Computation graphs and backpropagation

software represent our neural net equations as graph

$s = u^T h$

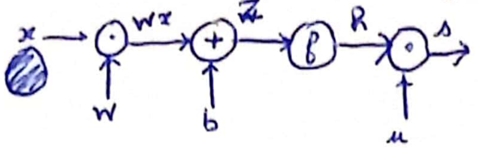
$h = f(z)$

$z = Wx + b$

x : input

source node : inputs

interior nodes : operations

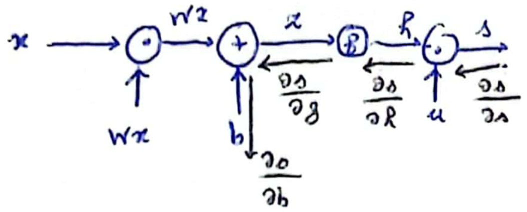


this is forward propagation

we want also to send back gradients to update the parameters of the model

$\Downarrow$

backpropagation



1 - F - propag:

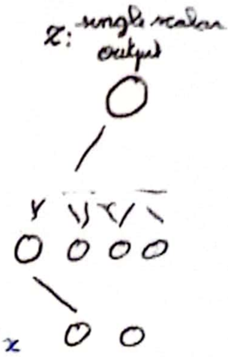
visit nodes in topological sort order

- compute value of node given predecessor

2 - back - propag

- initialize output gradient = 1
- visit nodes in reverse order

compute gradient of each node using gradient wrt successors



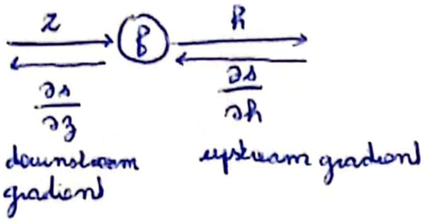
$\{y_1, y_2, \dots, y_n\}$: successors of x

$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \cdot \frac{\partial y_i}{\partial x}$$

backpropagation: single node

1/ input

$$h = f(z)$$

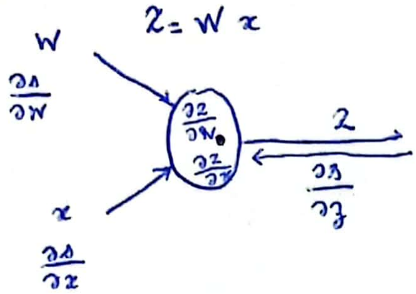


⇒ we need to workout $\frac{\partial h}{\partial z}$

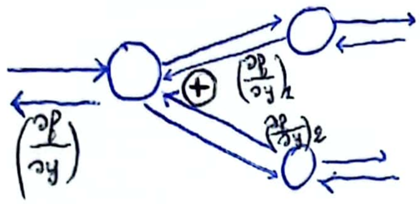
$$\frac{\partial s}{\partial z} = \frac{\partial s}{\partial h} \cdot \frac{\partial h}{\partial z}$$

2/ multiple inputs

$$z = Wx$$



3/ multiple output branches



$$\left(\frac{\partial f}{\partial y}\right) = \left(\frac{\partial f}{\partial y}\right)_1 + \left(\frac{\partial f}{\partial y}\right)_2$$

⚠ incorrect way to compute backpropagation is to compute gradients one by one

⇒ we need to compute all gradients at once

Math for neural networks

$\boxed{\text{math 51 or Engr 108 : matrix calculus}}$

$$z = 3y^2 \qquad \frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx} = 6x$$
$$y = x^2$$

$$h = f(z) \qquad \Rightarrow \frac{\partial h}{\partial x} = \frac{\partial h}{\partial z} \cdot \frac{\partial z}{\partial x}$$
$$z = Wx + b$$

$$h = f(z) \Rightarrow \frac{\partial h}{\partial z}$$

$$\text{if } z, h \in \mathbb{R}^n \Rightarrow \frac{\partial h}{\partial z} \in \mathbb{R}_{n,n} \ (\mathbb{R})$$

$$\left(\frac{\partial h}{\partial z}\right)_{ij} = \frac{\partial h_i}{\partial z_j}$$

