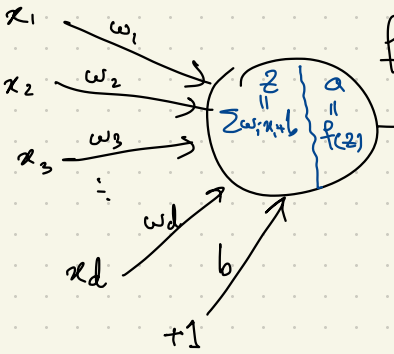
# Neural Networks

Building block : neuron




$$f\left(\sum_{i=1}^{d} w_i x_i + b\right) = a$$

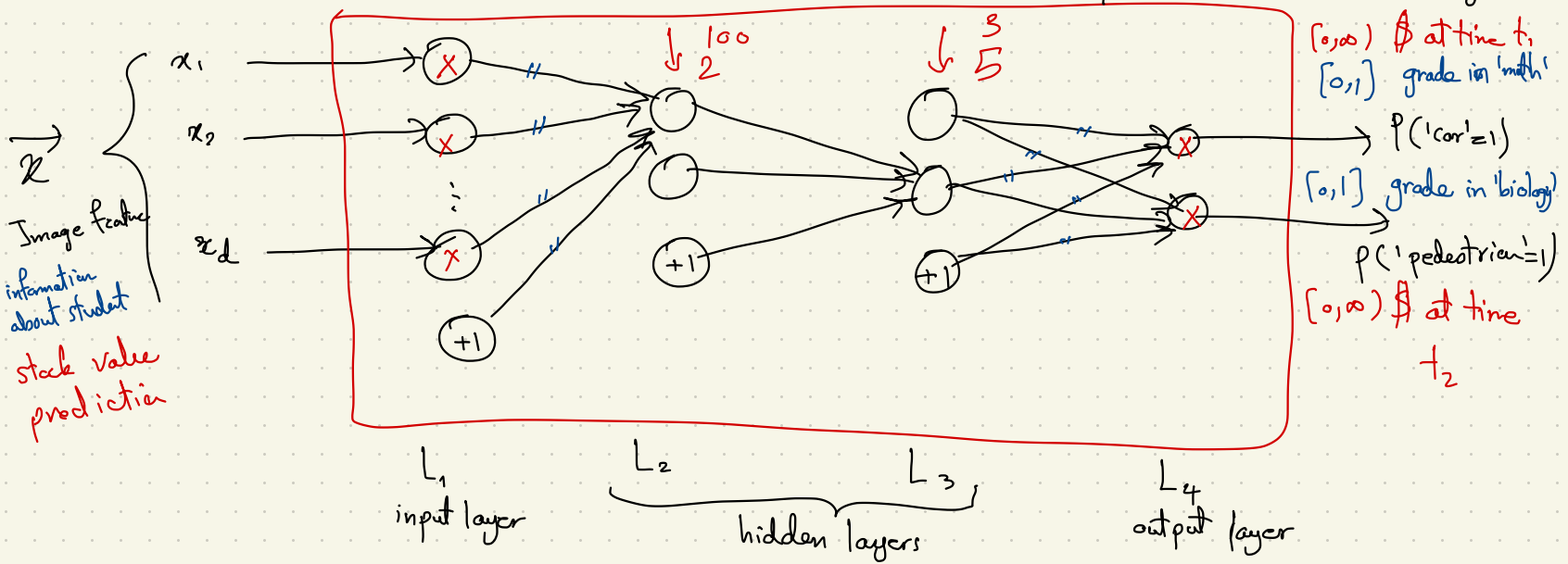
$z$ = total weighted input

$a$ = activation / output of neuron

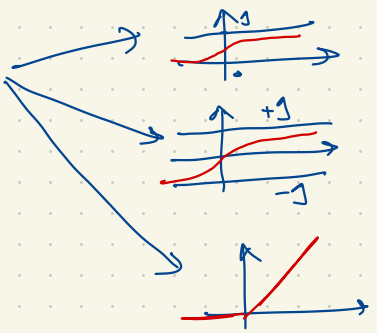
## Feedforward Neural Networks :

