Faculté des Sciences Université Mohammed V de Rabat	Deep Learning MSID 2024 / 2025	جامعة محمد الخامس كلية العلوم الرباط

TP 3

Rétropropagation (Back-Propagation)

Objectifs pédagogiques

- Implémenter étape par étape les principes fondamentaux des réseaux de neurones, notamment :
  - L'initialisation des paramètres.
  - La propagation avant (forward-propagation).
  - La rétropropagation (back-propagation) avec calcul des gradients.
  - Les défis des gradients (évanescents/explosifs) et leurs solutions.

Étape 1 : Initialisation des Poids et des Biais

Théorie : Pourquoi est-ce important ?

- *Problème d'initialisation aléatoire :* Les mauvaises initialisations peuvent ralentir l'apprentissage ou empêcher la convergence.
- *Méthode recommandée :* Xavier

Pour chaque couche **c** , les poids $W^{[c]}$ sont initialisés selon :

$$\boldsymbol{W^{[C]} \sim u(-\sqrt{\frac{1}{n_{c-1}}}, \sqrt{\frac{1}{n_{c-1}}})}$$

où n_{c-1} est le nombre de neurones de la couche précédente.

Implémenter une fonction Python pour initialiser les paramètres pour un réseau de deux couches.

Étape 2 : Propagation Avant (Forward-Propagation)

Théorie : Calculs dans un réseau de deux couches

1. Première couche :
   - Calcul de la somme pondérée :

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

   - Fonction d'activation sigmoïde :

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

2. Deuxième couche :
   - Calcul de la somme pondérée :

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

   - Fonction d'activation sigmoïde :

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

Implémenter la propagation avant.

```
def sigmoid (Z):
    return


def forward_propagation (X, parameters):
    W1, b1, W2, b2 = parameters["W1"], parameters["b1"], parameters["W2"], parameters["b2"]
    Z1   =
    A1   =
    Z2   =
    A1   =



    return
```

Étape 3 : Rétropropagation (Back-Propagation)

Théorie : Calcul des gradients

- *Objectif* : Mettre à jour les poids en minimisant la fonction de perte :

$$J = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log\left(A^{[2](i)}\right) + (1 - y^{(i)})\log(1 - A^{[2](i)})]$$

- *Étapes de la rétropropagation :*

Pour un réseau de deux couches :

1. Gradient de la couche de sortie :

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

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

2. Propagation vers la première couche :

$$dA^{[1]} = W^{[2]T}\, dZ^{[2]}$$

$$dZ^{[1]} = dA^{[1]}.(A^{[1]})'\,,\ (A^{[1]})' = A^{[1]}.(1 - A^{[1]})$$

$$dW^{[1]} = \frac{1}{m}\, dZ^{[1]}X^{T}\,,\ db^{[1]} = \frac{1}{m}\sum dZ^{[1]}$$

Implémenter la rétropropagation.

```
def back-propagation ():

# Gradients pour la couche de sortie

# Gradients pour la couche cachée

    return
```

Étape 4 : Problèmes des Gradients

Théorie :

- *Gradients évanescents :* Lorsqu'on utilise la sigmoïde, les gradients se réduisent à des valeurs proches de zéro.
- *Gradients explosifs :* Amplification des gradients, entraînement instable.

Solutions :

1. Utiliser **ReLU** pour éviter l'atténuation des gradients.

$$\mathbf{ReLU(Z) = max(0, Z)}$$

2. Initialisation adaptée des poids (Xavier).

```
def relu (Z):
  return


def relu_derivative (Z):
   return
```