1/ element wise transformations

like logistic transform

$$h = f(z) \Rightarrow h_i = f(z_i)$$

$$\left(\frac{\partial h}{\partial z}\right)_{ij} = \frac{\partial h_i}{\partial z_j} = \frac{\partial}{\partial z_j} f(z_i)$$
$$= \begin{cases} f'(z_i) & \text{if } i = j \\ 0 & \text{else} \end{cases}$$

$$\frac{\partial h}{\partial z} = \begin{pmatrix} f'(z_1) & & (0) \\ & \ddots & \\ (0) & & f(z_n) \end{pmatrix}$$

$$\left.\begin{aligned} &x \in \mathbb{R}^m \\ &W \in M_{n,m}(\mathbb{R}) \\ &b \in \mathbb{R}^n \end{aligned}\right\} z \in \mathbb{R}^n$$

$$\left.\begin{aligned} &\text{elementwise element} \\ &\text{transformation} \end{aligned}\right\} h \in \mathbb{R}^n \qquad h = f(z)$$

$$u \in \mathbb{R}^n \Rightarrow s \in \mathbb{R}$$

$$2/ \ \frac{\partial}{\partial x}(Wx + b) = W$$

$$3/ \ \frac{\partial}{\partial b}(Wx + b) = I$$

$$4/ \ \frac{\partial}{\partial u}(u^T h) = h^T \qquad \frac{\partial}{\partial h}(u^T h) = u^T$$

$$s = u^T h$$
$$h = f(Wx + b) = f(z) \qquad z = Wx + b$$
$$x = \text{input}$$

$$\frac{\partial s}{\partial b} = \frac{\partial s}{\partial h} \cdot \frac{\partial h}{\partial z} \cdot \frac{\partial z}{\partial b}$$
$$= u^T \cdot \text{diag}(f'(z)) \cdot I$$
$$= u^T \circ (f'(z))$$

Hadamard product

$$W \in \mathbb{R}^{n \times m} \Rightarrow \frac{\partial s}{\partial W} : 1 \text{ by } n \times m \text{ jacobian}$$

it is a huge row vector

$$\theta^{new} = \theta^{old} - \alpha \cdot \nabla_\theta J(\theta)$$

inconvenient to do so if we have a row vector

$\Rightarrow$ we apply the shape convention

$1 \times n \times n, n \times n, nm$

~~[illegible]~~

$W \in M_{n,m}(\mathbb{R})$ $\quad s \in \mathbb{R}$

$\Rightarrow \frac{\partial s}{\partial W} \in M_{1,nm}(\mathbb{R})$: row vector

$\Rightarrow$ we can't calculate $\theta^{new} = \theta^{old} - \alpha \cdot \nabla_\theta J(\theta)$

$\Rightarrow$ we leave pure math and we use the $\boxed{\text{shape convention}}$

the shape of the gradient is the shape of the parameters

$\Rightarrow W \in M_{n,m}(\mathbb{R}) \Rightarrow \frac{\partial s}{\partial W} \in M_{n,m}(\mathbb{R})$

$$\frac{\partial s}{\partial W} = \begin{bmatrix} \frac{\partial s}{\partial W_{11}} & & \frac{\partial s}{\partial W_{1m}} \\ \vdots & & \\ \frac{\partial s}{\partial W_{n1}} & \cdots & \frac{\partial s}{\partial W_{nm}} \end{bmatrix}$$

$$\frac{\partial s}{\partial W} = \underbrace{\frac{\partial s}{\partial h} \cdot \frac{\partial h}{\partial z}}_{\delta = u^T \cdot \operatorname{diag}(f'(z))} \cdot \frac{\partial z}{\partial W} \qquad 1,n \times n,nm$$

$$= \delta \cdot \frac{\partial z}{\partial W} \longrightarrow \in M_{n,nm}(\mathbb{R})$$

ans.

