Deep Learning

Implementation

Vectorisation

En programmation, cela consiste à mettre nos données dans des vecteurs, des matrices ou des tableaux à N-dimension afin d'effectuer des opérations mathématiques sur l'ensemble de ces données.

Exemple :
multiplier les éléments d'une liste

$$\begin{array}{l} [4, \quad 6, \quad 2, \quad \cdots, \quad 7] \times 2 \\ [8, \quad 12, \quad 4, \quad \cdots, \quad 14] \end{array}$$

Vectorisation

En programmation, cela consiste à mettre nos données dans des vecteurs, des matrices ou des tableaux à N-dimension afin d'effectuer des opérations mathématiques sur l'ensemble de ces données.

Exemple :
multiplier les éléments d'une liste

$$[4, \quad 6, \quad 2, \quad \cdots, \quad 7] \times 2$$
$$[8, \quad 12, \quad 4, \quad \cdots, \quad 14]$$

A la place :
Multiplier un vecteur tout entier

$$\begin{bmatrix} 4 \\ 6 \\ 2 \\ \vdots \\ 7 \end{bmatrix} \times 2 = \begin{bmatrix} 8 \\ 12 \\ 4 \\ \vdots \\ 14 \end{bmatrix}$$

Vectorisation

En programmation, cela consiste à mettre nos données dans des vecteurs, des matrices ou des tableaux à N-dimension afin d'effectuer des opérations mathématiques sur l'ensemble de ces données.

Exemple :
multiplier les éléments d'une liste

```
liste_A = [4, 6, 2, 7]

liste_B = [i * 2 for i in liste_A]
```

A la place :
Multiplier un vecteur tout entier

```
A = np.array([4, 6, 2, 7])

B = A * 2
```

Code plus simple
Execution plus rapide

Ré-écrire sous forme matricielle toutes les équations que l'on a vues dans les dernieres vidéos.

$$\boxed{\begin{aligned} z &= w_1x_1 + w_2x_2 + b \\ a &= \frac{1}{1+e^{-z}} \end{aligned}}$$

Modèle

$$\boxed{\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \times log(a^{(i)}) + (1-y^{(i)}) \times log(1-a^{(i)})}$$

Fonction Coût

$$w_1 = w_1 - \alpha\frac{\partial \mathcal{L}}{\partial w_1} \qquad \frac{\partial \mathcal{L}}{\partial w_1} = \frac{1}{m}\sum_{i=1}^{m}(a^{(i)} - y^{(i)})x_1$$

$$w_2 = w_2 - \alpha\frac{\partial \mathcal{L}}{\partial w_2} \qquad \frac{\partial \mathcal{L}}{\partial w_2} = \frac{1}{m}\sum_{i=1}^{m}(a^{(i)} - y^{(i)})x_2$$

Descente de Gradient

Matrices

Les matrices sont des tableaux à 2 dimensions dont on se sert pour résoudre facilement et rapidement une grande quantité de problèmes mathématiques.

$$\begin{array}{c c} & \begin{array}{cccc} 1 & 2 & \dots & \mathrm{n} \end{array} \\ \begin{array}{c} 1 \\ 2 \\ \vdots \\ \mathrm{m} \end{array} & \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \end{array} \in \mathbb{R}^{m \times n}$$

exemple de matrice de dimension (m, n)

(m lignes, n colonnes)

Dans le cadre du Deep Learning

Il y a 3 opérations élémentaires à connaître sur le calcul matriciel.

1. Les additions et les soustractions
2. Les transposées
3. Les multiplications

1. Additions et Soustractions

Pour additionner ou soustraire 2 matrices, il faut que leurs dimensions soient égales.

$$\overbrace{\begin{bmatrix}2 & 3\\1 & 4\\3 & 1\end{bmatrix}+\begin{bmatrix}3 & 1\\2 & 3\\1 & 0\end{bmatrix}}^{\text{mêmes dimensions}}=\begin{bmatrix}5 & 4\\3 & 7\\4 & 1\end{bmatrix}$$

$$\underbrace{\begin{bmatrix}2 & 3\\1 & 4\\3 & 1\end{bmatrix}+\begin{bmatrix}3 & 1\\1 & 2\end{bmatrix}}_{\text{dimensions différentes}}=\text{impossible}$$

2. Transposition

Consiste à faire pivoter la matrice sur sa diagonale, ce qui a pour effet d'interchanger ses dimensions (le nombre de lignes devient le nombre de colonnes et vice versa).