Shallow NN ← 1 hidden

Deep NN ← ≥ 2 hidden layers



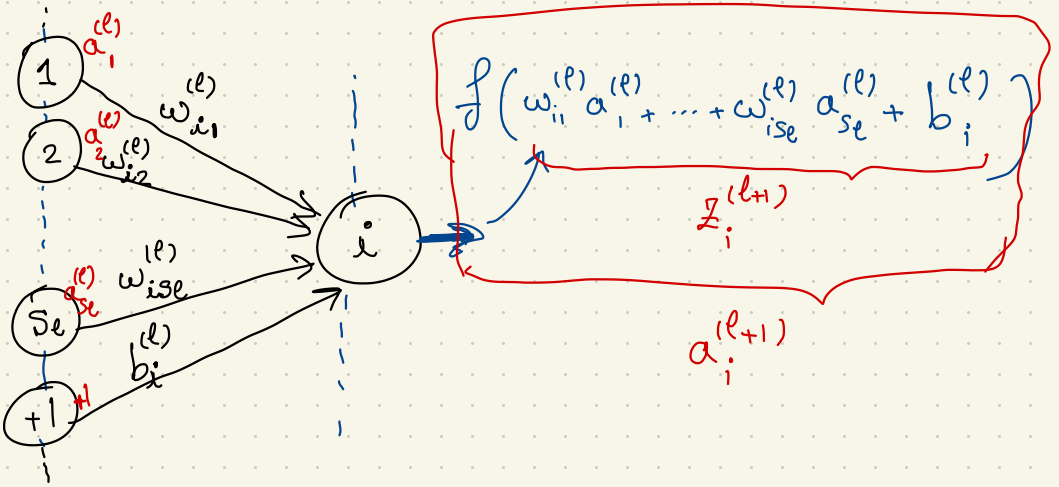
$f_{(\cdot)}$ = activation function

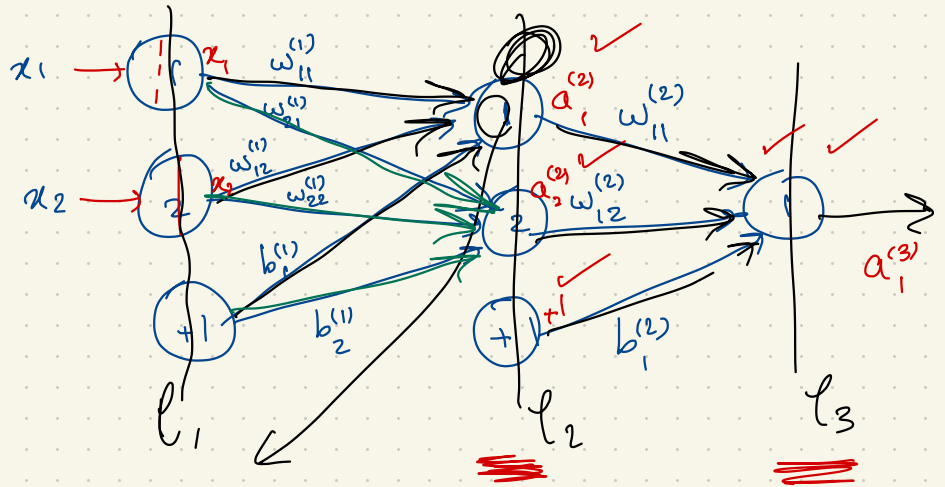



Sigmoid ⇒ grade $[0,1]$

tanh ⇒ class $\pm 1$

Relu ⇒ stock value $






$$z_1^{(1)} = a_1^{(1)} = x_1$$

$$z_2^{(1)} = a_2^{(1)} = x_2$$

$$\begin{cases} z_1^{(2)} = w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + b_1^{(1)} \\ a_1^{(2)} = f(z_1^{(2)}) \end{cases}$$