the answer is $\boxed{\left(\frac{\partial s}{\partial W}\right)_{\text{convention}} = \delta^T x^T}$ why the transpose?

$$\frac{\partial s}{\partial W} = \delta \cdot \frac{\partial}{\partial W}(Wx + b)$$

for a single weight W_{ij}

$$\frac{\partial z_i}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} W_i x + b_i$$

$$= \frac{\partial}{\partial W_{ij}} \sum_{i=1}^{d} W_{ik} x_k = x_j$$

$\hookrightarrow$ we want $\frac{\partial s}{\partial W} \in M_{n,m}(\mathbb{R})$ and not $M_{1,nm}(\mathbb{R})$

$\Rightarrow \delta \in M_{1,n}(\mathbb{R}) \quad x \in M_{m,1}(\mathbb{R})$

$\Rightarrow$ by transposing both we obtain the result

shape conventions

- $\frac{\partial s}{\partial b} = u^T \circ f'(z)$ ~~is~~ is a row vector

shape convention says that our gradient should have the same shape of $b \Rightarrow$ a column vector

$$\left(\frac{\partial s}{\partial b}\right)_{\text{convention}} = \left(u^T \circ f'(z)\right)^T$$

Jacobian form is useful to do calculus

$\Downarrow$

To do so ?

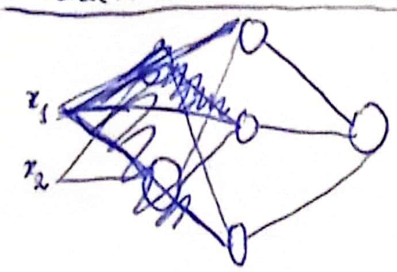
1/ use jacobian form as much as possible, reshape to follow the shape convention at the end (what we did for $\frac{\partial s}{\partial b}$)

2/ always follow the shape convention

Look at dimensions to figure out when to transpose and/or reorder terms

$\Downarrow$

Reseau de neuronnes à 2 couches



$$X = \begin{bmatrix} x_1^{(1)} & \cdots & x_1^{(m)} \\ x_2^{(1)} & \cdots & x_2^{(m)} \end{bmatrix}$$

m : nombre d'observations

ici on a 2 variables : $n^{[0]} = 2$

$$Z^{[1]} = W^{[1]} \cdot X + b^{[1]}$$

$$A^{[1]} = \frac{1}{1+e^{-Z^{[1]}}}$$

$$W \in M_{n^{[1]}, n^{[0]}}$$

$$b \in M_{n^{[1]}, 1}$$

$$Z^{[1]}, b^{[1]} \in M_{n^{[1]}, m}$$

$$Z^{[2]} = W^{[2]} \cdot A^{[1]} + b^{[2]}$$

$$A^{[2]} = \frac{1}{1+e^{Z^{[2]}}}$$

Fonction cout dans ce cas

$$\mathcal{L} = -\frac{1}{m}\sum y \log(A^{[2]}) + (1-y)\log(1-A^{[2]})$$

Backpropagation

$$\frac{\partial \mathcal{L}}{\partial W^{[2]}} = \underbrace{\boxed{\frac{\partial \mathcal{L}}{\partial A^{[2]}} \cdot \frac{\partial A^{[2]}}{\partial Z^{[2]}}}}_{dZ_2} \cdot \frac{\partial Z^{[2]}}{\partial W^{[2]}}$$

$$\frac{\partial \mathcal{L}}{\partial b^{[2]}} = \boxed{\frac{\partial \mathcal{L}}{\partial A^{[2]}} \cdot \frac{\partial A^{[2]}}{\partial Z^{[2]}}} \cdot \frac{\partial Z^{[2]}}{\partial b^{[2]}}$$

$$\cancel{\frac{\partial \mathcal{L}}{\partial W^{[1]}} = \frac{\partial \mathcal{L}}{\partial A^{[1]}} \cdot \frac{\partial A^{[1]}}{\partial Z^{[1]}} \cdot \frac{\partial Z^{[1]}}{\partial W^{[1]}}}$$

