Künstliche neuronale Netze

How to build an AI

M. Heckel

12.12.2017

Denkende Maschinen

Neuronale Netze

Den Computer rechnen lassen

Quellen

Denkende Maschinen — Ein Traum der Menschheit

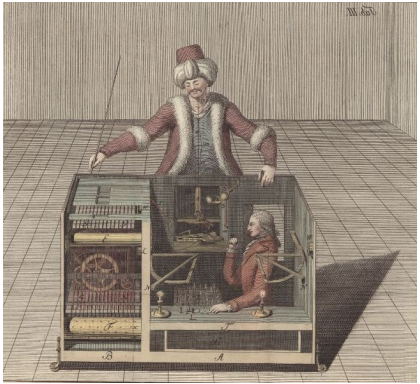


Abbildung: Schachtürke

- 1769: Schachtürke
- 1943: Theorie über neuronale Netze
- 1950: Turing-Test
- 1985: Backpropagation als Lernmethode
- 2017: KNN erstellt besseres KNN als Menschen

Was sind Neuronen?

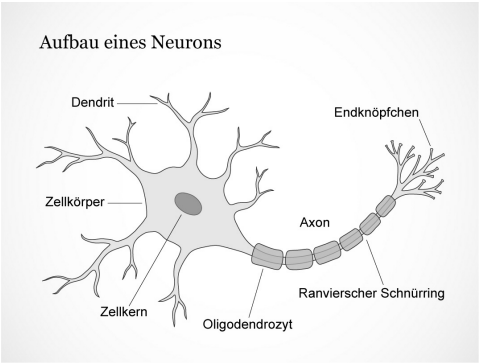


Abbildung: Biologisches Neuron

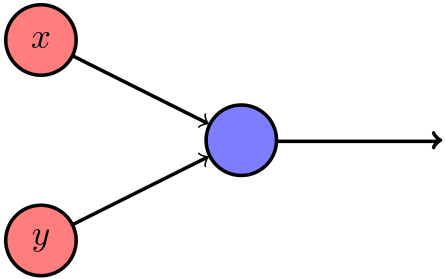


Abbildung: Formales Neuron

Künstliche Neuronen vernetzen

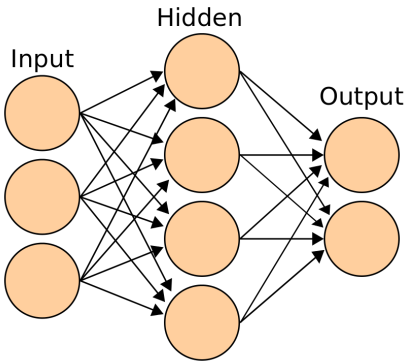


Abbildung: Vernetzte Neuronen

- 1 Input Layer
  Knotenanzahl = Anzahl der Eingangsparameter
- mindestens 1 Hidden Layer
  Knotenanzahl kann Leistung/Geschwindigkeit des Netzes optimieren
- 1 Output Layer
  Knotenanzahl = Anzahl der Ausgabewerte

Gewichte verteilen

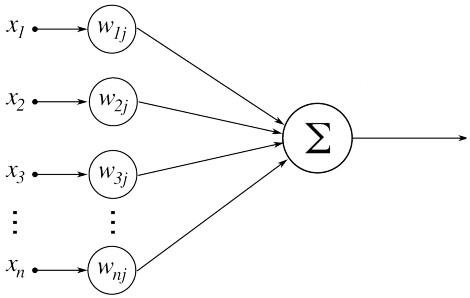


Abbildung: Gewichtete Eingänge

- Gewichte bestimmen Einfluss von Knoten auf andere Knoten
- Eigentliche ‘Intelligenz’ in Gewichten
- Lernen durch Anpassen der Gewichte

Bitte Aktivieren

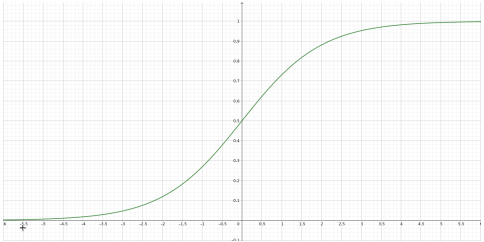


Abbildung: Logistische Funktion

- Künstliche Neuronen addieren alle Eingangswerte
- Nachteil: Unbegrenzter Wertebereich für Ausgang
- Lösung: Wertebereich begrenzen

Aus Fehlern lernen

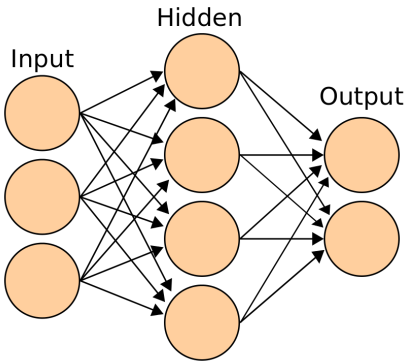
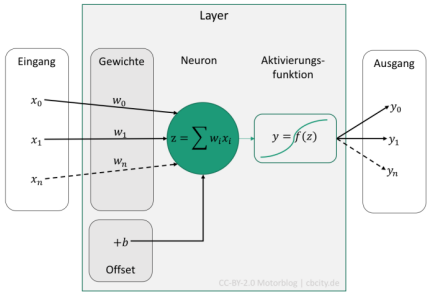


