Daniel Scholz · www.taramath.de · daniel@taramath.de · 9. März 2018

Neuronale Netze

Unter dem Begriff des maschinellen Lernens werden Verfahren zusammengefasst, welche ein System anhand von Trainingsdaten anlernen, um daraus anschließend Erfahrungen abzuleiten. (Künstliche) neuronale Netze wiederum bilden einen Zweig des maschinellen Lernens, in welchem menschliche Nervensysteme als biologisches Vorbild verwendet werden. Zentrale Anwendungsbeispiele von neuronalen Netzen finden sich in der Bildverarbeitung und Mustererkennung, in der Analyse von Zeitreihen sowie in der Regelungstechnik. Dieses Kapitel soll eine grundlegende Einführung in das Gebiet der neuronalen Netze bieten, welche bis hin zur Erkennung handgeschriebener Ziffern reichen soll. Wir beginnen in Abschn. 1 mit einer Einleitung, welche insbesondere auf die Analogie zwischen Neuronen eines Nervensystems und mathematischen Modell eingeht. Im folgenden Abschn. 2 wird das grundlegende Modell um verborgene Schichten erweitert. Damit ist ein System geschaffen, welches sich sehr nahe am biologischen Vorbild orientiert. Aus mathematischer Sicht ist es jedoch sinnvoll, an einigen Stellen vom biologischen Vorbild abzuweichen. In Abschn. 3 modifizieren wir daher das (diskrete) Modell, sodass wir ein kontinuierliches Modell erhalten. Einer der wesentlichen Vorteile besteht darin, dass wir damit eine stetig differenzierbare Zielfunktion erhalten, welche es in der Trainingsphase zu minimieren gilt. Genauer beschreiben wir in Abschn. 4 zunächst, was eine Trainingsmenge genau ist, bevor wir in Abschn. 5 ein Gradientenverfahren zur Optimierung der Zielfunktion vorstellen. Damit sind alle Grundlagen geschaffen, um ein neuronales Netz zu trainieren und anschließend zu testen. Als Anwendungsbeispiel stellen wir in Abschn. 6 die maschinelle Erkennung handgeschriebener Ziffern samt numerischer Ergebnisse im Detail vor und gehen dabei auch auf die Konfusionsmatrix ein.

1 Einleitung

Neuronale Netze sind ein Konstrukt aus dem Bereich des maschinellen Lernens, welche sich (wie viele andere Verfahren) die Biologie als Vorbild nehmen. Genauer

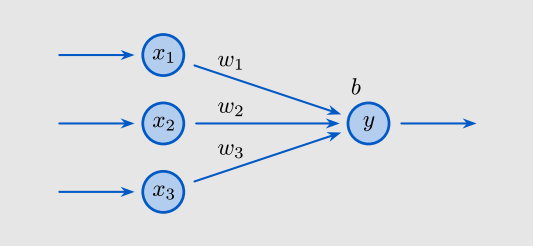


Abb. 1 Beispiel eines einfachen neuronalen Netzes mit drei Neuronen in der Eingangsschicht und einem Neuron in der Ausgangsschicht

wird das Lernverhalten eines Menschen simuliert, indem künstliche (stark vereinfachte) neuronale Netzwerke modelliert werden, wie sie im Gehirn vorgefunden werden können. Nach einer (möglichst intensiven) Lernphase kann das neuronale Netz verwendet werden, um gewisse Aussagen zu treffen. Als zentrales Anwendungsbeispiel untersuchen wir in Abschn. 6 die Mustererkennung, genauer das maschinelle Erkennen von handgeschriebenen Ziffern. Theoretische sowie anwendungsorientiere Grundlagen zu allen Inhalten diesen Kapitels können in diversen Lehrbüchern nachgeschlagen und vertieft werden, etwa in Rey und Wender (2010), Bishop (1995) und Nielsen (2015).

Wir beginnen zunächst damit, grundlegende Prinzipien und Notationen anhand eines sehr einfachen neuronalen Netzes zu beschreiben (Abb. 1): Gegeben sei ein ***Eingangssignal*** in Form eines Vektors $x \in \mathbb{R}^n$, welches n ***Neuronen*** modelliert. Speziell seien zunächst zur boolsche Werte zugelassen, d.h., es gelte zunächst

$$x \;=\; (x_1, \ldots, x_n) \qquad \text{mit} \qquad x_1, \ldots, x_n \;\in\; \{0, 1\}.$$

Dabei bedeutet $x_k = 1$, dass das k-te Neuron ***aktiviert*** ist, bzw. $x_k = 0$, dass das k-te Neuron nicht aktiviert ist.

Alle n Neuronen aus der sogenannten ***Eingangsschicht*** wirken nun gemäß einer Gewichtung

$$w \;=\; (w_1, \ldots, w_n) \qquad \text{mit} \qquad w_1, \ldots, w_n \;\in\; \mathbb{R}$$

auf ein Neuron y in der ***Ausgangsschicht***. Mathematisch gelte dabei

$$y \;=\; \varphi\Big(x^\top \cdot w + b\Big) \;=\; \varphi\left(\sum_{k=1}^{n} x_k \cdot w_k + b\right) \tag{1}$$

mit einem Schwellwert $b \in \mathbb{R}$ sowie der ***Aktivierungsfunktion***

$$\varphi(t) \;=\; \begin{cases} 0 & \text{falls } t \leq 0 \\ 1 & \text{falls } t > 0 \end{cases} . \tag{2}$$

Mit diesem einfachen Modell lässt sich das biologische Verhalten von neuronalen Netzen bereits gut beschreiben: Je nachdem welche Neuronen x_k aus der Eingangsschicht aktiviert sind, ist auch das Neuron y in der Ausgangsschicht aktiviert

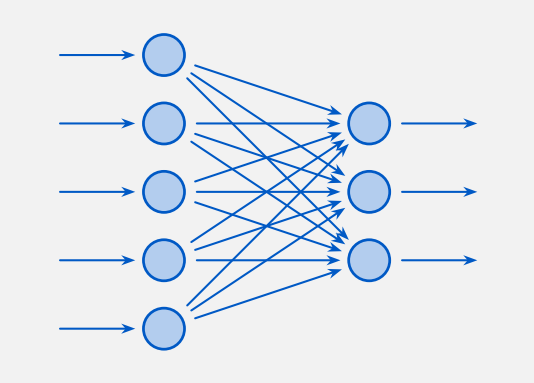


Abb. 2 Beispiel eines neuronalen Netzes mit fünf Neuronen in der Eingangsschicht und drei Neuronen in der Ausgangsschicht

bzw. nicht aktiviert. Dabei ist zu beachten, dass die Gewichte w_k positiv und negativ sein dürfen. Bei einem positiven Gewicht w_k wirkt das k-te Neuron anregend auf das Neuron y, falls das k-te Neuron aktiviert ist. Bei negativen Gewichten wirken die Neuronen entsprechend abstoßend.

