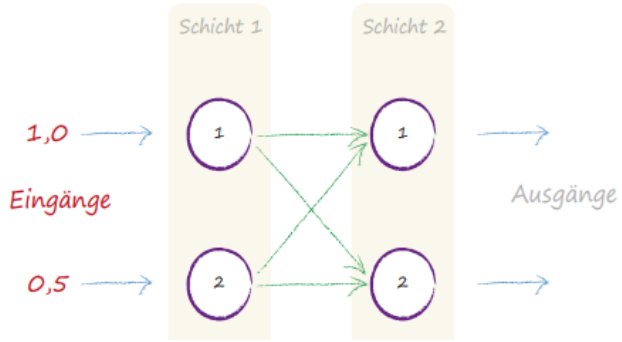
Vertiefung zu neuronalen Netzen - Kern des neuronalen Netzes

Wie lernt das neuronale Netz denn genau? Letztlich werden hierfür die Gewichte und die Anpassung der Gewichte im Rahmen der Vorwärtskalkulation und der Rückwärtskalkulation benutzt.

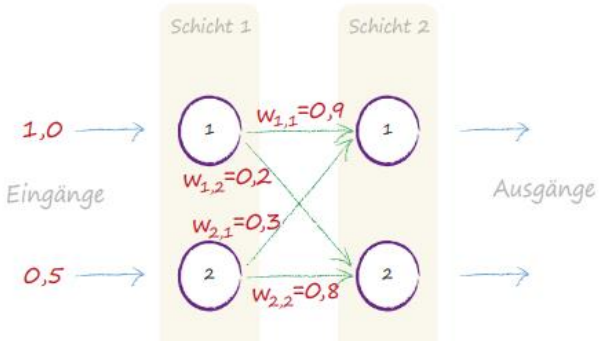
Vorwärtskalkulation

Arbeitsauftrag: Ergänzen Sie nachfolgende Übersicht mit den wesentlichen Informationen aus dem Erklärvideo.

Neuronales Netz mit Eingangswerten

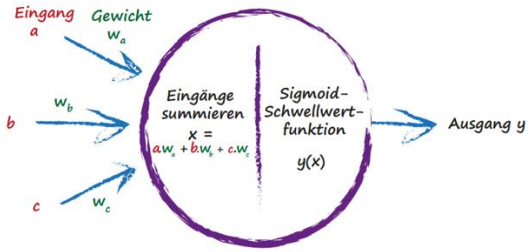


Neuronales Netz mit zufälligen Anfangsgewichten



Die furchteinflößende Vorwärtsberechnung

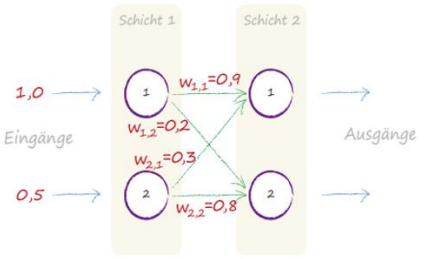
Knoten 1



Knoten 1

Aktivierungsfunktion

$$y = \frac{1}{(1+e^{-x})}$$

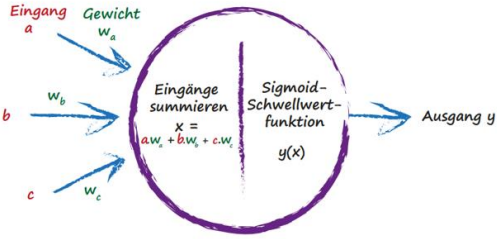


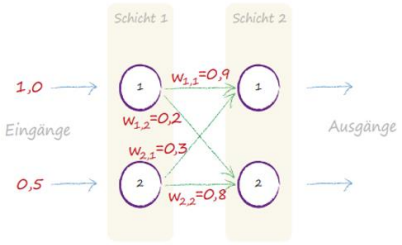
x = (Ausgabe vom ersten Knoten × Verknüpfungsgewicht) + (Ausgabe vom zweiten Knoten × Verknüpfungsgewicht)
x = (1,0 × 0,9) + (0,5 × 0,3)
x = 0,9 + 0,15
x = 1,05

y = 1/ (1+0,3499)
y = 1/1,3499
y = 0,7408

Die furchteinflößende Vorwärtsberechnung

Knoten 2





Knoten 2

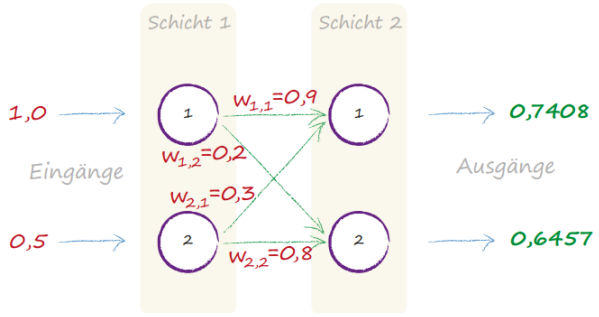
Aktivierungsfunktion

$$y = \frac{1}{(1+e^{-x})}$$

x = (Ausgabe vom ersten Knoten × Verknüpfungsgewicht) + (Ausgabe vom zweiten Knoten × Verknüpfungsgewicht)
x = (1,0 × 0,2) + (0,5 × 0,8)
x = 0,2 + 0,4
x = 0,6