$$\cancel{\frac{\partial \mathcal{L}}{\partial b^{[1]}} = \frac{\partial \mathcal{L}}{\partial A^{[1]}} \cdot \frac{\partial A^{[1]}}{\partial Z^{[1]}} \cdot \frac{\partial Z^{[1]}}{\partial b^{[1]}}}$$

$$\frac{\partial \mathcal{L}}{\partial W^{[1]}} = \underbrace{\boxed{\frac{\partial \mathcal{L}}{\partial A^{[2]}} \cdot \frac{\partial A^{[2]}}{\partial Z^{[2]}} \cdot \frac{\partial Z^{[2]}}{\partial A^{[1]}} \cdot \frac{\partial A^{[1]}}{\partial Z^{[1]}}}}_{dZ_1} \cdot \frac{\partial Z^{[1]}}{\partial W^{[1]}}$$

$$\frac{\partial \mathcal{L}}{\partial b^{[1]}} = \boxed{\frac{\partial \mathcal{L}}{\partial A^{[2]}} \cdot \frac{\partial A^{[2]}}{\partial Z^{[2]}} \cdot \frac{\partial Z^{[2]}}{\partial A^{[1]}} \cdot \frac{\partial A^{[1]}}{\partial Z^{[1]}}} \cdot \frac{\partial Z^{[1]}}{\partial b^{[1]}}$$

$$dZ_2 = \frac{\partial \mathcal{L}}{\partial A^{[2]}} \cdot \frac{\partial A^{[2]}}{\partial Z^{[2]}}$$

$$= \frac{1}{m}\sum\left(-\frac{y}{A^{[2]}} + \frac{(1-y)}{1-A^{[2]}}\right) \times A^{[2]}(1-A^{[2]})$$

$$= \frac{1}{m}\sum -y(1-A^{[2]}) + \frac{1}{m}\sum (1-y)A^{[2]}$$

$$= \frac{1}{m}\sum -y + yA^{[2]} + A^{[2]} - yA^{[2]}$$

$$\underbrace{dZ2}_{(n^{[2]}, m)} = \frac{1}{m}\sum \underbrace{A^{[2]}}_{(n^{[2]}, m)} - \underbrace{}_{(1, m)}$$

on fait un broadcasting pour y

$$A^{[2]} = n^{2}\overbrace{\begin{bmatrix} \quad \end{bmatrix}}^{m} - \begin{matrix} 1[\\ 1[\end{matrix} \overbrace{\quad}^{m} \begin{matrix}] \\ ] \end{matrix} m$$

⇒ on crée une matrice (n^2, m) en répétant le vecteur y

$$\text{alors } \underbrace{\frac{\partial \mathcal{L}}{\partial W^{[2]}}}_{(n^{[2]}, n^{[1]})} = \underbrace{dZ2}_{(n^{[2]}, m)} \times \underbrace{A^{[1]T}}_{(m, n^{[1]})}$$

$n^{[1]}$ m $n^{[2]}$ m

□

Gradient descent for NN

1- one hidden layer NN for binary classification

parameters : $w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}$

$(n^{[1]}, n^{[0]})$ $(n^{[1]}, 1)$ $(n^{[2]}, n^{[1]})$ $(n^{[2]}, 1)$

$n = n^{[0]}, n^{[1]}, n^{[2]} = 1$

Cost function : $J(w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]})$

$$= \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}, y)$$

Gradient descent

Repeat {

compute predictions $(\hat{y}^{(i)}, i \in \{1, ..., m\})$

$$dw^{[1]} = \frac{\partial J}{\partial w^{[1]}}$$

$$db^{[1]} = \frac{\partial J}{\partial b^{[1]}}$$

⋮

$$w^{[1]} = w^{[1]} - \alpha . dw^{[1]}$$

$$b^{[1]} = b^{[1]} - \alpha . db^{[1]}$$

Random initialization

if we initialize all weights to zero

for any example $a_1^{[1]} = a_1^{[2]}$

$\Rightarrow dz_1^{[1]} = dz_2^{[1]}$ (symmetry breaking problem

⇓

best initialization

$$w^{[1]} = np.random.randn(4,3) * 0.01$$

$$b^{[1]} = np.\text{zero}~~.random.randn~~(4,1)$$ (no pb for b)