Die folgende Aufgabe zeigt, dass wir bereits in der Lage sind, logische Verknüpfungen wie Und und Oder beschreiben zu können:

Aufgabe 1.1 Betrachte das neuronale Netz aus Abb. 1. Bestimme die Gewichte w_1 bis w_3 sowie den Schwellwert b derart, sodass folgendes Verhalten modelliert wird: Das Neuron y soll genau dann aktiviert sein, falls mindestens zwei Neuronen aus der Eingangsschicht aktiviert sind.

?

Anhand der Aufgabe erkennen wir bereits den späteren Nutzen von neuronalen Netzen: Angenommen, es sind Gewichte sowie Schwellwerte bekannt. Dann kann sehr einfach bestimmt werden, wie ein Eingangssignal x auf das Neuron y in der Ausgangsschicht wirkt. Die zentrale Aufgabe besteht später darin, die Gewichte sowie Schwellwerte derart zu bestimmen, dass ein gewünschtes Ziel wie beispielsweise eine Mustererkennung (möglichst gut) erreicht wird.

Um anwendungsorientierte Aufgaben modellieren zu können, müssen wir zunächst die Anzahl der Neuronen (deutlich) erhöhen. Insbesondere besteht auch die Ausgangsschicht in der Regel aus mehreren Neuronen, d.h. aus m Neuronen

$$y = (y_1, \ldots, y_m) \qquad \text{mit} \qquad y_1, \ldots, y_m \in \{0, 1\},$$

wobei $y \in \mathbb{R}^m$ nun als ***Ausgangssignal*** bezeichnet wird. Abb. 2 zeigt ein neuronales Netz mit fünf Neuronen in der Eingangsschicht und drei Neuronen in der Ausgangsschicht. Entsprechend sind $5 \cdot 3 = 15$ Gewichte sowie drei Schwellwerte

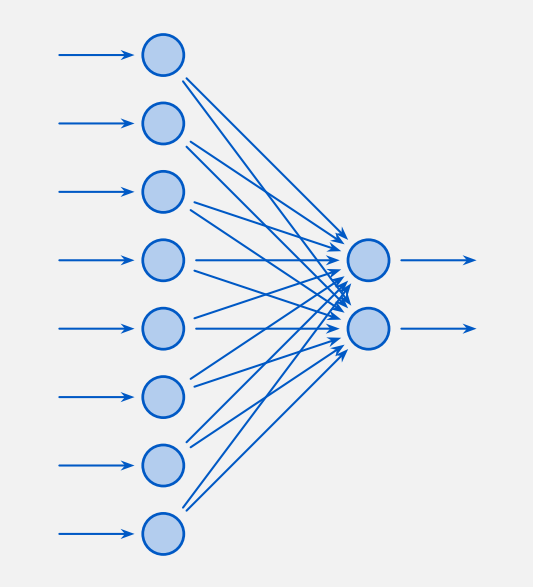


Abb. 3 Beispiel eines neuronalen Netzes mit acht Neuronen in der Eingangsschicht und zwei Neuronen in der Ausgangsschicht

zu bestimmen. Mit diesem Beispiel sind wir in der Lage, $5^2 = 32$ unterschiedliche Eingangssignale x zu verarbeiten und als Ausgangssignal y ergeben sich $3^2 = 8$ unterschiedliche Möglichkeiten.

Im Allgemeinen führen wir nun folgende Notationen ein: Es sei

$$W \in \mathbb{R}^{m \times n}$$

eine $(m \times n)$-Matrix zur Definition der Gewichte und

$$b = (b_1, \ldots, b_m) \in \mathbb{R}^m$$

sei ein Vektor zur Definition der Schwellwerte. Zudem gelte

$$\Phi(z) = \Big(\varphi(z_1), \ldots, \varphi(z_m)\Big) \tag{3}$$

für beliebige Vektoren $z = (z_1, \ldots, z_m) \in \mathbb{R}^m$ unter Verwendung der Aktivierungsfunktion $\varphi(t)$ aus Gl. (2). Damit erhalten wir analog der Berechnungen zuvor

$$y = \Phi\Big(W \cdot x + b\Big) \in \mathbb{R}^m \tag{4}$$

als Vektor der m Neuronen in der Ausgangsschicht in Abhängigkeit von x. Wir demonstrieren das Vorgehen anhand eines kleinen Beispiels:

Beispiel 1.1 (Teilbarkeit) *Die Aufgabe in diesem Beispiel besteht darin, ein neuronales Netz zu entwerfen, welchem als Eingangssignal x eine natürliche Zahl kleiner gleich 255 übergeben wird. Als Ergebnis y soll ausgegeben werden, ob x durch Zwei und ob x durch Vier teilbar ist.*

*Um dieses Ziel zu erreichen, kodieren wir zunächst die natürliche Zahl in Binärdarstellung. Um natürliche Zahlen bis 255 abbilden zu können, benötigen wir acht **Bits**, sodass*

$$x = (x_1, \ldots, x_8) = (0, 1, 0, 1, 1, 0, 0, 0)$$

beispielsweise eine 26 repräsentiert (wobei x_1 die 2^0 und x_8 die 2^7 darstellt). Mit anderen Worten besteht die Eingangsschicht somit aus $n = 8$ Neuronen, welche zusammen eine natürliche Zahl zwischen 0 und 255 darstellen.

Die Ausgabeschicht modellieren wir durch einen Vektor

$$y = (y_1, y_2) \in \{0, 1\}^2$$

mit folgender Bedeutung:

(1) *Es gelte $y_1 = 1$ genau dann, wenn x durch Zwei teilbar ist.*

(2) *Es gelte $y_2 = 1$ genau dann, wenn x durch Vier teilbar ist.*

Abb. 3 veranschaulicht das neuronale Netz zu diesem Beispiel. Zusammengefasst sind schließlich 16 Gewichte sowie zwei Schwellwerte zu bestimmen.

Weiterhin sei bemerkt, dass sich (mit gewissen Vorkenntnissen) die Aufgabe anhand der Binärzahl x einfach lösen lässt: Angenommen, es gilt $x_1 = 0$, dann ist die Zahl x (als Dezimalzahl) durch Zwei teilbar. Gilt zusätzlich $x_2 = 0$, dann ist x sogar durch Vier teilbar.

