M8-L1-P1

November 4, 2023

1 M8-L1 Problem 1

In this problem you will solve for $\frac{\partial L}{\partial W_2}$ and $\frac{\partial L}{\partial W_1}$ for a neural network with two input features, a hidden layer with 3 nodes, and a single output. You will use the sigmoid activation function on the hidden layer. You are provided an input sample x_0, the current weights W_1 and W_2, and the ground truth value for the sample, $t = -2$

$L = \frac{1}{2}e^T e$

[ ]:
```
import numpy as np

x0 = np.array([[-2], [-6]])

W1 = np.array([[-2, 1],[3, 8],[-12, 7]])
W2 = np.array([[-11, 2, 5]])

t = np.array([[-2]])
```

1.1 Define activation function and its derivative

First define functions for the sigmoid activation functions, as well as its derivative:

[ ]:
```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def del_sigmoid(x):
    s = sigmoid(x)
    return s*(1-s)
```

2 Forward propagation

Using your activation function, compute the output of the network y using the sample x_0 and the provided weights W_1 and W_2

[ ]:
```
a1 = W1 @ x0
x1 = sigmoid(a1)
a2 = W2 @ x1
y = a2
```

```
print(y)
```

```
[[-1.31123207]]
```

2.1 Backpropagation

Using your calculated value of y, the provided value of t, your σ and σ' function, and the provided weights W_1 and W_2, compute the gradients $\frac{\partial L}{\partial W_2}$ and $\frac{\partial L}{\partial W_1}$.

[ ]:
```
e = t - y
L = 0.5*(e.T @ e)

delta_2 = -e
dLdw2 = delta_2 * x1
delta_1 = delta_2 @ W2 @ del_sigmoid(a1)
dLdw1 = delta_1 * x0

print(dLdw2)
print(dLdw1)
```

```
[[8.21031503e-02]
 [2.43316128e-24]
 [1.04899215e-08]]
[[1.59095662]
 [4.77286987]]
```