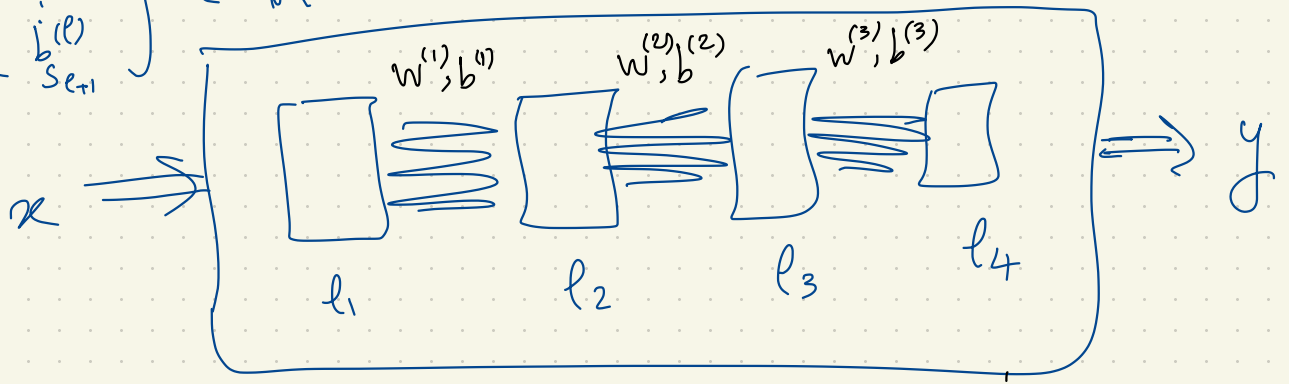
$$\begin{cases} z_2^{(2)} = w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + b_2^{(1)} \\ a_2^{(2)} = f(z_2^{(2)}) \end{cases}$$

$$\begin{cases} z_1^{(3)} = w_{11}^{(2)} a_1^{(2)} + w_{12}^{(2)} a_2^{(2)} + b_1^{(2)} \\ a_1^{(3)} = f(z_1^{(3)}) \end{cases}$$

$$W^{(\ell)} = \begin{bmatrix} w_{11}^{(\ell)} & w_{12}^{(\ell)} & \cdots & w_{1s_\ell}^{(\ell)} \\ \vdots & \vdots & & \vdots \\ w_{s_{\ell+1}1}^{(\ell)} & w_{s_{\ell+1}2}^{(\ell)} & \cdots & w_{s_{\ell+1}s_\ell}^{(\ell)} \end{bmatrix} \in \mathbb{R}^{s_{\ell+1} \times s_\ell}$$

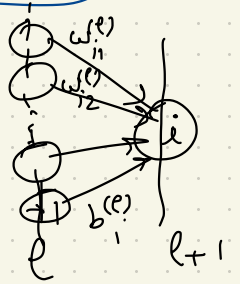
$s_\ell$ = # nodes in layer $\ell$ except $(+1)$.

$$b^{(\ell)} = \begin{bmatrix} b_1^{(\ell)} \\ \vdots \\ b_{s_{\ell+1}}^{(\ell)} \end{bmatrix} \in \mathbb{R}^{s_{\ell+1}}$$




$$z_i^{(\ell+1)} = \sum_{j=1}^{s_\ell} w_{ij}^{(\ell)} a_j^{(\ell)} + b_i^{(\ell)}$$

$$z_i^{(\ell+1)} = \begin{bmatrix} w_{i1}^{(\ell)} & w_{i2}^{(\ell)} & \cdots & w_{is_\ell}^{(\ell)} \end{bmatrix} \begin{bmatrix} a_1^{(\ell)} \\ \vdots \\ a_{s_\ell}^{(\ell)} \end{bmatrix} + b_i^{(\ell)}$$