*Falls diese Vorkenntnisse allerdings nicht bekannt sind, so kann das neuronale Netz trainiert werden, sodass die Aufgabe (mit einer gewissen Wahrscheinlichkeit) korrekt gelöst wird. Der Begriff **Trainieren** bedeutet in diesem Zusammenhang, dass die Gewichte und Schwellwerte derart bestimmt werden, sodass die Aufgabe (möglichst genau) gelöst wird.*

Aufgabe 1.2 Bestimme die Gewichte und Schwellwerte aus der Aufgabe zuvor, sodass das neuronale Netz für alle Eingabedaten (d.h. für alle Dezimalzahlen zwischen 0 und 255) ein korrektes Ergebnis y liefert.

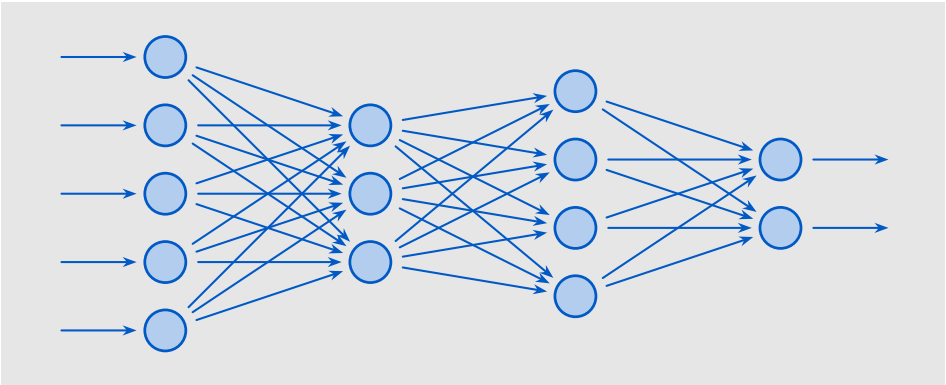


Abb. 4 Beispiel eines neuronalen Netzes mit zwei verborgenen Schichten. Die Eingangsschicht besteht aus fünf Neuronen, die Ausgangsschicht aus zwei Neuronen

2 Verborgene Schichten

Im Abschnitt zuvor haben wir neuronale Netze mit Eingangs- und Ausgangsschicht kennengelernt. In vielen Anwendungsfällen stellt es sich jedoch als äußerst sinnvoll heraus, ***verborgene Schichten*** einzuführen.

Genauer definieren wir zwischen Eingangs- und Ausgangsschicht weitere Schichten mit einer jeweils zuvor festgelegten Anzahl an Neuronen. Abb. 4 zeigt beispielsweise ein neuronales Netz mit zwei verborgenen Schichten: Die erste verborgene Schicht besteht aus drei Neuronen, die zweite verborgene Schicht aus vier Neuronen.

Dabei lässt sich den verborgenen Schichten kaum eine inhaltliche Erklärung zuweisen, vielmehr zeigen die späteren Anwendungen das Potenzial dieser zusätzlichen Schichten. Zudem ist auch diese Idee dem biologischen Vorbild entnommen, denn (echte) neuronale Netze sind über viele Schichten miteinander verknüpft.

Auch mathematisch lassen sich die verborgenen Schichten leicht einführen (wobei wir zur übsichtlicheren Darstellung bewusst nicht auf die Dimensionen der Vektoren und Matrizen eingehen): Wieder sei x ein Vektor zur Definition der Neuronen in der Eingangsschicht und y sei der Vektor der Neuronen in der Ausgangsschicht. Bei einem neuronalen Netz mit s verborgenen Schichten benötigen wir nun insgesamt $s+1$ Matrizen zur Definition der Gewichte, nämlich

$$\omega \;=\; \{W_{(0)},\; \ldots,\; W_{(s)}\}, \tag{5}$$

sowie $s+1$ Vektoren zur Definition der Schwellwerte, genauer

$$\beta \;=\; \{b_{(0)},\; \ldots,\; b_{(s)}\}. \tag{6}$$

In Abhängigkeit von ω und β sowie in Abhängigkeit von x lässt sich y rekursiv

bestimmen: Wir definieren $z_{(0)} = x$ und berechnen

$$z_{(k+1)} \;=\; \Phi\Big(W_{(k)} \cdot z_{(k)} + b_{(k)}\Big)$$

für $k = 0, \ldots, s$ unter Verwendung der Funktion $\Phi(z)$ aus Gl. (3). Als Ergebnis erhalten wir damit $y = z_{(s+1)}$.

Ausblick *Um auf die rekursive Darstellung zu verzichten, nutzen wir später teilweise auch folgende äquivalente Formulierung: Für ein neuronales Netz mit $s = 2$ verborgenen Schichten gilt $y = f(x, \omega, \beta)$ mit*

$$f(x, \omega, \beta) \;=\; \Phi\Big(W_{(2)} \cdot \Phi\Big(W_{(1)} \cdot \Phi\Big(W_{(0)} \cdot x + b_{(0)}\Big) + b_{(1)}\Big) + b_{(2)}\Big). \qquad (7)$$

Dabei wurde die Funktion $f(x, \omega, \beta)$ bereits an dieser Stelle neben $x \in \mathbb{R}^n$ bewusst auch in Abhängigkeit von ω und β definiert. Dabei ist stets zu beachten, dass ω eine Liste von $s+1$ Matrizen unterschiedlicher Dimensionen und β eine Liste von $s+1$ Vektoren unterschiedlicher Länge ist.

Zur Festlegung der Anzahl der verborgenen Schichten sowie zur Definition der Anzahl der Neuronen pro Schicht führen wir folgende Notation ein:

Notation 2.1 *Das **Design** eines neuronalen Netzes legen wir fest durch einen Vektor*

$$(n_0, \ldots, n_s, n_{s+1}) \;\in\; \mathbb{N}^{s+2}.$$

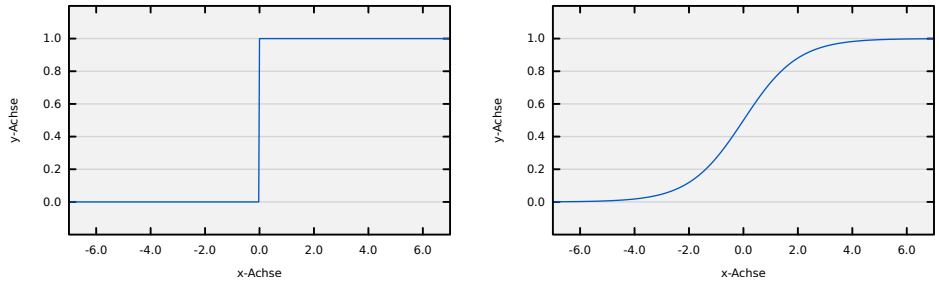
Dabei definiert $n_0 \in \mathbb{N}$ die Anzahl der Neuronen der Eingangsschicht, $n_{s+1} \in \mathbb{N}$ definiert die Anzahl der Neuronen der Ausgangsschicht und für $i = 1, \ldots, s$ legt $n_i \in \mathbb{N}$ die Anzahl der Neuronen in der i-ten verborgenen Schicht fest.