y = 1/ (1+0,5488)
y = 1/1,5488
y = 0,6457

Das neuronale Netz mit den berechneten



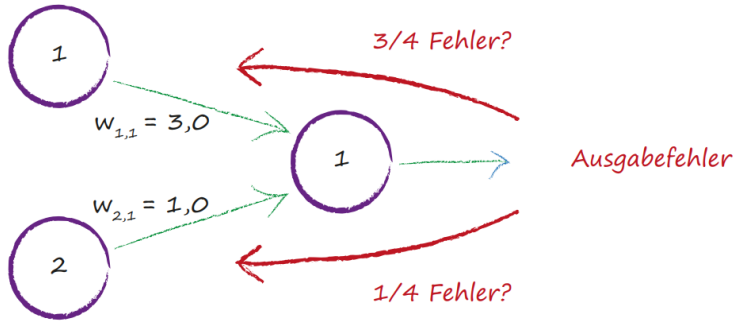
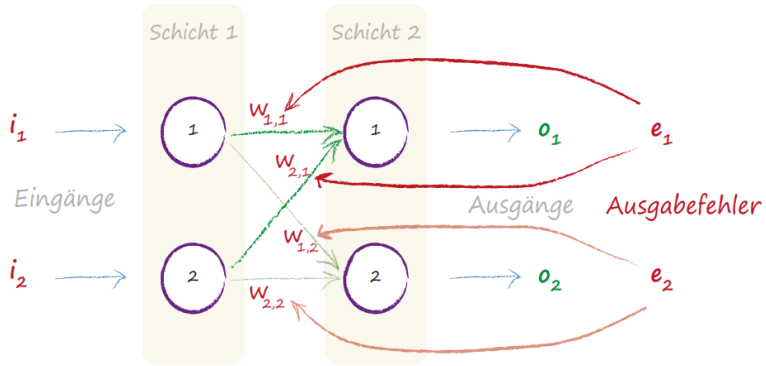
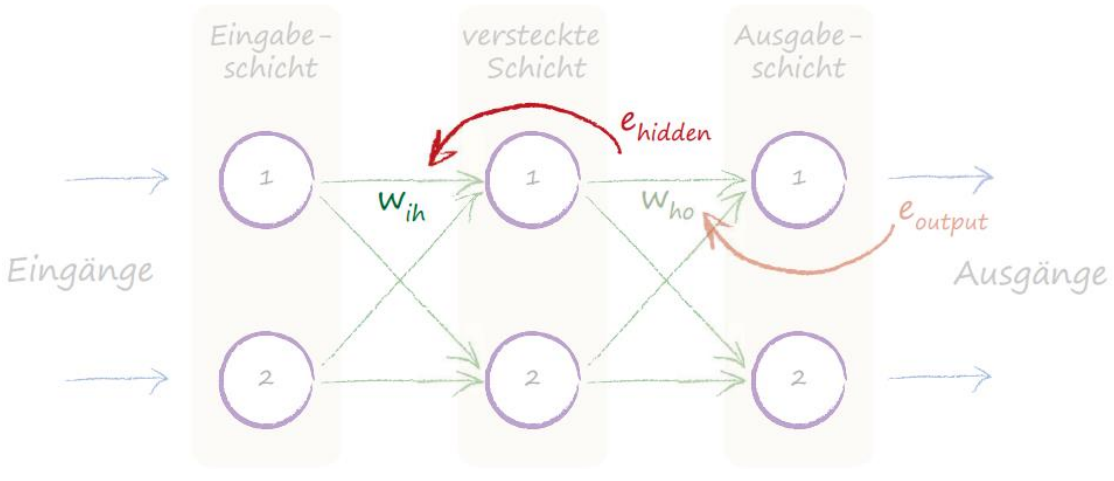
Arbeitsauftrag: Codenotizen Vorwärtskalkulation

Bitte machen Sie sich bei nachfolgenden Codeausschnitt zur Vorwärtskalkulation innerhalb eines neuronalen Netzes passende Notizen.

```
def query(self, inputs_list):
    # convert inputs list to 2d array
    inputs = numpy.array(inputs_list, ndmin=2).T

    # calculate signals into hidden layer
    hidden_inputs = numpy.dot(self.wih, inputs)
    # calculate the signals emerging from hidden layer
    hidden_outputs = self.activation_function(hidden_inputs)

    # calculate signals into final output layer
    final_inputs = numpy.dot(self.who, hidden_outputs)
    # calculate the signals emerging from final output layer
    final_outputs = self.activation_function(final_inputs)

    return final_outputs
```

Rückwärtskalkulation

Arbeitsauftrag: Ergänzen Sie nachfolgende Übersicht mit den wesentlichen Informationen aus dem Erklärvideo.