to have small numbers to avoid flat parts of the activation functions ⇒ faster computation

formulas for computing derivatives

$$z^{[1]} = w^{[1]} x + b^{[1]}$$

$$A^{[1]} = g^{[1]}(z^{[1]})$$

$$z^{[2]} = w^{[2]} z^{[1]} + b^{[1]}$$

$$A^{[2]} = g^{[2]}(z^{[2]})$$

Backpropagation

$$dz^{[2]} = A^{[2]} - Y$$

$$dw^{[2]} = \frac{1}{m} dz^{[2]} A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} np.sum(dz^{[2]}, axis=1, keepdims=True)$$

keepdims = True → prevent numpy to return rank 1 arrays

$$dz^{[1]} = \underbrace{w^{[2]T} dz^{[2]}}_{(n^{[1]}, m)} * \underbrace{g^{[1]'}(z^{[1]})}_{(n^{[1]}, m)}$$

$*$: element wise product

$$dw^{[1]} = \frac{1}{m} dz^{[1]} X^T$$

$$db^{[1]} = \frac{1}{n} np.sum(dz^{[1]}, axis=1, keepdims=True)$$

$$x,\ w^{[1]},\ b^{[1]} \rightarrow \boxed{z^{[1]} = w^{[1]}x + b^{[1]}} \rightarrow \boxed{a^{[1]} = g(z^{[1]})} \xrightarrow{\ w^{[2]},\ b^{[2]}\ } \boxed{z^{[2]} = w^{[2]}a^{[1]} + b^{[2]}} \rightarrow \boxed{a^{[2]} = \sigma(z^{[2]})} \rightarrow \boxed{\mathcal{L}(a^{[2]}, y)}$$

$$\mathcal{L}(a^{[2]}, y) = -y \log(a^{[2]}) + (1-y)\log(1-a^{[2]})$$

$$da^{[2]} = \frac{d\mathcal{L}}{da^{[2]}} = \frac{-y}{a^{[2]}} + \frac{1-y}{1-a^{[2]}}$$

$$dz^{[2]} = da^{[2]} \cdot \frac{da}{dz^{[2]}}$$

$$= da^{[2]} \cdot g'(3) \quad \text{or } g(3) = \sigma(3)$$

$$\Rightarrow dz^{[2]} = a^{[2]} - y$$

$$dw^{[2]} = dz^{[2]} a^{[1]T}$$

$$db^{[2]} = dz^{[2]}$$

$$da^{[1]} = w^{[2]T} \cdot dz^{[2]}$$

$$dz^{[1]} = w^{[2]T} \cdot dz^{[2]} \underbrace{*}_{\text{element wise product}} g^{[1]\prime}(z^{[1]})$$

d

$$dw^{[1]} = dz^{[1]} a^{[0]T}$$

$$db^{[1]} = dz^{[1]}$$

$$Z^{[i]} = \begin{bmatrix} | & & | \\ z^{[i](1)} & \cdots & z^{[i](m)} \\ | & & | \end{bmatrix}$$

$$z^{[i]} = w^{[i]} X + b^{[i]}$$

$$A^{[i]} = g^{[i]}(z^{[i]})$$

$$\Downarrow$$

vectorized

$$dZ^{[2]} = A^{[2]} - Y$$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} \cdot A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis=1, keepdims=True)$$

$$dZ^{[1]} = W^{[2]T} \cdot dZ^{[2]} * g^{[1]\prime}(Z^{[1]})$$

$$dW^{[2]} = \frac{1}{m} dZ^{[1]} A^{[0]T}$$

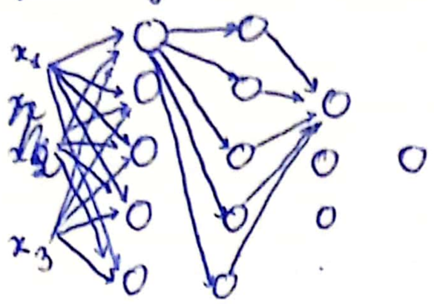
$$db^{[2]} = \frac{1}{m} \cdot np.sum(dZ^{[1]}, axis=1, keepdims=True)$$