Dank der Beschreibung des Designs eines neuronalen Netzes anhand eines Vektors wird direkt geklärt, wie viele verborgene Schichten vorhanden sind und wie viele Neuronen pro Schicht verwendet werden. Damit sind auch direkt die Dimensionen aller Matrizen aus ω und die Längen aller Vektoren aus β eindeutig festgelegt.

Beispiel 2.1 *Das Design des neuronalen Netzes zur Teilbarkei aus Beispiel 1.1 wird gegeben durch*

$$(8, 2).$$

Mit dieser Notation wird direkt deutlich, dass die Eingangsschicht aus acht Neuronen besteht, dass das neuronale Netz keine verborgenen Schichten hat und dass die Ausgangsschicht zwei Neuronen besitzt.



a Aktivierungsfunktion als Stufenfunktion

b Aktivierungsfunktion als Sigmoidfunktion

Abb. 5 Beispiele für Aktivierungsfunktionen $\varphi(t)$: Stufenfunktion aus Gl. (2) in **a** und Sigmoidfunktion aus Gl. (8) in **b**. Jeweils ist $\varphi(t)$ monoton steigend und es gilt $\varphi(t) \in [0, 1]$ für alle $t \in \mathbb{R}$

?

Aufgabe 2.1 Angenommen, das Design eines neuronalen Netzes sei gegeben durch

$$(6,\ 4,\ 2,\ 5).$$

Welche Dimensionen haben dann die Matrizen aus ω und welche Längen haben die Vektoren aus β?

Abschließend sei bemerkt, dass sich in praktischen Anwendungen meist weder eine Aussage darüber treffen lässt, wie viele verborgenen Schichten verwendet werden sollten, noch wie viele Neuronen pro verborgener Schicht definiert werden sollten. Diese Fragen werden daher häufig anhand bekannter Daten empirisch beantwortet.

3 Kontinuierliches Modell

Obwohl das diskrete Modell mit den boolschen Variablen aus den beiden Abschnitten zuvor die Realität sinnvoll nachbildet, ist aus mathematischer Sicht ein kontinuierliches Modell deutlich besser geeignet: Erstens wird es im Folgenden deutlich einfacher sein, eine stetig differenzierbare Zielfunktion zu minimieren, und zweitens werden auch die Eingabedaten praktischer Anwendungen häufig durch kontinuierliche Werte gegeben.

Im Detail wird das bisherige Modell an zwei Stellen verallgemeinert, nämlich bei den Definitionen des Eingangssignals sowie der Aktivierungsfunktion: Das Eingangssignal $x \in \mathbb{R}^n$ sei nun nicht mehr ein boolscher Vektor, sondern es gelte

$$x \ = \ (x_1, \ldots, x_n) \qquad \text{mit} \qquad x_1, \ldots, x_n \ \in \ [0, 1].$$

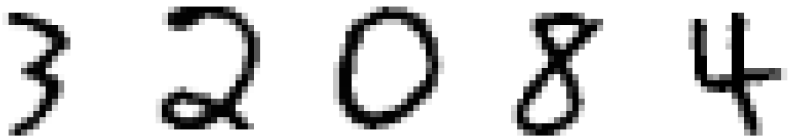


Abb. 6 Beispiele für handgeschriebene Ziffern, welche jeweils aus 28×28 Pixel bestehen

Dies bedeutet, dass die Aktivierungen der n Neuronen jeweils durch eine kontinuierliche Zahl aus dem Intervall $[0, 1]$ modelliert wird. Zudem wählen wir als Aktivierungsfunktion im Folgenden nicht mehr die Stufenfunktion (Abb. 5a), sondern die ***Sigmoidfunktion***

$$\varphi(t) \;=\; \frac{1}{1+\exp(-t)} \tag{8}$$

(Abb. 5b). Analog der Notationen zuvor führen wir mit

$$\Phi(z) \;=\; \Big(\varphi(z_1),\, \ldots,\, \varphi(z_m)\Big) \tag{9}$$

die komponentenweise Anwendung der Sigmoidfunktion auf einen beliebigen Vektor $z = (z_1, \ldots, z_m) \in \mathbb{R}^m$ ein.

Damit haben wir schließlich das kontinuierliche Modell eines neuronalen Netzes beschrieben, welches wir im weiteren Verlauf des Kapitels verwenden werden. Bevor wir uns ein zentrales Beispiel dazu ansehen, fassen wir das Modell nochmals zusammen:

Zusammenfassung 3.1 Gegeben sei ein neuronales Netz bestehend aus Eingangsschicht, $s \geq 0$ verborgenen Schichten und Ausgangsschicht. Dabei seien die zugehörigen Gewichte $\omega = \{W_{(0)}, \ldots, W_{(s)}\}$ sowie die Schwellwerte $\beta = \{b_{(0)}, \ldots, b_{(s)}\}$ durch entsprechende Matrizen bzw. Vektoren bekannt.

Um ein (beliebiges) Eingangssignal $x \in [0, 1]^n$ auszuwerten, setzen wir $z_{(0)} = x$ und berechnen

$$z_{(k+1)} \;=\; \Phi\Big(W_{(k)} \cdot z_{(k)} + b_{(k)}\Big) \qquad \text{für } k = 0, \ldots, s.$$

Dabei ist $\Phi(z)$ die komponentenweise Anwendung der Sigmoidfunktion aus den Gln. (8) bzw. (9) und als Ergebnis erhalten wir das Ausgangssignal $y = z_{(s+1)}$.

