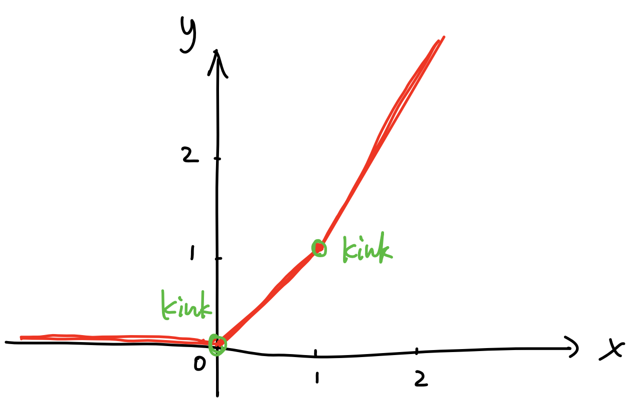
$2x-2<0$. $x<0$. $\Rightarrow$ $y = 0+0 = 0$

(2) if $0<x<1$.

$2x-2<0$. $x>0$ $\Rightarrow$ $y = 0 + x = x$

(3) if $x>1$.

$2x-2>0$. $x>0$ $\Rightarrow$ $y = 2x-2+x = 3x-2$



Remark: (1) FNN is a function

(2) The function in this example has two kinks

(3) FNN with 1 hidden layer with 2 neurons : two kinks

FNN with 2 hidden layer with 2 neurons : four kinks

<u>example</u>: FNN with 2 hidden layer with 2 neurons each layer <u>(HW)</u>

input: $x^{(0)} = z^{(1)}$

hidden layer1: $z^{(2)} = \sigma(W^{(1)} z^{(1)} + b^{(1)})$

hidden layer 2: $z^{(3)} = \sigma(W^{(2)} z^{(2)} + b^{(2)})$

output: $y = W^{(3)} z^{(3)} + b^{(3)}$

expressivity of neural network is increasing when we use more hidden layers and more neurons

<u>Definition</u> (FNN): An FNN is a function $f: \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$. For an input $x \in \mathbb{R}^{d_{in}}$, the output $y \in \mathbb{R}^{d_{out}}$ is given by:

$$\begin{cases} z^{(1)} = x \\ z^{(i+1)} = \sigma(W^{(i)} z^{(i)} + b^{(i)}), \quad i = 1, 2, \cdots, L \\ y = W^{(L+1)} z^{(L+1)} + b^{(L+1)} \end{cases}$$

Here, we say that the number of hidden layers is L. The number of neurons in each hidden layer is denoted by H_i for $1 \leq i \leq L$.

Therefore, the parameters have the dimension:

$$W^{(1)} \in \mathbb{R}^{H_1 \times d_{in}}, \quad b^{(1)} \in \mathbb{R}^{H_1}$$

$$W^{(2)} \in \mathbb{R}^{H_2 \times H_1}, \quad b^{(2)} \in \mathbb{R}^{H_2}$$

$$\vdots$$

$W^{(L)} \in \mathbb{R}^{H_L \times H_{L-1}}$. $b^{(L)} \in \mathbb{R}^{H_L}$

$W^{(L+1)} \in \mathbb{R}^{d_{out} \times H_L}$. $b^{(L+1)} \in \mathbb{R}^{d_{out}}$

<u>Example:</u> An FNN with $d_{in} = d_{out} = 1$. 3 hidden layers.

5 neurons in each hidden layer ($H_1 = H_2 = H_3 = 5$)

total number of parameters in this FNN?

$W^{(1)} \in \mathbb{R}^{5\times 1}$. $b^{(1)} \in \mathbb{R}^5 \rightarrow 5 + 5 = 10$

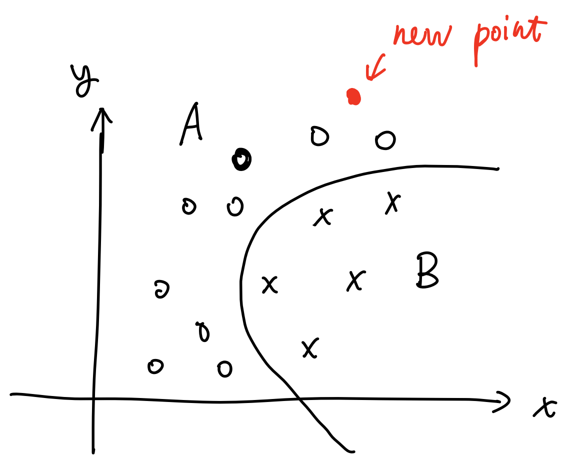
$W^{(2)} \in \mathbb{R}^{5\times 5}$. $b^{(2)} \in \mathbb{R}^5 \rightarrow 25 + 5 = 30$

$W^{(3)} \in \mathbb{R}^{5\times 5}$. $b^{(3)} \in \mathbb{R}^5 \rightarrow 25 + 5 = 30$

$W^{(4)} \in \mathbb{R}^{1\times 5}$. $b^{(4)} \in \mathbb{R}^1 \rightarrow 5 + 1 = 6$

total number: $10 + 30 + 30 + 6 = \underline{76}$

<u>classification problem:</u>



FNN: $y = \sigma(W^{(3)}\sigma(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}) + b^{(3)})$