$$\begin{bmatrix} z_1^{(l+1)} \\ \vdots \\ z_i^{(l+1)} \\ \vdots \\ z_{s_{l+1}}^{(l+1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(l)} & w_{12}^{(l)} & \cdots & w_{1s_l}^{(l)} \\ \vdots & & & \\ w_{i1}^{(l)} & w_{i2}^{(l)} & \cdots & w_{is_l}^{(l)} \\ \vdots & & & \\ w_{s_{l+1}1}^{(l)} & w_{s_{l+1}2}^{(l)} & \cdots & w_{s_{l+1}s_l}^{(l)} \end{bmatrix} \begin{bmatrix} a_1^{(l)} \\ \vdots \\ a_{s_l}^{(l)} \end{bmatrix} + \begin{bmatrix} b_1^{(l)} \\ \vdots \\ b_i^{(l)} \\ \vdots \\ b_{s_{l+1}}^{(l)} \end{bmatrix}$$

$z^{(l+1)} \in \mathbb{R}^{s_{l+1}}$ $\quad W^{(l)} \in \mathbb{R}^{s_{l+1} \times s_l}$ $\quad a^{(l)} \in \mathbb{R}^{s_l}$ $\quad b^{(l)} \in \mathbb{R}^{s_{l+1}}$

Forward Propagation

$$\vec{z}^{(l+1)} = W^{(l)} \vec{a}^{(l)} + \vec{b}^{(l)}$$

$$\vec{a}^{(l+1)} = f(\vec{z}^{(l+1)})$$

$$\vec{z}^{(1)} = \vec{a}^{(1)} = \vec{x}$$

$l = 1 \rightarrow (z^{(2)}, a^{(2)})$

$l = 2 \rightarrow (z^{(3)}, a^{(3)})$

$\vdots$

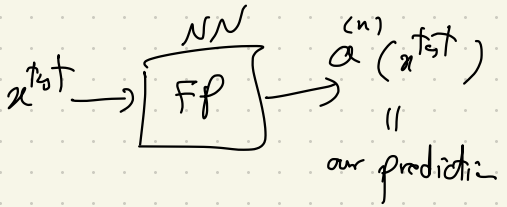
$a^{(n)} = \text{output}$

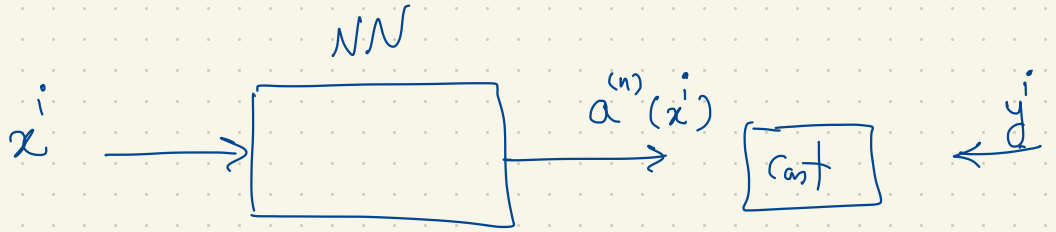
We learn weights and biases $\{W^{(\ell)}, b^{(\ell)}\}_{\ell=1}^{n-1}$ using training data.

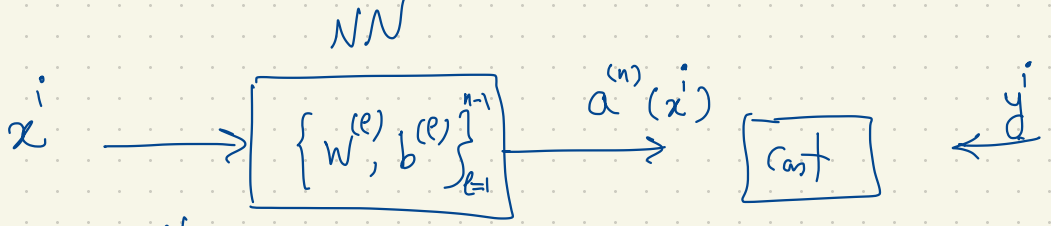
$D = \{(x^i, y^i)\}_{i=1}^{N}$ training data $(x^i \in \mathbb{R}^d, y^i \in \mathbb{R}^m)$