Natürlich ist die Bestimmung geeigneter Gewichte ω und Schwellwerte β von zentraler Bedeutung, worauf wir in den folgenden Abschnitten im Detail eingehen werden. Zuvor betrachten wir jedoch ein wichtiges Beispiel:

Beispiel 3.1 (Ziffererkennung) *In diesem Beispiel wollen wir zeigen, wie ein neuronales Netz zur Erkennung handgeschriebener Ziffern verwendet werden kann. Genauer soll das Eingangssignal x ein Schwarz-Weiß-Bild einer handgeschriebener Ziffer repräsentieren und das Ausgangssignal y des neuronalen Netzes sei ein Vektor, welcher angibt, um welche Ziffer es sich im Bild handelt.*

Die genaue Vorgehensweise, welche wir im Folgenden mehrfach aufgreifen werden, sei folgendermaßen: Wir verwenden Schwarz-Weiß-Bilder handgeschriebener Ziffern, welche jeweils aus 28×28 Pixel bestehen und welche bereits auf eine geeignete Größe skaliert wurden (Abb. 6). Das Eingangssignal $x = (x_1, \ldots, x_n)$ besteht daher aus einem Vektor der Länge $n = 28 \cdot 28 = 784$, wobei die Pixel des Bildes von x_1 oben links nach x_{784} unten rechts der Reihe nach angeordnet werden. Dabei bedeutet ein weißer Pixel im Bild als Wert eine 0 im Eingangssignal und ein schwarzer Pixel im Bild bedeutet eine 1 im Eingangssignal. Alle Graustufen dazwischen werden auf entsprechende Werte aus dem Intervall $[0, 1]$ gesetzt, sodass zusammengefasst jedes Eingangssignal $x = [0, 1]^{784}$ ein Schwarz-Weiß-Bild bestehend aus 28×28 Pixel beschreibt.

Für das Ausgangssignal gibt es mehrere Möglichkeiten, um eine Ziffer zwischen 0 und 9 zu beschreiben. Wir entscheiden uns dabei für folgende Idee: Als Ausgangsschicht wählen wir zehn Neuronen, d.h. $y = (y_1, \ldots, y_{10}) \in [0, 1]^{10}$, wobei jedes Neuron genau eine Ziffer repräsentiert. Das neuronale Netz erkennt nun diejenige Ziffer im Bild, dessen zugehöriges Neuron den größten Wert annimmt. Mathematische formulieren lässt sich dies folgendermaßen: Das neuronale Netz hat im Bild des Eingangssignals x die Ziffer $p \in \{0, \ldots, 9\}$ erkannt, falls

$$y_{p+1} \;=\; \max\{y_1, \ldots, y_{10}\}$$

gilt. Mit anderen Worten ist y_{p+1} eine Wahrscheinlichkeit dafür, dass es sich im Bild um die Ziffer p handelt. Entsprechend wird die Ziffer mit der größten Wahrscheinlichkeit gewählt.

Das Beispiel zuvor werden wir im weiteren Verlauf diesen Kapitels noch mehrfach analysieren und erweitern. Insbesondere sei nochmals darauf hingewiesen, das sowohl die Anzahl der verborgenen Schichten als auch die Gewichte ω und Schwellwerte β von zentraler Bedeutung dafür sind, ob das neuronale Netz die Ziffern korrekt erkennt oder eben nicht. Darauf werden wir in den folgenden Abschnitten weiter eingehen.

Aufgabe 3.1 Ein weiterer Vorteil der Sigmoidfunktion (8) im Vergleich zur Stufenfuktion besteht darin, dass ihre Ableitung vergleichsweise einfach bestimmt werden kann: Zeige, dass

$$\varphi'(t) \;=\; \varphi(t) \cdot \Big(1 - \varphi(t)\Big) \tag{10}$$

gilt. Auch diese Aussage werden wir später mehrfach anwenden.

4 Trainingsmenge

Um ein neuronales Netz geeignet auszulegen, muss zunächst das Design gemäß Notation 2.1 festgelegt werden, d.h., es ist die Anzahl der verborgenen Schichten sowie die Anzahl der Neuronen pro Schicht zu definieren. Wie zuvor bereits erwähnt, spielen hier Erfahrungswerte eine große Rolle und es kann kaum eine allgemeine Hilfestellung dazu gegeben werden.

Nachdem das Design des neuronalen Netzes festgelegt wurde, sind die Gewichte ω und Schwellwerte β derart zu bestimmen, dass das neuronale Netz möglichst zuverlässige Ergebnisse liefert. Die allgemeine Vorgehensweise dazu ist folgende: Anhand einer (sehr) großen Menge von Eingangssignalen mit jeweils bekanntem Ausgangssignal wird das neuronale Netz ***trainiert***, d.h., es werden die Gewichte ω und die Schwellwerte β bestimmt. Genauer sei

$$\mathcal{X} \;=\; \Big\{ \big(x_{(1)}, y_{(1)}\big)\,,\; \ldots,\; \big(x_{(r)}, y_{(r)}\big) \Big\}$$

eine ***Trainingsmenge*** bestehend aus r Paaren von Eingangs- und Ausgangssignalen des neuronalen Neztes, wobei $y_{(i)} \in \mathbb{R}^m$ geweils das bekannte bzw. gewünschte Ausgangssignal zum Eingangssignal $x_{(i)} \in \mathbb{R}^n$ ist.

Beispiel 4.1 *Eine Trainingsmenge zur Ziffererkennung (Beispiel 3.1) besteht aus einer großen Anzahl von Bildern handgeschriebener Ziffern, welche wie zuvor jeweils durch ein Eingangssignal x repräsentiert werden. Zudem sei die Ziffer zu jedem Bild bekannt, sodass zu jedem Bild das zugehörige (gewünschte) Ausgangssignal y angegeben werden kann. Genauer gelte*

$$y \;=\; (1,\,0,\,0,\,0,\,0,\,0,\,0,\,0,\,0,\,0),$$

falls es sich beim zugehörigen Eingangssignal x um ein Bild mit der Ziffer 0 *handelt, und analog gelte beispielsweise*

$$y \;=\; (0,\,0,\,0,\,0,\,0,\,0,\,0,\,1,\,0,\,0),$$

falls es sich beim zugehörigen Eingangssignal x um ein Bild mit der Ziffer 7 *handelt. Damit ist eine Trainingsmenge $\mathcal{X}$ zur Ziffererkennung festgelegt, wobei durchaus viele Tausend Bilder herangezogen werden sollten.*

Unter Verwendung der Trainingsmenge können wir die Bestimmung der Gewichte ω und Schwellwerte β nun als Optimierungsproblem formulieren. Dazu sei

$$f(x, \omega, \beta) \tag{11}$$

das Ausgangssignal des neuronalen Netzes in Abhängigkeit von x, ω und β, s. auch Gl. (7) und Zusammenfassung 3.1. Damit besteht die Bestimmung der Gewichte ω

sowie der Schwellwerte β darin, die Zielfunktion

$$g(\omega, \beta) \;=\; \sum_{(x,y)\in\mathcal{X}} \frac{1}{2} \cdot \Big\| f(x, \omega, \beta) - y \Big\|_2^2 \tag{12}$$

zu minimieren, wobei $\| \cdot \|_2$ die Euklidische Norm sei. Ein globales Minimum der Funktion $g(\omega, \beta)$ bedeutet somit eine optimale Bestimmung der Gewichte und Schwellwerte bezüglich der Trainingsmenge $\mathcal{X}$ und der Euklidischen Norm.