Deep L-layer NN

The number of hidden layers is a hyperparameter

The number of neurons in hidden layers is a hyperparameter

example : 4 layer neural network



$L = \#\ \text{layers} = 4$

$n^{[l]} = \#$ units in layer $l \in \{1, \dots, L\}$

$n^{[1]} = 5 \quad n^{[2]} = 5 \quad n^{[3]} = 3 \quad n^{[4]} = 1$

$n^{[0]} = n = \#$ features

$a^{[l]}$ = activation in layer l

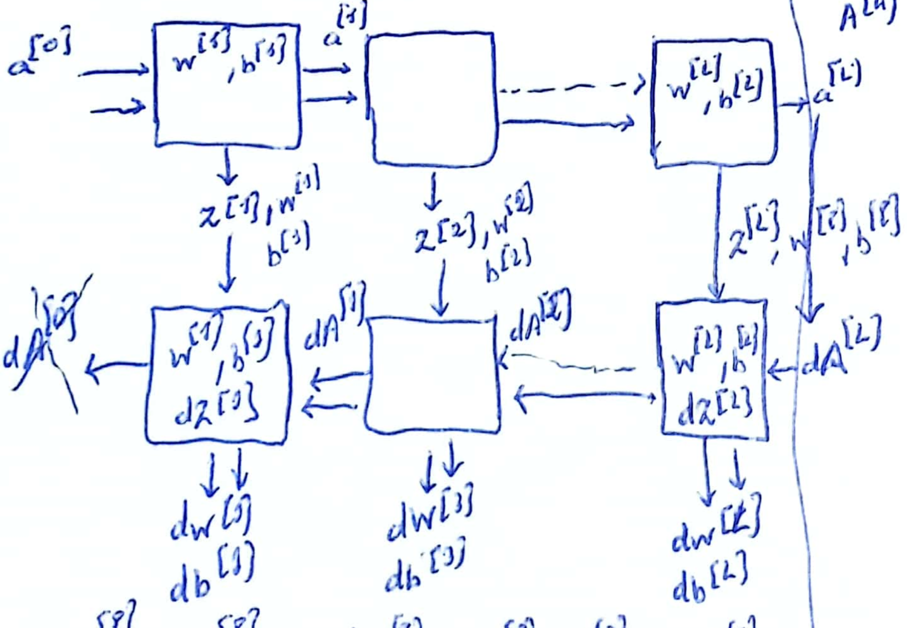
$a^{[l]} = g^{[l]}(z^{[l]})$

$w^{[l]}$ = weights for $z^{[l]}$

$b^{[l]}$

$X = a^{[0]} \qquad a^{[L]} = \hat{y}$

computation graph for forward and backward propagation (one iteration of gradient descent)



$W^{[l]} = W^{[l]} - \alpha\, dW^{[l]} \qquad b^{[l]} = b^{[l]} - \alpha\, db^{[l]}$

forward propagation for a single training example

$$z^{[1]} = w^{[1]^T} x + b^{[1]}$$
$$a^{[1]} = g^{[1]}(z^{[1]})$$
$$z^{[2]} = w^{[2]^T} a^{[1]} + b^{[2]}$$
$$a^{[2]} = g^{[2]}(z^{[2]})$$
$$\vdots$$
$$z^{[4]} = w^{[4]^T} a^{[3]} + b^{[4]}$$
$$a^{[4]} = g^{[4]}(z^{[4]})$$