Find out $(W^{(1)}, W^{(2)}, W^{(3)}, b^{(1)}, b^{(2)}, b^{(3)})$. such that

$$y(x^{(i)}) = \begin{cases} (1, 0), & \text{if } x^{(i)} \in A \\ (0, 1) & \text{if } x^{(i)} \in B \end{cases}$$

Define the Loss function (or called cost function):

$$L(W^{(1)}, W^{(2)}, W^{(3)}, b^{(1)}, b^{(2)}, b^{(3)}) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2}\| y^{(i)} - y(x^{(i)}) \|_2^2$$

Training neural network (optimization problem):

Find $W^{(1)}, W^{(2)}, W^{(3)}, b^{(1)}, b^{(2)}, b^{(3)}$, such that the loss funtion is minimized.

$f(x)$ take the min value at $x = x^*$.

$\downarrow$

Optimization: minimizing $f(x)$. $x^* = \arg\min f(x)$

example: $f(x) = x^2$. take the min at $x = 0$.

Gradient descent: find out min of $f(x)$

$$f(x+\Delta x) = f(x) + \Delta x \cdot f'(x) + O(\Delta x^2) \leftarrow \text{Taylar expansion}$$

$$\approx f(x) + \Delta x \cdot f'(x)$$

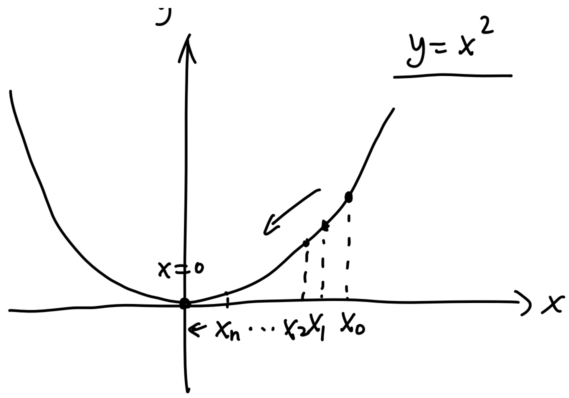
take $\Delta x = -\underline{\eta} \cdot f'(x)$. $\eta > 0$

$$f(x+\Delta x) \approx f(x) - \eta \cdot f'(x) \cdot f'(x)$$

$$= f(x) - \eta \cdot (f'(x))^2 \leqslant f(x)$$

$$f(\underbrace{x - \eta \cdot f'(x)}) \leqslant f(x)$$

$$\shortparallel$$



$$x_1 = x_0 - \eta \cdot f'(x_0)$$
$$= x_0 - \eta \cdot 2x_0$$
$$x_2 = x_1 - \eta \cdot f'(x_1)$$

take the initial guess $x = x_0$. do the iteration

$$x_{n+1} = x_n - \eta \cdot f'(x_n).$$

η ← learning rate — small positive constant

generalization to vector case:

$$f(x+\Delta x) = f(x) + \underbrace{\Delta x}_{\text{vector}} \cdot \underbrace{\nabla f(x)}_{\text{vector}} + O(\Delta x^2)$$
$$\approx f(x) + \Delta x \cdot \nabla f(x)$$

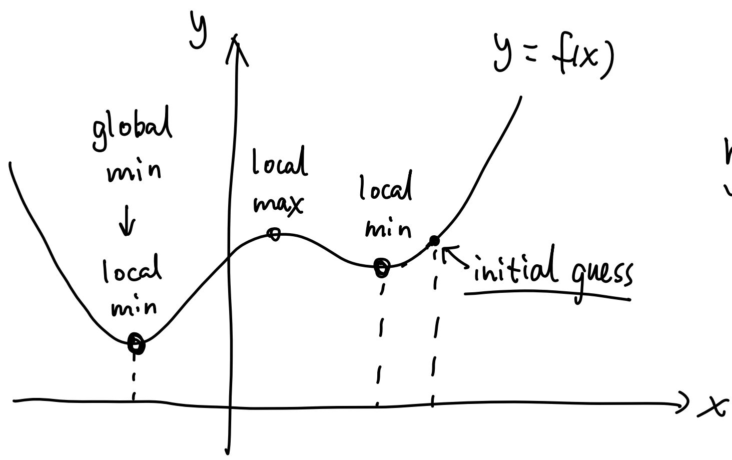
take $\Delta x = -\eta \nabla f(x)$. η small positive constant

$$f(x+\Delta x) \approx f(x) + \left(-\eta \nabla f(x) \cdot \nabla f(x)\right)$$
$$= f(x) - \eta \left|\nabla f(x)\right|^2 \leq f(x)$$

$$f(x - \eta \nabla f(x)) \leq f(x)$$

take the initial guess $x = x_0$. do the iteration

$$x_{n+1} = x_n - \eta \nabla f(x_n)$$



min f(x)

critical point : $f'(x) = 0$

local min : critical point x_0 + $f''(x_0) > 0$

global min : min in the local min

gradient decent may not always find global minimum (HW)