Schließlich sei bemerkt, dass die Funktion $f(x, \omega, \beta)$ dank des kontinuierlichen Modells stetig differenzierbar ist (in allen Variablen). Damit ist aber auch $g(\omega, \beta)$ stetig differenzierbar und es können entsprechende Methoden der Optimierung angewandt werden, beispielsweise das Gradientenverfahren, wie wir im folgenden Abschnitt genauer untersuchen werden.

Ausblick *Neben der Trainingsmenge $\mathcal{X}$ ist in der Regel auch eine **Testmenge** $\mathcal{T}$ analog zu $\mathcal{X}$ bestehend aus Paaren von Eingangs- und Ausgangssignalen des neuronalen Netztes gegeben: Anhand von $\mathcal{X}$ wird das neuronale Netz trainiert und anhand von $\mathcal{T}$ wird die Güte der bestimmten Gewichte und Schwellwerte getestet.*

Es ergibt sich somit ein Prozentsatz der korrekten Ergebnisse bezogen auf die Testmenge $\mathcal{T}$. Ist dieser Prozentsatz zu klein, so ist entweder das Design des neuronalen Netzes zu überdenken oder aber es ist eine größer Trainingsmenge heranzuziehen.

5 Gradientenverfahren

In diesem Abschnitt wollen wir erläutern, wie die Zielfunktion (12), d.h.

$$g(\omega, \beta) \;=\; \sum_{(x,y)\in\mathcal{X}} \frac{1}{2} \cdot \Big\| f(x, \omega, \beta) - y \Big\|_2^2,$$

geeignet minimiert werden kann. Offenbar handelt es sich bei $g(\omega, \beta)$ dank des kontinuierlichen Modells um eine stetig differenzierbare Funktion, sodass gängige Verfahren der nichtlinearen oder globalen Optimierung angewandt werden könnten. Jedoch ist die Anzahl der Variablen in den meisten Anwendungsfällen so groß, dass es kaum möglich ist, ein globales Minimum zu finden.

Aufgabe 5.1 Gegeben sei ein neuronales Netzwerk mit dem Design

(784, 30, 10),

(Notation 2.1). Bestimme die Anzahl der Variablen der Zielfunktion $g(\omega, \beta)$. Welche Rolle spielt dabei die Größe der Trainingsmenge $\mathcal{X}$?

Angewandt wird daher meist eine Variante des ***Gradientenverfahrens***: Ausgehend von einer Startlösung $(\omega_{(0)}, \beta_{(0)})$ wird

$$\Big(\omega_{(k+1)}, \beta_{(k+1)}\Big) \;=\; \Big(\omega_{(k)}, \beta_{(k)}\Big) \;-\; \frac{t}{r} \cdot \nabla g\Big(\omega_{(k)}, \beta_{(k)}\Big) \tag{13}$$

für $k = 0, \ldots, q-1$ berechnet. Dabei ist r die Anzahl der Elemente der Trainingsmenge $\mathcal{X}$ und weiter sei $t > 0$ eine feste ***Schrittweite*** bzw. ***Lernrate*** sowie $q \in \mathbb{N}$ eine zuvor festgelegt Anzahl an Iterationen.

Sofern keine weiteren Vorkenntnisse eingebracht werden können, wird $(\omega_{(0)}, \beta_{(0)})$ meist zufällig gewählt: Genauer werden alle Einträge der Matrizen aus $\omega_{(0)}$ sowie der Vektoren aus $\beta_{(0)}$ auf zufällige Werte aus dem Intervall $[-1, 1]$ gesetzt.

Das Problem besteht damit schließlich noch in der Berechnung des Gradienten $\nabla g(\omega, \beta)$, welches wir am Beispiel ohne verborgene Schichten herleiten wollen:

Herleitung *Angenommen, ein neuronales Netz besitzt keine verborgenen Schichten. Dann gilt $\omega = \{W\}$ für eine Matrix $W \in \mathbb{R}^{m \times n}$ sowie $\beta = \{b\}$ für einen Vektor $b \in \mathbb{R}^m$. Zudem folgt, s. auch Gl.(4),*

$$f(x, \omega, \beta) \;=\; \Phi(W \cdot x + b) \;\in\; \mathbb{R}^m,$$

sodass wir den Gradienten von

$$g(\omega, \beta) \;=\; g(W, b) \;=\; \sum_{(x,y) \in \mathcal{X}} \frac{1}{2} \cdot \Big\| \Phi(W \cdot x + b) - y \Big\|_2^2$$

zu bestimmen haben.

Dazu schreiben wir $y = (y_1, \ldots, y_m) \in \mathbb{R}^m$ und mit $w_1, \ldots, w_m \in \mathbb{R}^n$ bezeichnen wir die Zeilenvektoren der Matrix $W \in \mathbb{R}^{m \times n}$. Unter Verwendung der Kettenregel erhalten wir schließlich

$$\nabla g(\omega, \beta) \;=\; \nabla g(W, b) \;=\; \left(\sum_{(x,y) \in \mathcal{X}} p(W, b, x, y) \cdot x^\top, \; \sum_{(x,y) \in \mathcal{X}} p(W, b, x, y) \right).$$

Dabei gilt

$$p(W, b, x, y) \;=\; \Big(p_1(W, b, x, y), \;\ldots,\; p_m(W, b, x, y) \Big) \;\in\; \mathbb{R}^m$$

mit

$$p_i(W, b, x, y) \;=\; \Big(\varphi\left(x^\top \cdot w_i + b\right) - y_i \Big) \cdot \varphi'\left(x^\top \cdot w_i + b\right)$$

für $i = 1, \ldots, m$. Unter Verwendung der für die Sigmoidfunktion $\varphi(t)$ gültigen Regel

$$\varphi'(t) \;=\; \varphi(t) \cdot \Big(1 - \varphi(t)\Big) \qquad \text{für alle } t \in \mathbb{R},$$

s. auch Gl. (10), folgt weiter

$$p_i(W,b,x,y) \;=\; \Big(\varphi\left(x^\top \cdot w_i + b\right) - y_i\Big) \cdot \varphi\left(x^\top \cdot w_i + b\right) \cdot \Big(1 - \varphi\left(x^\top \cdot w_i + b\right)\Big).$$