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} \in \mathbb{R}^{4\times 3}$$

$$\begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix} \in \mathbb{R}^{3\times 4}$$

3. Multiplication Matricielle

Pour multiplier 2 matrices, il faut que le nombre de colonnes de la première soit égal au nombre de lignes de la deuxième

$$\overset{A}{\underbrace{\begin{bmatrix}2 & 1 & 3\\3 & 4 & 1\end{bmatrix}}} \cdot \overset{B}{\underbrace{\begin{bmatrix}3 & 1\\2 & 3\\1 & 0\end{bmatrix}}}$$

$$2 \times \boxed{3 \qquad 3} \times 2$$

3. Multiplication Matricielle

Le résultat du produit est une combinaison linéaire entre les lignes de la matrice de gauche et les colonnes de la matrice de droite.

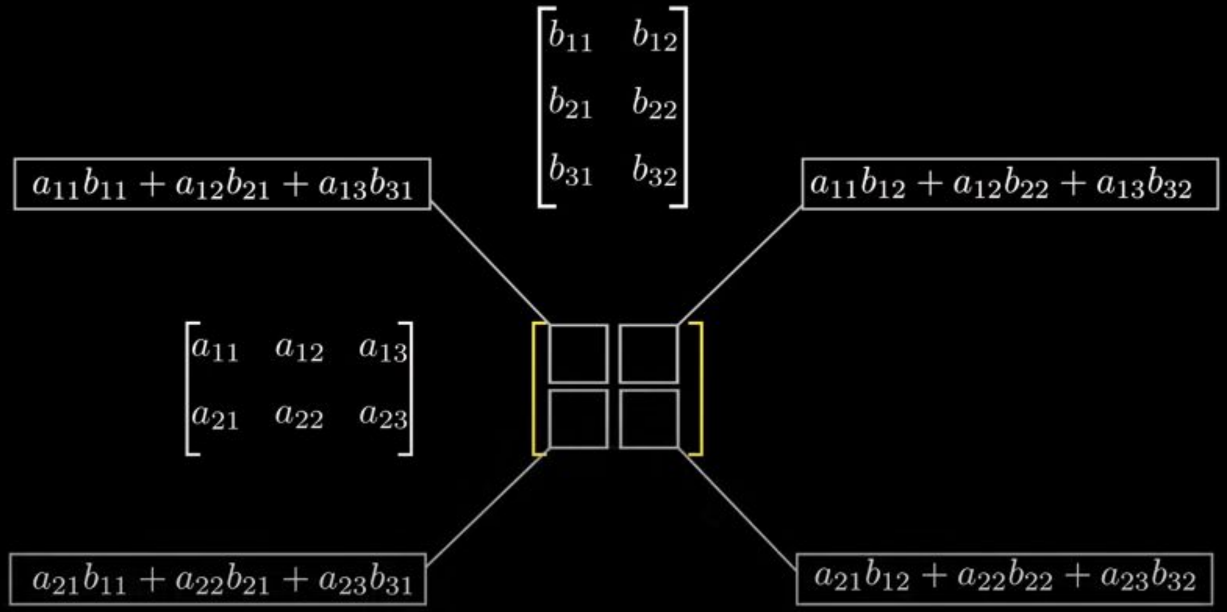
$$\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

$$\boxed{a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \quad \begin{bmatrix} \boxed{} & \\ & \end{bmatrix}$$

3. Multiplication Matricielle

Le résultat du produit est une combinaison linéaire entre les lignes de la matrice de gauche et les colonnes de la matrice de droite.