$\begin{pmatrix} \text{car} \\ \text{pedestria} \end{pmatrix} \in \mathbb{R}^2$ $\begin{pmatrix} 10 \\ 11 \\ \vdots \\ 9 \end{pmatrix} \in \mathbb{R}^{10}$

Once learned, classification of a test sample $x^{test}$ is just applying FP to $x^{test}$

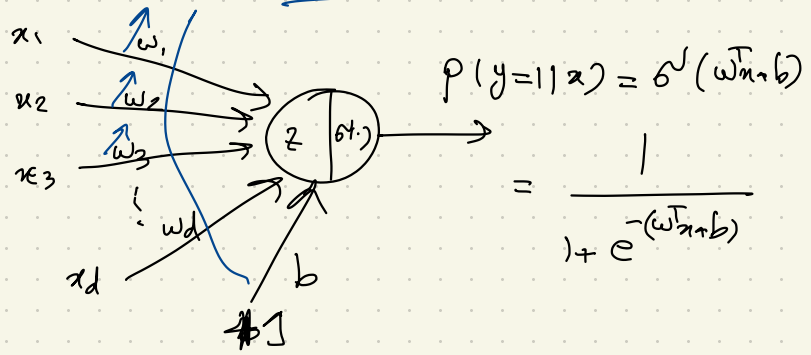


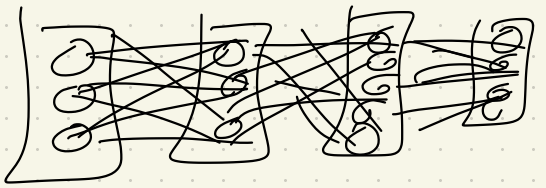


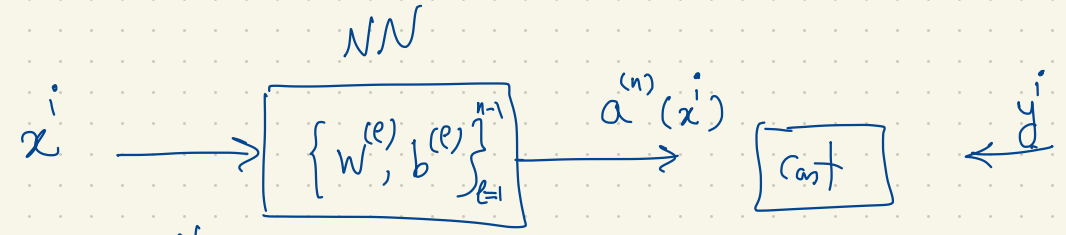

$$\min_{\{W^{(\ell)}, b^{(\ell)}\}_{\ell=1}^{n-1}} \frac{1}{N} \sum_{i=1}^{N} \text{cost}\left(a^{(n)}(x^i), y^i\right) + \lambda \text{ regularizer}$$

Logistic regression $(+\lambda \|w\|_2^2)$




Simplest NN with only 1 neuron whose activation is sigmoid func.






$$\min_{\{W^{(\ell)},b^{(\ell)}\}_{\ell=1}^{n-1}} \frac{1}{N}\sum_{i=1}^{N} \text{cost}\left(a^{(n)}(x^i), y^i\right) + \lambda \text{ regularizer}$$

$$\min_{\{W^{(\ell)},b^{(\ell)}\}_{\ell=1}^{n-1}} \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2}\left\| a^{(n)}(x^i) - y^i \right\|_2^2 + \frac{\lambda}{2}\sum_{\ell=1}^{n-1} \left\| W^{(\ell)} \right\|_F^2$$

