Projet_4_Aminata_Ndiaye

January 17, 2023

1 Projet 4

[2]:
```
import numpy as np
```

[31]:
```
def dot(W, x):
    value = np.dot(W, x)

    def vjp(u):
        return np.outer(u, x), W.T.dot(u)

    return value, vjp
```

#1) Implémenter la fonction relu et son VJP

[29]:
```
#Compute the VJP of the relu function


def relu(x):
  value=np.zeros(len(x))
  value[x>=0]=1
  def vjp(u):
    vjp_wrt_x = u * value
    return vjp_wrt_x,  # The comma is important!

  return value, vjp

x=np.array([1, -9])
u=np.array([-2, 1])
value, vjp = relu(x)
print(value)
print(vjp(u))
```

```
[1. 0.]
(array([-2.,  0.]),)
```

[30]:
```
def df_relu(x):
    n = len(x)
    value = np.maximum(x, 0)
```

```
    def vjp(u):
        epsilon = 1e-6
        e= np.ones(n)
        vjp_wrt_x = 1/ epsilon* (u * (np.maximum(x + epsilon*e, 0) - np.
 ↪maximum(x, 0)))
        return vjp_wrt_x

    return value, vjp

x=np.array([1, -9])
u=np.array([-2, 1])

value, df_vjp = df_relu(x)
print(value)
print(df_vjp(u))
```

```
[1 0]
[-2.  0.]
```

On obtient le même résultat

##2) Question 2

[35]:
```
def mlp2(x, W1, W2):

  dot_W1_x, vjp_dot = dot(W1, x)
  relu_W1_x, vjp_relu = relu(dot_W1_x)
  value, vjp_dot_W2 = dot(W2, relu_W1_x)


  def vjp(u):
      vjpW2, vjp_x2 = vjp_dot_W2(u)
      vjp_x1 = vjp_relu(vjp_x2.T)
      vjpW1, vjpx = vjp_dot(np.array(vjp_x1).T)

      return vjpx, vjpW1, vjpW2


  return value, vjp
x=np.array([1,  1])
W1=np.array([[1,2],[1,-1]])
W2=np.ones((2, 2))
u=np.array([0, -4])
value, vjp = mlp2(x, W1, W2)
print(value)
print(vjp(u))
```

```
[2. 2.]
(array([[-8.],
```

```
       [-4.]]), array([[-4., -4.],
       [-4., -4.]]), array([[ 0.,  0.],
       [-4., -4.]]))
```

On vérifie notre implémentation en utilisant les différences finies

#3) Question 3

[32]:
```
def squared_loss(y_pred, y):
    residual = y_pred - y

    def vjp(u):
        vjp_y_pred, vjp_y = residual * u, -residual * u
        return vjp_y_pred, vjp_y

    value = 0.5 * np.sum(residual ** 2)
    # The code requires every output to be an array.
    return np.array([value]), vjp



value, vjp = squared_loss(np.array([6, 5]), np.array([0, 4]))
print(value)
print(vjp(4))
```

```
[18.5]
(array([24,  4]), array([-24,  -4]))
```

1.1 4) Question 4

[41]:
```
def loss(x, y, W1, W2):
  value_mlp2, vjp_mlp2 = mlp2(x,W1, W2)
  value_squaredloss, vjp_squaredloss = squared_loss(value_mlp2,y)
  def vjp(u):
    vjp_x_sl, vjp_wrt_y = vjp_squaredloss(u)
    vjp_wrt_x, vjp_wrt_W1, vjp_wrt_W2 = vjp_mlp2 (vjp_x_sl)
    return  vjp_wrt_x,vjp_wrt_y, vjp_wrt_W1, vjp_wrt_W2

  value=value_squaredloss
  return value, vjp

x=np.array([1, 0])
y=np.array([1, 0])
w1=np.array([[1, 2],[4, 1]])
W2=np.array([[1, 2],[0, -3]])

value, vjp = loss(x, y ,W1, W2)
print(value)
print(vjp(1))
```

```
[6.5]
(array([[15.],
       [-9.]]), array([-2.,  3.]), array([[ 2.,  0.],
       [13.,  0.]]), array([[ 2.,  2.],
       [-3., -3.]]))
```