Neuronales Netzwerk: Vorwärts- und Rückwärtsdurchlauf

In diesem Notebook betrachten wir ein einfaches neuronales Netzwerk mit einem Eingabeschicht, einer versteckten Schicht und einer Ausgabeschicht. Wir führen einen Vorwärtsdurchlauf durch, berechnen die Fehler und aktualisieren die Biases über zwei Iterationen mit der Backpropagation-Methode.

In [10]:
```python
# Importieren der benötigten Bibliotheken

#python
import numpy as np

# Import libary time to check execution date+time
import time
# print the date & time of the notebook
print('**********************************************************************
print("Actual date & time of the notebook:",time.strftime("%d.%m.%Y %H:%M
print('**********************************************************************

#check versions of libraries
#print('random version is: {}'.format(random.__version__))
print('numpy version is: {}'.format(np.__version__))

```

```
**********************************************************************
Actual date & time of the notebook: 13.09.2024 15:17:53
**********************************************************************
numpy version is: 1.18.1
```

Definieren der Aktivierungsfunktion und ihrer Ableitung

Wir Verwenden die Sigmoid-Aktivierungsfunktion und ihre Ableitung zur Berechnung der Neuronenaktivierungen und ihrer Gradienten.

In [11]:
```python
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

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

Initialisieren der Parameter

Definieren der Eingaben, Zielausgaben, Gewichte und Biases für das Netzwerk.

In [12]:
```
x = np.array([0.05, 0.10])  # Eingaben
y_target = np.array([0.01, 0.99])  # Zielausgaben

# Gewichte und Biases der ersten Schicht
W1 = np.array([[0.15, 0.25],
               [0.20, 0.30]])
b1 = np.array([0.35, 0.35])

# Gewichte und Biases der zweiten Schicht
W2 = np.array([[0.40, 0.50],
               [0.45, 0.55]])
b2 = np.array([0.60, 0.60])

# Lernrate
eta = 0.5
```

Vorwärtsdurchlauf

Berechnung der Aktivierungen für die versteckten und Ausgabeschichten.

In [13]:
```
def forward_pass(x, W1, b1, W2, b2):
    z1 = np.dot(x, W1) + b1
    h = sigmoid(z1)
    z2 = np.dot(h, W2) + b2
    y = sigmoid(z2)
    return h, y, z1, z2
```

Rückwärtsdurchlauf und Bias-Update

Berechnung der Gradienten für die Biases und deren Aktualisierung.

In [14]:
```
def backward_pass(x, y_target, h, y, z1, z2, W2):
    # Fehler für die Ausgabeschicht
    delta_y = y - y_target
    d_z2 = delta_y * sigmoid_derivative(z2)

    # Gradienten der Biases der Ausgabeschicht
    grad_b2 = d_z2

    # Fehler für die versteckte Schicht
    d_h = np.dot(d_z2, W2.T) * sigmoid_derivative(z1)

    # Gradienten der Biases der versteckten Schicht
    grad_b1 = d_h

    return grad_b1, grad_b2

def update_biases(b1, b2, grad_b1, grad_b2, eta):
    b1_new = b1 - eta * grad_b1
    b2_new = b2 - eta * grad_b2
    return b1_new, b2_new
```

Iteration 1

Durchführung des Vorwärts- und Rückwärtsdurchlaufs und Aktualisierung der Biases nach der ersten Iteration.

In [15]:
```
# Initiale Berechnungen
h, y, z1, z2 = forward_pass(x, W1, b1, W2, b2)
grad_b1, grad_b2 = backward_pass(x, y_target, h, y, z1, z2, W2)
b1_new, b2_new = update_biases(b1, b2, grad_b1, grad_b2, eta)

# Iteration 1
print(f"Initiale Biases: b1 = {b1}, b2 = {b2}")
print(f"Biases nach Iteration 1: b1 = {b1_new}, b2 = {b2_new}")
```

```
Initiale Biases: b1 = [0.35 0.35], b2 = [0.6 0.6]
Biases nach Iteration 1: b1 = [0.34561432 0.34502287], b2 = [0.53075072 0.61
904912]
```

Iteration 2

Durchführung des Vorwärts- und Rückwärtsdurchlaufs und Aktualisierung der Biases nach der zweiten Iteration.

In [16]:
```
# Eine zweite Iteration
h, y, z1, z2 = forward_pass(x, W1, b1_new, W2, b2_new)
grad_b1, grad_b2 = backward_pass(x, y_target, h, y, z1, z2, W2)
b1_new, b2_new = update_biases(b1_new, b2_new, grad_b1, grad_b2, eta)

# Iteration 2
print(f"Biases nach Iteration 2: b1 = {b1_new}, b2 = {b2_new}")

```

```
Biases nach Iteration 2: b1 = [0.34106066 0.33985788], b2 = [0.46037054 0.63
764129]
```

In [17]:
```
#Abschluss
# print current date and time

print("Date & Time:",time.strftime("%d.%m.%Y  %H:%M:%S"))
# end of import test
print ("*** End of Program Iterationen von Biases ***")

```

```
Date & Time: 13.09.2024  15:21:58
*** End of Program Iterationen von Biases ***
```