$\Downarrow$

train a NN

Frobenius norm

$$\|A\|_F^2 \triangleq \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2$$

$A$: $m \times n$

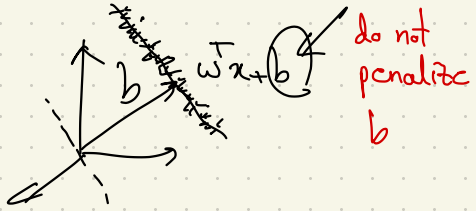
$$\min_{\{w^{(l)},b^{(l)}\}_{l=1}^{n-1}} \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left\| a^{(n)}(x^i) - y^i \right\|_2^2 + \frac{\lambda}{2}\sum_{l=1}^{n-1}\left\| w^{(l)} \right\|_F^2$$

Cost

Solve via G.D.

$$\frac{\partial \text{cost}}{\partial w^{(l)}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \frac{1}{2}\left\| a^{(n)}(x^i) - y^i \right\|_2^2}{\partial w^{(l)}} + \lambda w^{(l)}$$

$$\frac{\partial \text{Cost}}{\partial b^{(l)}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \frac{1}{2}\left\| a^{(n)}(x^i) - y^i \right\|_2^2}{\partial b^{(l)}}$$




do not penalize b

* Initialize $\{W^{(\ell)}, b^{(\ell)}\}_{\ell=1}^{n-1}$ ( e.g. $\mathcal{N}(0, \varepsilon^2)$, $\varepsilon \simeq 0.01$ )

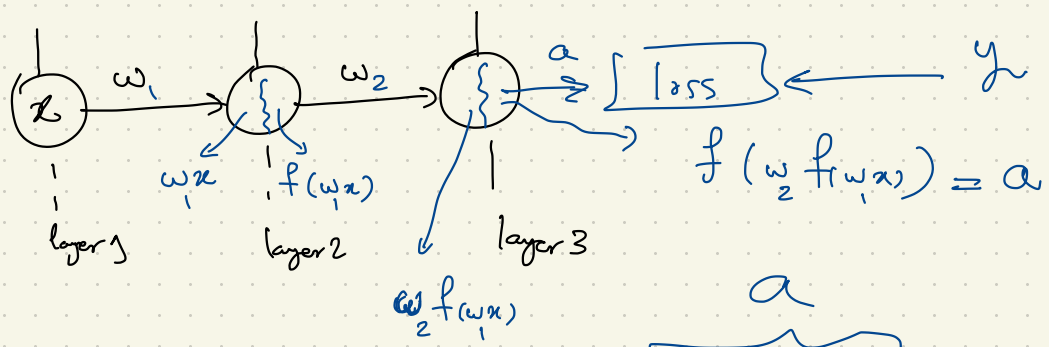
* For $t = 1, 2, \ldots, T$ ( until convergence )

$$\begin{cases} W^{(\ell)} \leftarrow W^{(\ell)} - \eta \dfrac{\partial \text{cost}}{\partial W^{(\ell)}} \\ b^{(\ell)} \leftarrow b^{(\ell)} - \eta \dfrac{\partial \text{cost}}{\partial b^{(\ell)}} \end{cases} \quad \forall \ell = 1, \ldots, n-1$$

$$\left\{ W^{*(\ell)}, b^{*(\ell)} \right\}$$

NNs have a mechanism to efficiently compute derivative wrt weights and biases : **Back propagation**

$$\left\{ \frac{\partial \text{cost}}{\partial W^{(n-1)}} \atop \frac{\partial \text{cost}}{\partial b^{(n-1)}} \right\} \longrightarrow \left\{ \frac{\partial \text{cost}}{\partial W^{(n-2)}} \atop \frac{\partial \text{cost}}{\partial b^{(n-2)}} \right\} \longrightarrow \cdots \longrightarrow \left\{ \frac{\partial \text{cost}}{\partial W^{(1)}} \atop \frac{\partial \text{cost}}{\partial b^{(1)}} \right\}$$

