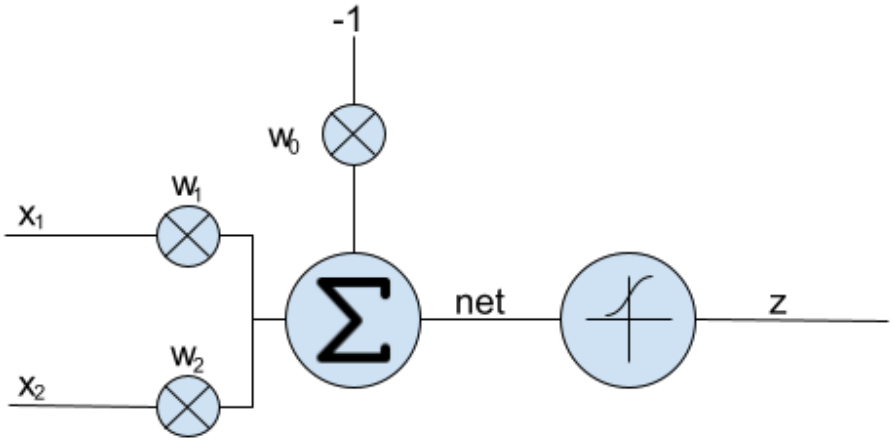
Die Mathematik neuronaler Netze

Eine Einführung

By
Luca Ritz



2021
Luca Ritz

Inhaltsverzeichnis

1 Einführung **2**

2 Die Mathematik neuronaler Netze **3**
2.1 Das Perceptron 3
2.1.1 Lernverfahren 4
2.1.2 Das Problem mit XOR und nichtlinearen Funktionen 6
2.2 Neuronale Netze 6
2.2.1 Lernverfahren mit Gradientenabstieg 7
2.2.2 Lernverfahren mit Backpropagation 7
2.2.3 XOR und die Lösung 7

Abbildungsverzeichnis **8**

Glossar **9**

3 Anhang **10**
3.1 Die Ableitung 1. Grades 10
3.2 Die partielle Ableitung 10
3.3 Die Sigmoide 11
3.4 Der Gradient 13
3.5 Das Gradientenabstiegsverfahren 14

Kapitel 1

Einführung

In Anbetracht der Tatsache, dass neuronale Netzwerke in Zukunft eine grössere Rolle spielen, wird in diesem Dokument das Ziel verfolgt, die Mathematik dahinter verständlich zu erklären. In einem ersten Schritt soll geklärt werden, was ein neuronales Netz überhaupt ist. Dieses besteht bekanntermassen aus mehreren Schichten, diese Schichten wiederum aus mehreren Neuronen. Die Neuronen setzen sich aus einer gewichteten Summe und einer Aktivierungsfunktion zusammen.

Für Leser, welche sich im Gebiet der Differentialrechnung und Optimierung noch nicht so gut auskennen, sei hier an dieser Stelle geraten, den Anhang zu lesen. Sobald dies getan und verstanden ist, kann mit dem Hauptteil begonnen werden.

Grundsätzlich bildet ein neuronales Netz einen gegebenen Input auf einen Output ab, ist also nichts weiteres als eine Funktion. Diese Funktion wiederum ist mehrdimensional, hat also mehrere Input-Variablen und bildet diese wiederum auf einen mehrdimensionalen Output um $f : \mathbb{R}^m \to \mathbb{R}^n$. Die Input-Variablen sind die Gewichte, die Output-Werte die Klassen[1]. Grundsätzlich werden nun die Gewichte dieses neuronalen Netzes so trainiert, dass eine gewählte Fehlerfunktion möglichst minimiert wird. Man befindet sich hier im Bereich der Optimierung. In Abbildung 1.1 ist eine solche lineare Funktion (in rot) gegeben, welche die Datenpunkte möglichst optimal annähert.

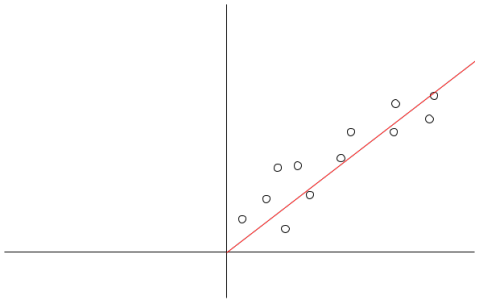


Abbildung 1.1: Annäherung an Datenpunkte über lineare Funktion

[1]Sollen z.B. in einem Bild Hunde und Katzen erkannt werden, so gibt es zwei Klassen „Hund“ und „Katze“. In dem Fall ist die Output-Dimension zweidimensional.

Kapitel 2

Die Mathematik neuronaler Netze

2.1 Das Perceptron

Zu Beginn steht das Perceptron. Es wird hier wie in Abbildung 2.1 veranschaulicht. Es besteht aus der Summe mehrerer gewichteter Eingabewerte wie auch einem Bias, welcher die Schwelle einer Aktivierung verschiebt. Die Gewichte werden mit w_i, die Eingabewerte mit x_i bezeichnet. Diese Summe, im folgenden als net bezeichnet, wird in eine Aktivierungsfunktion, hier eine Sigmoide, gegeben. Der resultierende Wert wird als z bezeichnet. Der Bias hat den fixen Inputwert -1, er wird wiederum über ein Gewicht w_0 trainiert/eingestellt.