Vectorisation de nos équations

$$z = w_1 x_1 + w_2 x_2 + b$$

$$a = \frac{1}{1 + e^{-z}}$$

Modèle

$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \times log(a^{(i)}) + (1 - y^{(i)}) \times log(1 - a^{(i)})$$

Fonction Coût

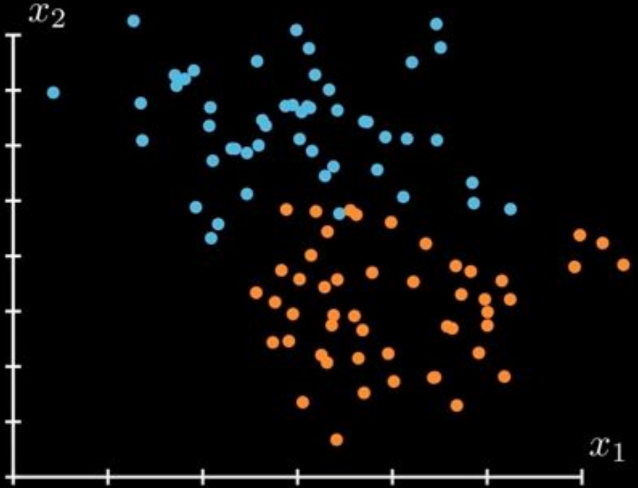
$$w_1 = w_1 - \alpha \frac{\partial \mathcal{L}}{\partial w_1} \qquad \left(\frac{\partial \mathcal{L}}{\partial w_1} = \frac{1}{m} \sum_{i=1}^{m} (a^{(i)} - y^{(i)}) x_1 \right)$$

$$w_2 = w_2 - \alpha \frac{\partial \mathcal{L}}{\partial w_2} \qquad \left(\frac{\partial \mathcal{L}}{\partial w_2} = \frac{1}{m} \sum_{i=1}^{m} (a^{(i)} - y^{(i)}) x_2 \right)$$

Descente de Gradient

```
X[:,0]
```

1. Vectorisation du dataset



$y = 0$ (plante non-toxique)
$y = 1$ (plante toxique)

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \\ \vdots & \vdots \\ x_1^{(m)} & x_2^{(m)} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^{m \times 1}$$

Par convention

m : nombre de données

n : nombre de variables

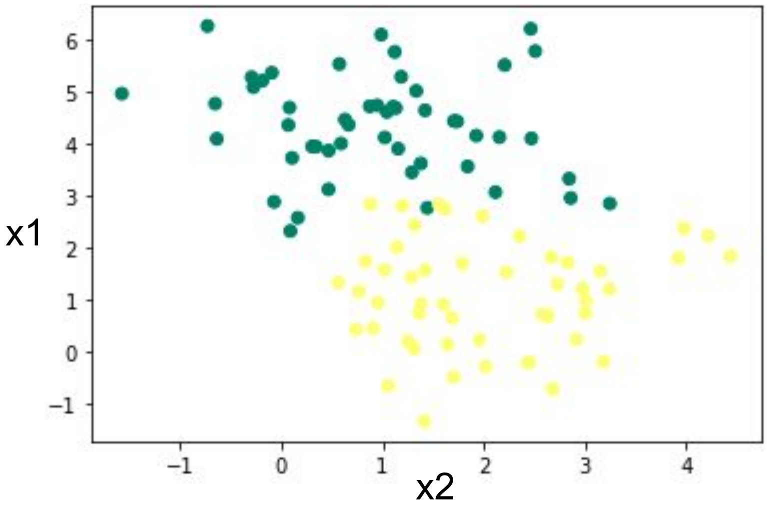
$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \\ \vdots & \vdots \\ x_1^{(m)} & x_2^{(m)} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^{m \times 1}$$