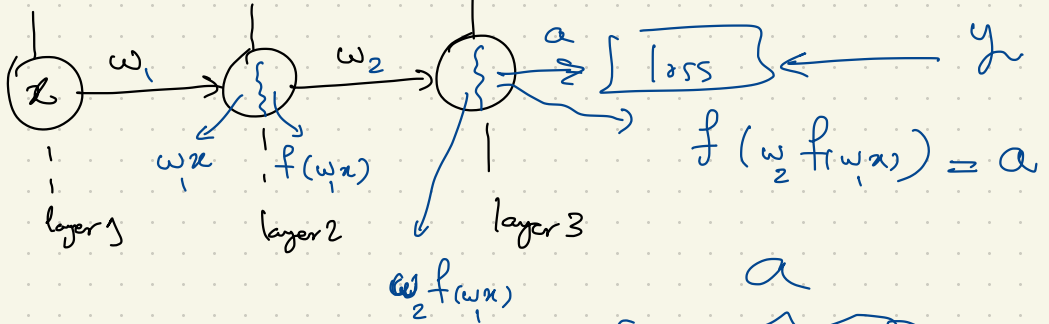



$$\text{Cost} = \frac{1}{2}(a-y)^2 = \frac{1}{2}\left(\overbrace{f(w_2 f(w_1 x))}^{a} - y\right)^2$$

To compute $\frac{\partial \text{cost}}{\partial w_1 \text{ or } w_2}$ need to compute $\frac{\partial a}{\partial w_1 \text{ or } w_2}$

$$\frac{\partial \text{Cost}}{\partial w_1} = \frac{\partial \text{Cost}}{\partial a} \times \frac{\partial a}{\partial w_1}$$

$$= \frac{\partial \frac{1}{2}(a-y)^2}{\partial a} \times \frac{\partial a}{\partial w_1} = (a-y) \times \frac{\partial a}{\partial w_1}$$




$$\text{Cost} = \frac{1}{2}(a-y)^2 = \frac{1}{2}\left(\overbrace{f(w_2 f(w_1 x))}^{a} - y\right)^2$$

$$\frac{\partial a}{\partial w_2} = \frac{\partial f(w_2 f(w_1 x))}{\partial w_2} = \alpha f'(w_2 \alpha)$$

$$= f(w_1 x) \times f'(w_2 f(w_1 x))$$

$$\frac{\partial a}{\partial w_1} = \frac{\partial a}{\partial (w_2 f(w_1 x))} \times \frac{\partial (w_2 f(w_1 x))}{\partial w_1}$$

$$= f'(w_2 f(w_1 x)) \times w_2 \times \frac{\partial f(w_1 x)}{\partial w_1}$$

$$x \times f'(w_1 x)$$

Pytorch : .Backward ()