Abbildung: Vernetzte Neuronen

- Fehler werden anteilig auf Eingänge aufgeteilt
- Anpassung der Gewichte durch Funktion zur Fehlerkorrektur
- Lernrate gibt an, wie stark das Netzwerk lernt
- Berechnung der Soll-Werte für vorherige Schicht
- . . .

Ein ‘vollständiges’ Neuron



- Gewichte für alle Eingänge
- Summieren der Eingänge
- Anwenden der Aktivierungsfunktion auf Summe
- Weitergabe dieses Wertes an alle folgenden Neuronen

Abbildung: Modell eines künstlichen Neurons

Den Computer rechnen lassen

- Computer können besser und schneller rechnen
- Neuronale Netze lassen sich mathematisch sehr gut beschreiben
- Anfängliches Problem: ‘Können Maschinen denken?’
  - Klassifizieren von Bildern/Objekten auf Bildern
  - Spracherkennung
  - Selbstfahrende Autos
  - . . .

Matrizenmultiplikation — doch sinnvoll anwendbar

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

Abbildung: Matrizenmultiplikation

- Berechnung der Gewichte und Addition der Ergebnisse durch Produkt von Eingangsmatrix und Gewichtsmatrix
- Anwendung der Aktivierungsfunktion auf jedes Element der Matrix
- Ergebnismatrix wird Eingangsmatrix der nächsten Schicht (bei letzter Schicht Ausgangsmatrix)

Matrizenmultiplikation praktisch

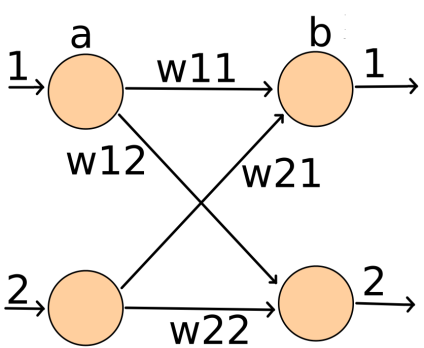


Abbildung: Einfaches neuronales Netz

- $b_1 = a_1 \cdot w_{11} + a_2 \cdot w_{21}$
- $b_2 = a_1 \cdot w_{12} + a_2 \cdot w_{12}$

- $$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \times \begin{pmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{pmatrix}$$

$$= \begin{pmatrix} (a_1 \cdot w_{11}) + (a_2 \cdot w_{21}) \\ (a_1 \cdot w_{12}) + (a_2 \cdot w_{22}) \end{pmatrix}$$

Einfaches praktisches Beispiel

```
class neuralNetwork:
    def __init__(self, inputNodes, hiddenNodes, outputNodes, learningRate):
        self.inodes = inputNodes
        self.hnodes = hiddenNodes
        self.onodes = outputNodes
        self.lr = learningRate

        self.wih = numpy.random.rand(self.hnodes, self.inodes) - 0.5
        self.who = numpy.random.rand(self.onodes, self.hnodes) - 0.5

    def activation_function(self, x):
        return special.expit(x)

    def train(self, inputs_list, targets_list):
        inputs = numpy.array(inputs_list, ndmin=2).T
        targets = numpy.array(targets_list, ndmin=2).T

        hidden_inputs = numpy.dot(self.wih, inputs)
        hidden_outputs = self.activation_function(hidden_inputs)
        final_inputs = numpy.dot(self.who, hidden_outputs)
        final_outputs = self.activation_function(final_inputs)

        output_errors = targets - final_outputs
        hidden_errors = numpy.dot(self.who.T, output_errors)

        self.who += self.lr * numpy.dot((output_errors * final_outputs *
                (1.0 - final_outputs)), numpy.transpose(hidden_outputs))
        self.wih += self.lr * numpy.dot((hidden_errors * hidden_outputs *
                (1.0 - hidden_outputs)), numpy.transpose(inputs))

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

        return final_outputs

    def persist(self, filename):
        to_save = {}
        to_save['input_nodes'] = self.inodes
        to_save['hidden_nodes'] = self.hnodes
        to_save['output_nodes'] = self.onodes
        to_save['learning_rate'] = self.lr

        to_save['input_hidden'] = self.wih.tolist()
        to_save['hidden_output'] = self.who.tolist()

        with open (filename, "w") as outfile:
            json.dump(to_save, outfile)
        outfile.close()
```

Abbildung: Quellcode des neuronalen Netzes

- Neuronales Netz in Python
- Erkennt handgeschriebene Zahlen (28×28 px)
- Training mit mnist-Daten
- Beste Rate: 96,54%
- Berechnung vieler Kombinationen:
  - 50, 100, . . . , 1000 Knoten in Hidden Layer
  - Lernrate von 0.15, 0.16, . . . 0.35
  - Jeweils 10 Versuche mit zufälligen Startparametern
  - 16 Tage, 15:52:12

Wie sieht unser neuronales ‘Gehirn’ von innen aus?

