Cálculo do gradiente

In [1]:

```
import numpy as np
```

In [2]:

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

In [3]:

```
def sigmoid_prime(x):
        return sigmoid(x) * (1 - sigmoid(x))
```

In [4]:

```
learnrate = 0.5
x = np.array([1, 3, 4, 7])
y = np.array(0.5)
b = 0.5
```

In [5]:

```
# Pesos iniciais
w = np.array([0.5, -0.5, 0.3, 0.2])
```

In [6]:

```
h = np.dot(x, w)+b
```

In [7]:

```
nn_output = sigmoid(h)
print(nn_output)
```

```
0.8909031788043871
```

In [8]:

```
# TODO: erro Calcular de rede neural
error = y - nn_output
print(error)
```

```
-0.3909031788043871
```

In [9]:

```
# TODO: Calcule o termo de erro
# Lembre-se, isso requer o gradiente de saída, para o qual não  adicionamos
# especificamente uma variável.
error_term = error * sigmoid_prime(h)
```

In [10]:

```
# TODO: Calcule a mudança nos pesos
del_w = learnrate * error_term * x
```

In [11]:

```
print(del_w)
```

```
[-0.02256441 -0.04512882 -0.06769323 -0.09025764]
```

In [12]:

```
w = w + del_w
```

In [13]:

```
print(w)
```

```
[ 0.47743559 -0.54512882  0.23230677  0.10974236]
```

In [14]:

```
h = np.dot(x, w)+b
```

In [15]:

```
nn_output = sigmoid(h)
print(nn_output)
```

```
0.7355697130589234
```