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs


X, y = make_blobs(n_samples=100, n_features=2,
centers=2, random_state=0)

plt.scatter(X[:,0], X[:,1], c=y, cmap="summer")
```



Vectorisation de nos équations

$$\boxed{Z = X \cdot W + b}$$

$$X = \begin{bmatrix} x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \in \mathbb{R}^{m \times n} \qquad W = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} \in \mathbb{R}^{n \times 1} \qquad b = \begin{bmatrix} b \\ b \\ \vdots \\ b \end{bmatrix} \in \mathbb{R}^{m \times 1}$$

$$a = \frac{1}{1 + e^{-z}}$$

```
Z = X.dot(W) + b
```

Vectorisation de nos équations

$$Z = X \cdot W + b$$

$$\boxed{a = \frac{1}{1 + e^{-z}}}$$

```
a = 1 / (1 + np.exp(-Z))
```

Vectorisation de nos équations

$$Z = X \cdot W + b$$

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

$$\boxed{\mathcal{L} = -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \times log(a^{(i)}) + (1 - y^{(i)}) \times log(1 - a^{(i)})}$$

```
1 / len(y) * np.sum(-y * np.log(a) - (1 - y) * np.log(1 - a))
```

5. Vectorisation de la Descente de Gradient

$$w_1 = w_1 - \alpha \frac{\partial \mathcal{L}}{\partial w_1}$$

$$w_2 = w_2 - \alpha \frac{\partial \mathcal{L}}{\partial w_2}$$

$$b = b - \alpha \frac{\partial \mathcal{L}}{\partial b}$$

```
dW = 1 / len(y) * np.dot(X.T, A - y)
```

$$\frac{\partial \mathcal{L}}{\partial W} = \underbrace{\begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_1} \\ \\ \frac{\partial \mathcal{L}}{\partial w_2} \end{bmatrix}}_{(2,\ 1)} = \begin{bmatrix} \frac{1}{m} \sum_{i=1}^{m} (a^{(i)} - y^{(i)}) x_1^{(i)} \\ \\ \frac{1}{m} \sum_{i=1}^{m} (a^{(i)} - y^{(i)}) x_2^{(i)} \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial W} = \underbrace{\begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_1} \\ \\ \frac{\partial \mathcal{L}}{\partial w_2} \end{bmatrix}}_{(2,\ 1)} = \frac{1}{m} \begin{bmatrix} (a^{(1)} - y^{(1)}) x_1^{(1)} + (a^{(2)} - y^{(2)}) x_1^{(2)} + \cdots + (a^{(m)} - y^{(m)}) x_1^{(m)} \\ (a^{(1)} - y^{(1)}) x_2^{(1)} + (a^{(2)} - y^{(2)}) x_2^{(2)} + \cdots + (a^{(m)} - y^{(m)}) x_2^{(m)} \end{bmatrix}$$

$$= \frac{1}{m} \underbrace{\begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(m)} \\ \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(m)} \end{bmatrix}}_{X^T} \cdot \left(\underbrace{\begin{bmatrix} a^{(1)} \\ a^{(2)} \\ \vdots \\ a^{(m)} \end{bmatrix}}_{A} - \underbrace{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}}_{y} \right)$$