$$\Rightarrow \quad z^{[l]} = w^{[l]^T} a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g^{[l]}(z^{[l]})$$

forward propagation for the whole training set

$$\left.\begin{aligned}
Z^{[1]} &= W^{[1]} X + b^{[1]} \\
A^{[1]} &= g^{[1]}(Z^{[1]}) \\
Z^{[2]} &= W^{[2]} A^{[1]} + b^{[2]} \\
A^{[2]} &= g^{[2]}(Z^{[2]}) \\
Z^{[3]} &= W^{[3]} A^{[2]} + b^{[3]} \\
A^{[3]} &= g^{[3]}(Z^{[3]}) \\
Z^{[4]} &= W^{[4]} A^{[3]} + b^{[4]} \\
A^{[4]} &= g^{[4]}(Z^{[4]})
\end{aligned}\right\}$$

On n'a pas le choix autre de faire un for loop over the L layers of the NN

Forward propagation

Input : $a^{[l-1]}$

output $a^{[l]}$, cache $(z^{[l]}, w^{[l]}, b^{[l]})$

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$$

$$A^{[l]} = g^{[l]}(z^{[l]})$$

backward

$$dZ^{[l]} = dA^{[l]} * g^{[l]'}(z^{[l]})$$

element wise product

$$dW^{[l]} = \frac{1}{m} dz^{[l]} A^{[l-1]T}$$

$$db^{[l]} = \frac{1}{m} \cdot np.sum(dz^{[l]}, axis=1, Keepdims=true)$$

$$dA^{[l-1]} = w^{[l]T} \cdot dz^{[l]}$$

TD 6 : Reseaux de neuronnes

Def 1

$g : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$

$(x, w, b) \mapsto g(x, w, b) = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right)$

σ : fonction d'activation

$\sigma(x) = \dfrac{1}{1+e^{-x}}$

Def 2 : apprentissage du neuronne : estimation par les moindres carrés des paramètres du neuronne

1) $\begin{cases} r : \mathbb{R}^{n} \times \mathbb{R} \longrightarrow \mathbb{R}^{K} \\ (w, b) \longrightarrow \begin{pmatrix} y^{1} - \sigma\left(\sum_{i=1}^{n} (w_i x_i + b)\right) \\ \vdots \\ y^{K} - \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right) \end{pmatrix} \end{cases}$

n (w_i) ; b ; on a K points ; indice pas puissance ; indice pas puissance

$\begin{cases} \min f(w,b) = \frac{1}{2} \| r(\beta) \|^2 \\ (w,b) \in \mathbb{R}^n \times \mathbb{R} \end{cases}$

2) ce n'est pas linéaire car σ est non linéaire

3) $\sigma = Id$

$r(w,b) = \begin{pmatrix} y^1 - \sum_{i=1}^{n} (w_i x_i + b) \\ \\ y^K - \sum_{i=1}^{n} (w_i x_i + b) \end{pmatrix}$

$= \begin{pmatrix} y_1 \\ \vdots \\ y^K \end{pmatrix} - \begin{pmatrix} (x^1)^T & 1 \\ \vdots & \vdots \\ (x^K)^T & 1 \end{pmatrix} \begin{pmatrix} w \\ b \end{pmatrix}$

$x^1 = \begin{pmatrix} x_1^1 \\ \vdots \\ x_n^1 \end{pmatrix}$

$w = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}$

$X = \begin{pmatrix} (x^1)^T & 1 \\ \vdots & \vdots \\ (x^K)^T & 1 \end{pmatrix}$

1.4/ $r(\beta) = \begin{pmatrix} y^1 - \sigma(\langle w, x^1\rangle + b) \\ \vdots \\ y^K - \sigma(\langle w, x^K\rangle + b) \end{pmatrix}$

$$\nabla_f(\beta) = J_f(\beta)^T \cdot r(\beta)$$

$$J_f(\beta) = \left(\begin{array}{c|c} -\sigma'(\langle w, x^1\rangle + b)\,(x^1)^T & -\sigma'(\langle w, x^1\rangle + b) \\ \vdots & \\ -\sigma'(\langle w, x^K\rangle + b)\,(x^K)^T & -\sigma'(\langle w, x^K\rangle + b) \end{array}\right)$$