Aufteilung je Verknüpfungsgewicht		
Fehler von mehreren Ausgabenkoten zurückführen		
Aufteilung des Fehlers über mehrere Schichten		

Arbeitsauftrag: Codenotizen Rückwärtskalkulation

Bitte machen Sie sich bei nachfolgenden Codeausschnitt zur Rückwärtskalkulation innerhalb eines neuronalen Netzes passende Notizen.

```python
# train the neural network
def train(self, inputs_list, targets_list):
    # convert inputs list to 2d array
    inputs = numpy.array(inputs_list, ndmin=2).T
    targets = numpy.array(targets_list, ndmin=2).T

    # calculate signals into hidden layer
    hidden_inputs = numpy.dot(self.wih, inputs)
    # calculate the signals emerging from hidden layer
    hidden_outputs = self.activation_function(hidden_inputs)

    # calculate signals into final output layer
    final_inputs = numpy.dot(self.who, hidden_outputs)
    # calculate the signals emerging from final output layer
    final_outputs = self.activation_function(final_inputs)

    # output layer error is the (target - actual)
    output_errors = targets - final_outputs

    # hidden layer error is the output_errors, split by weights, recombined at hidden nodes
    hidden_errors = numpy.dot(self.who.T, output_errors)

    # update the weights for the links between the hidden and output layers
    self.who += self.lr * numpy.dot((output_errors * final_outputs * (1.0 - final_outputs)), numpy.transpose(hidden_outputs))

    # update the weights for the links between the input and hidden layers
    self.wih += self.lr * numpy.dot((hidden_errors * hidden_outputs * (1.0 - hidden_outputs)), numpy.transpose(inputs))

    pass
```

Arbeitsauftrag: Notieren Sie sich Ihre Antworten zu der Formsbefragung und korrigieren Sie diese, falls Ihre Antworten fehlerhaft waren, anschließend.

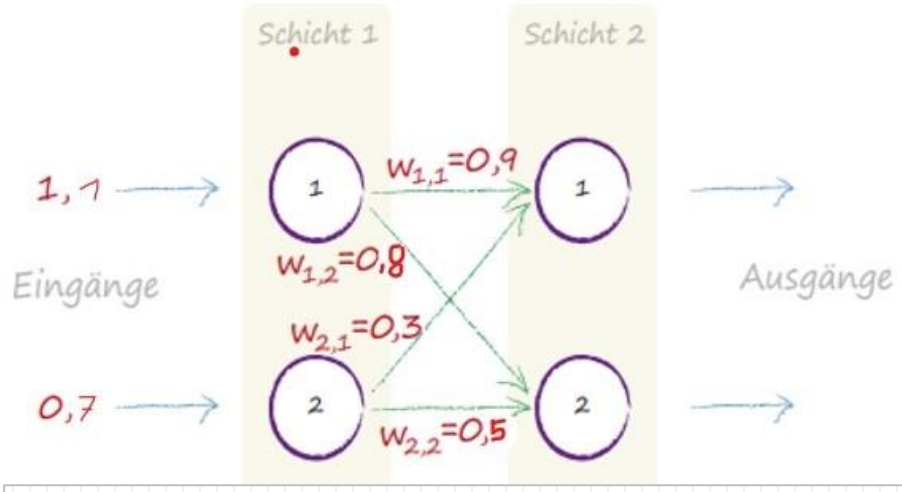
1

Markieren Sie alle korrekten Aussagen (1 Punkt)

- ◯ Ein neuronales Netz besteht immer aus mindestens drei Schichten.
- ◯ Das menschliche Gehirn ist den künstlichen neuronalen Netzen nachempfunden.
- ◯ Jeder Konten eins neuronalen Netzes ist mit jedem anderen Konten in jeder Schicht verbunden.
- ◯ Die stärke der Verbindung von Konten wird mit KwH ausgedrückt.
- ◯ Alles Quatsch!

2

Berechnen Sie die Summe der Signaleingänge bei dem Konten 2 der Schicht 2 (ohne Aktivierungsfunktion): (1 Punkt)



4

Ordnen Sie den Inhalt der query-Funktion in die richtige Reihenfolge. (1 Punkt)

```
final_inputs = numpy.dot(self.who, hidden_outputs)
final_outputs = self.activation_function(final_inputs)
hidden_inputs = numpy.dot(self.wih, inputs)
hidden_outputs = self.activation_function(hidden_inputs)
inputs = numpy.array(inputs_list, ndmin=2).T
return final_outputs
```

5

Welche Beschreibung trifft auf die Rückwärtskalkulation zu? (1 Punkt)

- [ ] Der Fehler wird anteilig am Gewicht der Verbindung zurückgeführt.
- [ ] Die Rückwärtskalkulation dient dazu, dass die Gewichte im Netz angepasst werden.
- [ ] Die Rückwärtskalkulation dient der Verschlüsselung der durch das Netz geschleusten Daten
- [ ] Der Fehler wir bei drei Verbindungen zu je einen Drittel zurückgeführt.
- [ ] Die Lernrate ("learningrate") gibt an, wie stark/in welchem Ausmaß Gewichte grundsätzlich ermittelte Fehlerkorrektur übernehmen.

6

Beschreiben Sie in Ihren Worten den Sinn, den folgende Codezeile in der Funktion train hat:
output_errors = targets - final_outputs