```
dW = 1 / len(y) * np.dot(X.T, A - y)
```

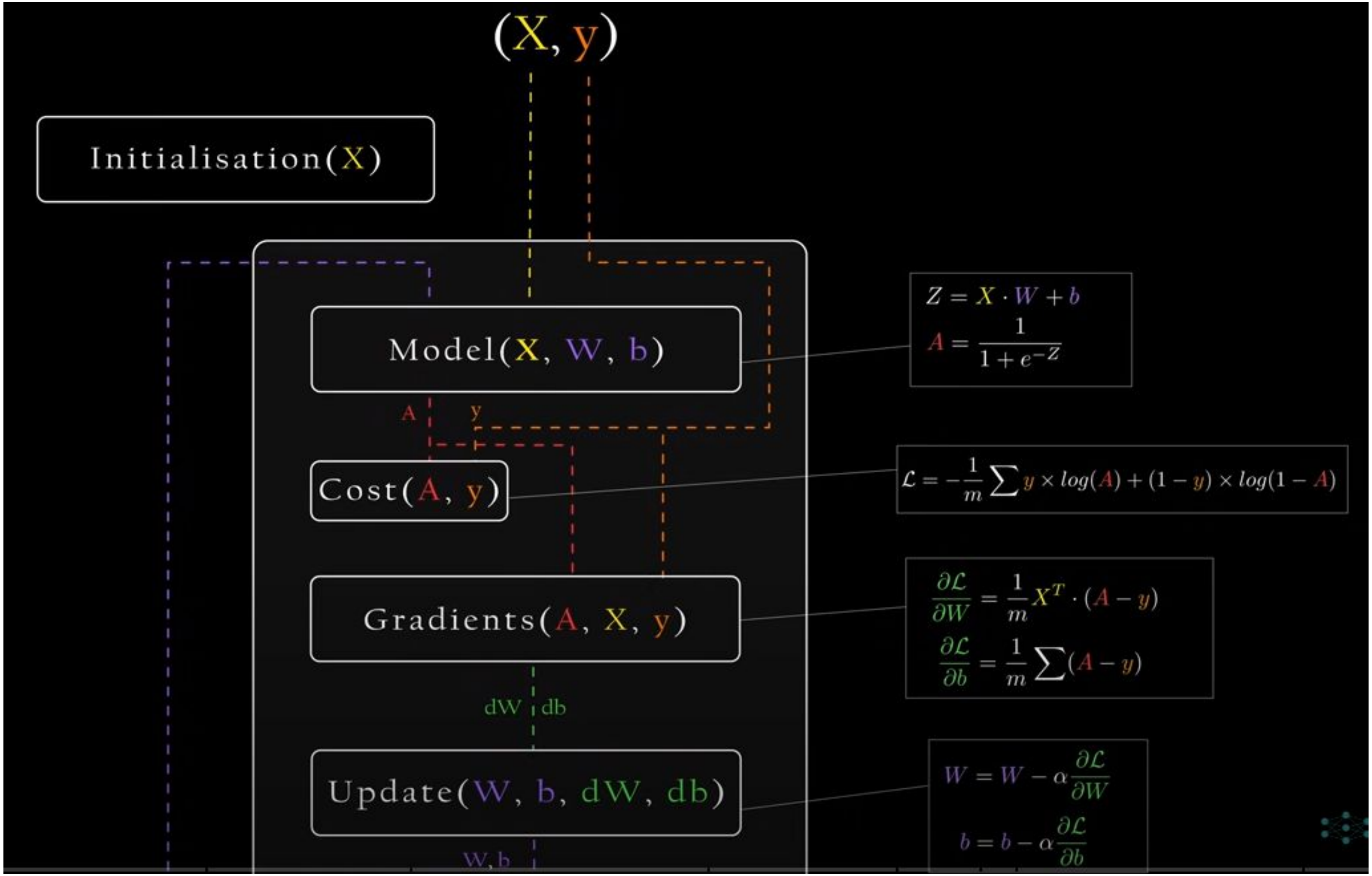
6. Vectorisation des Gradients

Le paramètre b étant un nombre réel (et non un vecteur) sa dérivée est elle aussi un nombre réel.

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} \boxed{a^{(1)} - y^{(1)} + a^{(2)} - y^{(2)} + \ldots + a^{(m)} - y^{(m)}}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{m} \sum (A - y)}$$

```
db = 1 / len(y) * np.sum(A - y)
```


(X, y)
Initialisation(X)
Model(X, W, b)
$Z = X \cdot W + b$
$A = \frac{1}{1 + e^{-Z}}$
A
y
Cost(A, y)
$\mathcal{L} = -\frac{1}{m} \sum y \times log(A) + (1 - y) \times log(1 - A)$
Gradients(A, X, y)
$\frac{\partial \mathcal{L}}{\partial W} = \frac{1}{m} X^T \cdot (A - y)$
$\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{m} \sum (A - y)$
dW
db
Update(W, b, dW, db)
$W = W - \alpha \frac{\partial \mathcal{L}}{\partial W}$
$b = b - \alpha \frac{\partial \mathcal{L}}{\partial b}$
W, b