classification
or
regression

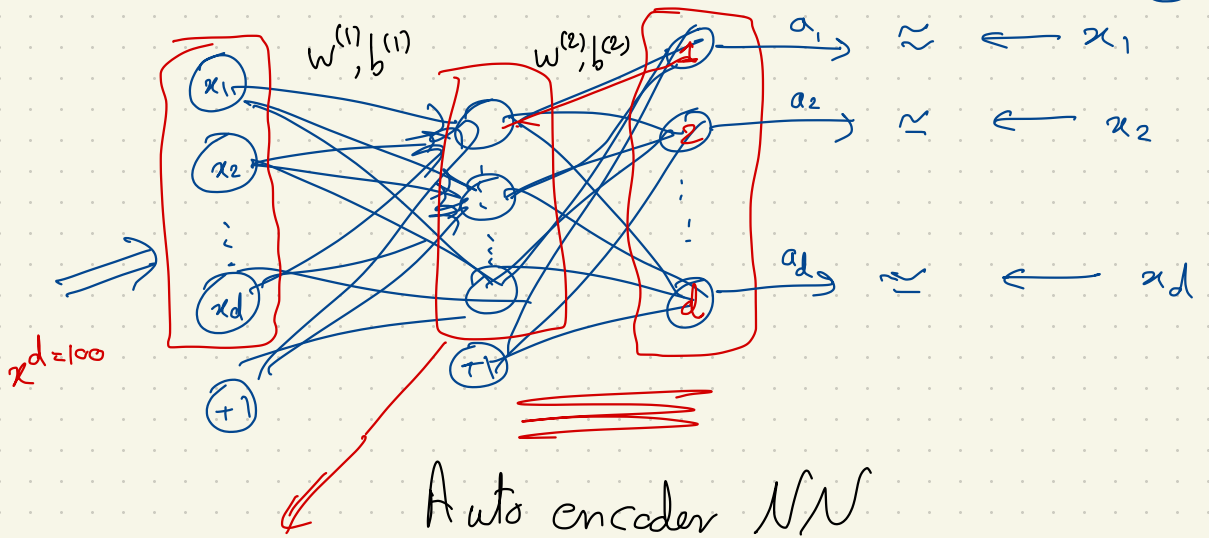
$f(z) =$ Relu, Identity $\Rightarrow$ regression

$f(z) =$ Sigmoide / tanh $\Rightarrow$ $p('car'=1 \mid x) \in [0,1]$

[0,1] [-1,1] $\in [-1,+1]$

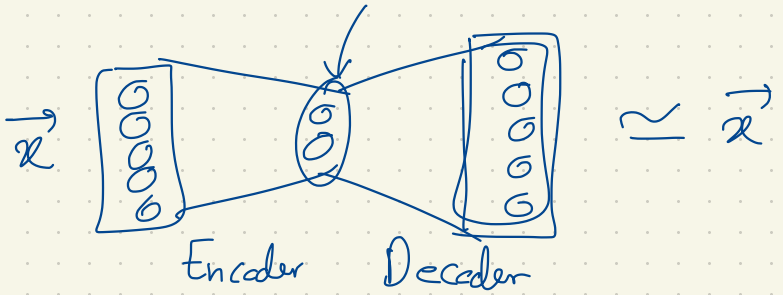
Classification

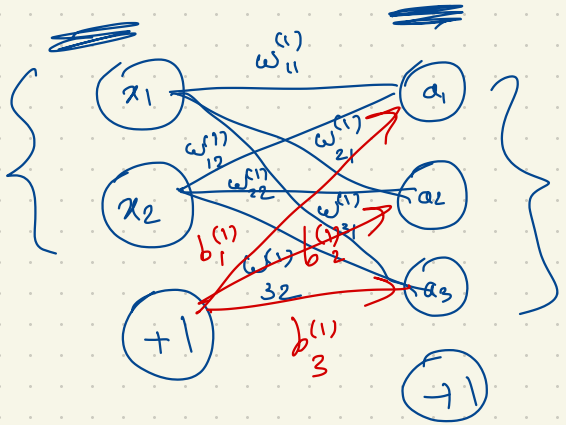
NNs can be used to reduce data dimension (dimensionality reduction)




Auto encoder NN

# nodes = 5






$l_1$ $\qquad$ $l_2$

$S_{l_1} = 2$ $\qquad$ $S_{l_2} = 3$

$$W^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} \end{bmatrix}$$

$3 \times 2$

$S_{l_2} \times S_{l_1}$

$$b^{(1)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \end{bmatrix}$$

$3 \times 1$

$S_{l_2}$

$$\begin{cases} W^{(l)} \in \mathbb{R}^{S_{l+1} \times S_l} \\ b^{(l)} \in \mathbb{R}^{S_{l+1}} \end{cases}$$

How to select architecture ?