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, $t = -2$

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

In [1]:

```
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]])
```

Define activation function and its derivative

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

In [8]:

```
# YOUR CODE GOES HERE
def sigmoid(x):
    return 1./(1.+np.exp(-x))

def dsigmoid(x):
    return np.exp(-x)/(1.+np.exp(-x))**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

In [9]:

```
# YOUR CODE GOES HERE
y = sigmoid(W2@sigmoid(W1@x0))
```

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 $\frac{\partial L}{\partial W_1}$.

In [11]:

```
# YOUR CODE GOES HERE
dLdW2 = dsigmoid(W1@x0)*sigmoid(W1@x0)
```