0.021445744255497605, -0.4840623517981541, -0.1826322789883879, 0.27530017694477965, -0.48456187652263943, -0.29147063735211304,

-0.4473445990380996, -0.2868462417762218, -0.34616849444980285, -0.20592702777445654, -0.38949150718959796, -0.05119126994237922,

-0.17928139408080046, 0.21747374131069308, -0.30284744742287734, -0.17728535029609271, 0.27260333701553824, 0.27723740828700877,

-0.2768843231857418, -0.3919568769968604, -0.46311670204300676, -0.34118633276791993, -0.13912295404681457, -0.12173648386798225,

-0.341418701322107, 0.05770749525263829, 0.30226284776861573, -0.4239099124134689, -0.4738543151426418, -0.27325942788608887,

-0.16223374975692634, -0.484837092169024, 0.1450650044725733, 0.1844589614518828, 0.09702195617714789, 0.12960122030266844,

-0.40128423371885485, -0.3448130286263044, -0.19132373586017593, -0.24614659979907189, -0.47258466735892224, -0.32595629764500555,

0.17114918703863258, 0.4422756699445487, -0.06585924395933059, 0.04632302162931292, -0.10103517135913395, -0.0022941229938536842,

-0.09157915534002478, 0.38246966354770473, -0.2071086250035489, 0.3542432335057683, -0.14921448519916597, 0.39885719427759475,

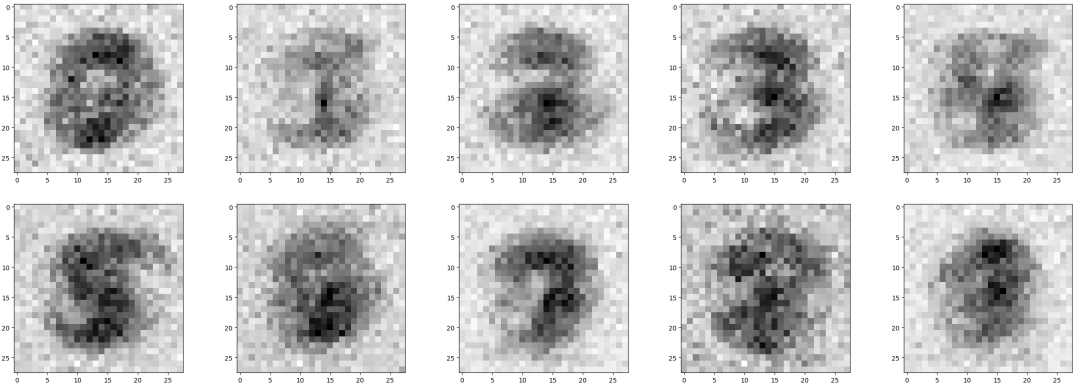
-0.4828458824281246, 0.33046322158359426, -0.31109643465406595, 0.10995946577141308, 0.04080520481731899, -0.2784905327975385,

-0.09173793056414498, -0.1484964753815975, 0.02164213072218011, -0.16741160044338432, -0.3985646872480814, -0.12537705328435447,

0.2671657702575242, 0.16985420654092304, -0.007687313205691635, -0.22554542178080503, 0.4090370774500335, -0.2825936190830333,

0.15204876410981225, 0.21163847660025054, -0.2675590026086629, -0.31462473421182563, -0.39210488437793023, -0.011681643736228552,

Wie sieht unser neuronales ‘Gehirn’ von innen aus?



Quellen

- Neuronale Netze selbst programmieren ISBN:978-3-96009-043-4
- https://upload.wikimedia.org/wikipedia/commons/6/6e/Racknitz_-_The_Turk_3.jpg
- https://upload.wikimedia.org/wikipedia/commons/7/7f/ArtificialNeuronModel_deutsch.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/4/4b/Perceptron-or-task.svg/2000px-Perceptron-or-task.svg.png
- http://berkeleysciencereview.com/wp-content/uploads/2015/06/Artificial_neural_network.svg_.png
- http://data-science-hack.com/wp-content/uploads/2015/09/neuronen-netzwerk2-1-1.png
- http://berkeleysciencereview.com/wp-content/uploads/2015/06/Artificial_neural_network.svg_.png
- http://www.cbcity.de/wp-content/uploads/2016/03/Neuronales_Netz_Layer-770x578.png
- http://www.mathematrix.de/wp-content/uploads/matrixmul2.png
- http://berkeleysciencereview.com/wp-content/uploads/2015/06/Artificial_neural_network.svg_.png