Dabei ist insgesamt zu beachten, dass

$$\sum_{(x,y)\in\mathcal{X}} p(W,b,x,y) \cdot x^\top \;\in\; \mathbb{R}^{m\times n} \qquad \text{und} \qquad \sum_{(x,y)\in\mathcal{X}} p(W,b,x,y) \;\in\; \mathbb{R}^m$$

gilt. Zusammenfassend haben wir damit den Gradienten $\nabla g(\omega,\beta)$ für neuronale Netze ohne verborgene Schichten bestimmt und können somit das Gradientenverfahren wie zuvor beschrieben anwenden.

Falls das neuronale Netz eine oder mehrere verborgene Schichten enthält, so kann der Gradient auf ähnliche Art und Weise bestimmt werden. Wir verzichten jedoch auf eine detaillierte Darstellung dazu und verweisen für den allgemeinen Fall auf die folgende im Kern nicht schwierige, in der Notation jedoch durchaus herausfordernde Aufgabe:

Aufgabe 5.2 Bestimme den Gradienten $\nabla g(\omega,\beta)$ der Zielfunktion $g(\omega,\beta)$, falls das zugehörige neuronale Netz eine beliebige Anzahl von verborgenen Schichten enthält. Beginne zunächst mit einer vorborgenen Schicht und nur zwei Neuronen pro Schicht.

Wir sind nun in der Lage, das Gradientenverfahren (13) zur Optimierung der Zielfunktion $g(\omega,\beta)$ anzuwenden und können damit die Gewichte ω sowie Schwellwerte β eines neuronalen Netzes unter Verwendung einer Trainingsmenge $\mathcal{X}$ bestimmen. Diese Vorgehensweise wird im Rahmen von neuronalen Netzen häufig auch als ***Backpropagation*** bezeichnet.

Es sei jedoch bemerkt, dass das Gradientenverfahren bestenfalls in einem lokalen Minimum terminiert. Eine Aussage zur Güte insgesamt ist damit nicht gegeben bzw. hierzu ist auf eine Testmenge $\mathcal{T}$ zurückzugreifen.

Schließlich führen wir eine weitere Verinfachung ein, welche die Laufzeiten der ***Trainingsphase*** maßgeblich verringern kann: Da der Gradient $\nabla g(\omega,\beta)$ im Wesentlichen aus einer Summe über den Elementen der Trainingsmenge $\mathcal{X}$ besteht, nämlich

$$\nabla g(\omega,\beta) \;=\; \sum_{(x,y)\in\mathcal{X}} \nabla\left(\frac{1}{2}\cdot\Big\|f(x,\omega,\beta) - y\Big\|_2^2\right),$$

kann ein ***stochastisches*** Gradientenverfahren eingesetzt werden. Genauer wählen wir eine ***Losgröße*** bzw. ***Batchsize*** $v \in \mathbb{N}$ mit $1 \leq v \leq r$, wobei r wie zuvor die Anzahl der Elemente der Trainingsmenge $\mathcal{X}$ ist. Nun sei

$$\mathcal{X}_v \;\subset\; \mathcal{X}$$

eine Teilmenge bestehend aus v zufällig gewählten Elementen der Trainingsmenge $\mathcal{X}$. Damit verwenden wir die Approximation

$$\nabla g(\omega, \beta) \;\approx\; \frac{r}{v} \cdot \sum_{(x,y)\in\mathcal{X}_v} \nabla \left(\frac{1}{2} \cdot \left\| f(x, \omega, \beta) - y \right\|_2^2 \right). \tag{14}$$

Das stochastische Gradientenverfahren liefert damit insbesondere dann Vorteile in der Laufzeit der Trainingsphase, falls die Trainingsmenge $\mathcal{X}$ sehr groß ist. Schließlich fassen wir die Trainingsphase zusammen:

Zusammenfassung 5.1 Gegeben sei ein neuronales Netz samt Design (Notation 2.1). Weiter sei

$$\mathcal{X} \;=\; \Big\{ \big(x_{(1)}, y_{(1)}\big),\; \ldots,\; \big(x_{(r)}, y_{(r)}\big) \Big\}$$

eine Trainingsmenge bestehend aus r Paaren von Eingangs- und Ausgangssignalen. Zudem sei $t > 0$ eine Lernrate, $v \in \mathbb{N}$ mit $1 \leq v \leq r$ sei eine Losgröße und q sei die Anzahl an Iterationen.

Initialisiere die Einträge aller Gewichte und Schwellwerte der Startlösung $(\omega_{(0)}, \beta_{(0)})$ mit zufälligen Werten aus dem Intervall $[-1, 1]$. Anschließend berechne

$$\Big(\omega_{(k+1)}, \beta_{(k+1)}\Big) \;=\; \Big(\omega_{(k)}, \beta_{(k)}\Big) \;-\; \frac{t}{v} \cdot \sum_{(x,y)\in\mathcal{X}_v} \nabla \left(\frac{1}{2} \cdot \left\| f(x, \omega, \beta) - y \right\|_2^2 \right)$$

für $k = 0, \ldots, q-1$, wobei $\mathcal{X}_v$ in jeder Iteration eine neue zufällig gewählte Teilmenge von $\mathcal{X}$ bestehend aus v Elementen sei.

Die Trainingsphase terminiert schließlich mit den Gewichten und Schwellwerten $(\omega_{(q)}, \beta_{(q)})$, dessen Güte anhand einer Testmenge $\mathcal{T}$ evaluiert werden sollte.

#

Als Größenordnung kann für die Losgröße v beispielsweise etwa ein Hundertstel von der Anzahl r der Trainingsdaten gewählt werden. Zudem sollte die Anzahl der Iterationen q mindestens ein Zehntel von r sein.

Abschließend sei nochmals bemerkt, dass das Ergebnis der Trainingsphase nicht deterministisch ist: Erstens werden die Startwerte zufällig generiert und zweitens werden beim stochastischen Gradientenverfahren zufällige Teilmengen $\mathcal{X}_v$ von $\mathcal{X}$ gewählt.