Exercice 2 ; reseau à plusieurs couches

2.1/ L : nombre de couches $= 4$

$l : 1, \dots, 4$ couches

$n_1 = 2$

$n_2 = 5$ $\qquad n_0 = 4$

$n_3 = 3$

$n_4 = 1$

$W^1 \in M_{(n_1, n_0)}(\mathbb{R})$ $\qquad b^1 \in \mathbb{R}^{2 = n_2} = \begin{bmatrix} b_1^1 \\ b_2^1 \end{bmatrix}$

$W^2 \in M_{(n_2, n_1)}(\mathbb{R})$ $\qquad b^2 \in \mathbb{R}^{5 = n_2}$

$W^3 \in M_{(n_3, n_2)}(\mathbb{R})$ $\qquad b^3 \in \mathbb{R}^3$

$W^4 \in M_{(n_4, n_3)}(\mathbb{R})$ $\qquad b^4 \in \mathbb{R}^1$

2.2/ a^l : sortie de la couche l : activation de la couche l

→ on applique σ à chaque élément du vecteur

$$a^{l+1} = \vec{\sigma}_{n_{l+1}}\left(W^{l+1} \cdot a^l + b^{l+1}\right)$$

$$= \Big(\sigma$$

$$\beta = \{ W^1, b^1 ; W^2, b^2 ; W^3, b^3 ; W^4, b^4 \}$$

$$\in \mathcal{M}_{n_1, n_0}(\mathbb{R}) \times \mathbb{R}^{n_1} \times \mathcal{M}_{n_2, n_1}(\mathbb{R}) \times \mathbb{R}^{n_2} \times \mathcal{M}_{n_3, n_2}(\mathbb{R}) \times \mathbb{R}^{n_3} \times \mathcal{M}_{n_4, n_3}(\mathbb{R}) \times \mathbb{R}^{n_4} = \mathcal{E}$$

~~en route~~

terminaison de tous les neurones

alors $\underset{\in \mathbb{R}^4}{x}$, $\underset{\in \mathcal{E}}{\beta}$

$$y(x, \beta) = a^4 = \vec{\sigma}_{n_4}\left(W^4 \vec{\sigma}_{n_3}\left(W^3 \vec{\sigma}_{n_2}\left(W^2 \vec{\sigma}_{n_1}\left(W^1 x + b^1\right) + b^2\right) + b^3\right) + b^4\right)$$

$r : \mathcal{E} \to \mathbb{R}^K$

2.3/
$$r(\beta) = \begin{pmatrix} y^1 - y(x^1, \beta) \\ \vdots \\ y^K - y(x^K, \beta) \end{pmatrix}$$

$$\begin{cases} \min f(\beta) = \frac{1}{2} \|r(\beta)\|^2 \\ \beta \in \mathcal{E} \end{cases}$$

2.4/ ~~les~~ dérivés partielles par rapport à $\beta^4 = (W^4, b^4)$

$$\frac{\partial y}{\partial \beta^4} = \vec{\sigma}'_{n_4}\left(W^4 a^3(x, \beta_1, \beta_2, \beta_3) + b^4\right) \odot \left[a^3(x, \beta_1, \beta_2, \beta_3)^T, 1\right]$$

produit terme à terme

$$\frac{\partial y}{\partial \beta^3} = \vec{\sigma}'_{n_4}\left(W^4 a^3(x, \beta_1, \beta_2, \beta_4) + b^4\right) \odot \left[\frac{\partial}{\partial \beta^3}\left(W^4 a^3(x, \beta_1, \beta_2, \beta_3) + b^4\right)\right]$$

$$\parallel$$

$$\left[W^4\left(\frac{\partial}{\partial \beta^3}\left(a^3(x, \beta_1, \beta_2, \beta_3)\right)\right)\right]$$