Lernrate t	3
Losgröße v	200
Anzahl der Iterationen q	10 000

Tab. 1 Parameter der Trainingsphase zur Ziffererkennung

6 Anwendungsbeispiel Ziffererkennung

Ein zentrales Anwendungsgebiet von neuronalen Netzen ist die Mustererkennung in Bildern. In diesem Abschnitt greifen wir daher die Beispiele 3.1 und 4.1 zur Erkennung handgeschriebener Ziffern auf und präsentieren insbesondere numerische Ergebnisse dazu.

Wir haben in Beispiel 3.1 bereits erläutert, wie wir ein Schwarz-Weiß-Bild einer handgeschriebenen Ziffer als Eingangssignal $x \in [0,1]^{784}$ kodieren können. Zudem besitzt die Ausgangsschicht zehn Neuronen, d.h. $y = (y_1, \ldots, y_{10}) \in \mathbb{R}^{10}$, wobei der größte Eintrag des Vektors y darüber entscheidet, um welche Ziffer es sich im Eingangssignal x bzw. im zugehörigen Bild handelt.

Bislang haben wir jedoch keine weiteren Angaben zum Design des zugehörigen neuronalen Netzes gemäß Notation 2.1 gemacht. Wir haben allerdings bereits mehrfach erwähnt, dass es sich dabei um reine Erfahrungswerte handelt. Wir entscheiden uns für eine verborgene Schicht (d.h. $s = 1$) bestehend aus 30 Neuronen, sodass das zugehörige Design schließlich gegeben wird durch

$$(784,\ 30,\ 10).$$

Als Grundlage für unsere Untersuchungen nutzen wir die ***MNIST*** Datenbank für handgeschriebene Zahlen, s. auch LeCun et al. (1998) sowie Nielsen (2015). Wir erhalten damit eine Trainingsmenge $\mathcal{X}$ bestehend aus $r = 60\,000$ Elementen sowie eine Testmenge $\mathcal{T}$ bestehend aus 10 000 Elementen.

Unter Verwendung der Parameter aus Tab. 1 liefert uns die Trainingsphase aus Zusammenfassung 5.1 Gewichte und Schwellwerte, sodass anschließend etwa

$$95\,\%$$

der insgesamt 10 000 Ziffern der Testmenge $\mathcal{T}$ korrekt erkannt werden. Dies ist ein sehr gutes Ergebnis, sofern berücksichtig wird, dass die Ziffern durchaus unleserlich geschrieben und somit auch von einem Menschen kaum erkannt werden können. In diesem Zusammenhang zeigt Abb. 7 einige Beispiele von Ziffern, welche nicht korrekt erkannt wurden.

Zur Beurteilung der Güte bzw. Leistungsfähigkeit eines neuronalen Netzes wird neben der einfachen Prozentangabe häufig auch eine ***Konfusionsmatrix*** herangezogen. Dabei handelt es sich um eine charakteristische Darstellung zur Visualisierung der Ergebnisse anhand einer quadratischen Matrix $A = (a_{i,j})$: Jede Zeile der Matrix

a Ist 2, erkannt 7 **b** Ist 5, erkannt 3 **c** Ist 9, erkannt 4 **d** Ist 6, erkannt 0 **e** Ist 3, erkannt 5

f Ist 2, erkannt 8 **g** Ist 4, erkannt 9 **h** Ist 9, erkannt 4 **i** Ist 8, erkannt 9 **j** Ist 6, erkannt 0

Abb. 7 Beispiele einiger Ziffern, welche selbst nach umfangreicher Trainingsphase zu den nicht korrekt erkannten Ziffern aus der Testmenge $\mathcal{T}$ gehören

steht für ein bekanntes Resultat und jede Spalte für das Ergebnis des neuronalen Netzes. Genauer beschreibt $a_{i,j}$ die Anzahl der Elemente der Testmenge $\mathcal{T}$, welche das (bekannte) Resultat i haben und als j erkannt wurden. Im idealen Falle, d.h., falls alle Elemente der Testmenge korrekt erkannt werden, sind alle Einträge der Konfusionsmatrix außerhalb der Diagonalen gleich 0.

Am Beispiel der Ziffererkennung ergibt sich daher eine (10×10)-Matrix, in Abb. 8 dargestellt wurde. Es ist abzulesen, dass 980 Elemente der Testmenge einer Ziffer 0 entsprechen, von denen 969 korrekt erkannt wurden. Andererseits wurden 34 Elemente der Testmenge als Ziffer 4 erkannt, obwohl es sich um eine Ziffer 9 handelte. Durch die Darstellung einer Konfusionsmatrix können somit die Defizite des neuronalen Netzes hervorgehoben werden.

Um abschließend einen Eindruck über den Einfluss des Designs zu erhalten, wurde die exakt gleiche Trainingsphase wie zuvor durchgeführt (Tab. 1), nun jedoch zum Design

$$(784,\ 10),$$

d.h. ohne verborgene Schichten. Um Vergleich zu den 95 % zuvor werden damit nur etwa

$$84\,\%$$

der Ziffern der Testmenge $\mathcal{T}$ korrekt erkannt, sodass auf verborgene Schichten im Allgemeinen nicht verzichtet werden sollte.

	1026	1140	1016	1027	1004	866	968	1013	965	975
980	969	0	0	2	0	3	3	1	2	0
1135	0	1116	2	5	0	1	5	2	4	0
1032	13	1	959	8	11	2	9	9	18	2
1010	2	2	17	950	1	12	1	11	12	2
982	2	1	4	1	935	0	10	2	4	23
892	9	1	4	21	4	821	14	3	11	4
958	10	3	4	2	5	13	916	1	4	0
1028	1	7	21	7	6	0	0	970	1	15
974	9	3	5	18	8	7	10	5	903	6
1009	11	6	0	13	34	7	0	9	6	923

Abb. 8 Konfusionsmatrix zur Erkennung hangeschriebener Ziffern nach der Durchführung der zuvor beschriebenen Trainingsphase. Dabei verdeutlicht die Farbgebung die größten Defizite des neuronalen Netzes

Literatur

C.M. Bishop, 1995. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, 1. Auflage.

Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**: 2278–2324.

M.A. Nielsen, 2015. *Neural Networks and Deep Learning.* Determination Press. Version vom 12. Dezember 2016.

G.D. Rey, K.F. Wender, 2010. *Neuronale Netze: Eine Einführung in die Grundlagen, Anwendungen und Datenauswertung.* Hogrefe Verlag, Göttingen, 2. Auflage.