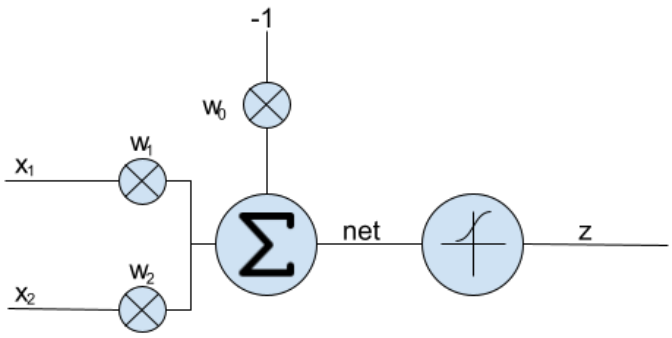


Abbildung 2.1: Das Perceptron

Am Ende dieses Perceptrons steht die Berechnung eines Fehlers. An dieser Stelle wird die quadratische Distanz gewählt. Man berechnet also die quadratische Differenz zwischen dem erwarteten Wert d sowie dem effektiven Wert z.

$$P_{err} = \sum_{i}^{n} (d_i - z_i)^2 \tag{2.1}$$

Im Falle des Perceptrons gibt es nur einen Output-Wert. Daher lässt sich die Fehlerfunktion wie in 2.2 darstellen.

$$P_{err} = (d - z)^2 \tag{2.2}$$

2.1.1 Lernverfahren

Um nun die Gewichte entsprechend ihrem Anteil am Fehler zu korrigieren, wird das Gradientenabstiegsverfahren[1] verwendet. Dazu muss die Ableitung der Fehlerfunktion bekannt sein. Um diese Ableitung nun einfacher zu gestalten, wird die Fehlerfunktion mit einem konstanten Faktor $\frac{1}{2}$ multipliziert. Dieser Faktor hat keinen Einfluss auf Minima oder Maxima, da er an jedem Punkt der Funktion angewendet wird.

$$P_{err} = \frac{1}{2} \cdot (d-z)^2 \tag{2.3}$$

Nun hat eine Maximierung ebenfalls einen vereinfachenden Vorteil aufgrund der Kettenregel bei der Ableitungsbildung, weswegen die Fehlerfunktion gedreht wird. Es ergibt sich die endgültige Fehlerfunktion.

$$P_{err} = -\frac{1}{2} \cdot (d-z)^2 \tag{2.4}$$

Entsprechend lautet die Ableitung unter Anwendung der Kettenregel:

$$\frac{\delta P_{err}}{\delta z} = -\frac{1}{2} \cdot 2 \cdot (d-z) \cdot -1 \tag{2.5}$$

$$\frac{\delta P_{err}}{\delta z} = (d-z) \tag{2.6}$$

Die Gewichte können anhand des Gradienten korrigiert werden.

$$\begin{pmatrix} w_{0,neu} & w_{1,neu} & w_{2,neu} \end{pmatrix} = \begin{pmatrix} w_{0,alt} & w_{1,alt} & w_{2,alt} \end{pmatrix} + \lambda \cdot \vec{\nabla} \tag{2.7}$$

Dieser Gradient setzt sich nun aus der Ableitungskette der verschiedenen Funktionen zusammen, welche nacheinander aufgerufen und jeweils als Input für die nächste dienen. Dazu wird das Perceptron aus Abbildung 2.1 von hinten her aufgerollt. Der Gradient lautet demnach:

$$\vec{\nabla} = \begin{pmatrix} \frac{\delta P_{err}}{\delta w_0} & \frac{\delta P_{err}}{\delta w_1} & \frac{\delta P_{err}}{\delta w_2} \end{pmatrix} \tag{2.8}$$

An dieser Stelle wird nun die Ableitungskette für w_0 näher erläutert. In einem ersten Schritt muss also die Ableitung der Fehlerfunktion nach w_0 gebildet werden. Wie bereits erwähnt, werden die Funktionen nacheinander aufgerufen und dienen sich gegenseitig als Input. Von hinten her aufgerollt lautet die Aufrufreihenfolge:

$$P_{err} = (d - z(net(w_0, w_1, w_2))) \tag{2.9}$$

Für z und net gilt:

$$z = \frac{1}{1+e^{-net}} \tag{2.10}$$

$$net = -1 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 \tag{2.11}$$

[1] Siehe Kapitel 3.5

Um nun den Wert des Gradienten für w_0 zu bilden, muss die Funktion P_{err} für w_0 partiell abgeleitet werden. Dies geschieht durch konsequentes Anwenden der Kettenregel.

$$\frac{\delta P_{err}}{\delta z(w_0)} = (d - z) \tag{2.12}$$

$$\frac{\delta z(w_0)}{\delta net(w_0)} = z(1 - z) \tag{2.13}$$

$$\frac{\delta net(w_0)}{\delta w_0} = -1 \tag{2.14}$$

Die gesamte Ableitungskette nach Anwendung der Kettenregel lautet nun:

$$\frac{\delta P_{err}}{\delta w_0} = \frac{\delta P_{err}}{\delta z(w_0)} \cdot \frac{\delta z(w_0)}{\delta net(w_0)} \cdot \frac{\delta net(w_0)}{\delta w_0} \tag{2.15}$$

$$\frac{\delta P_{err}}{\delta w_0} = (d - z) \cdot z \cdot (1 - z) \cdot -1 \tag{2.16}$$

Mit dem Ausdruck 2.16 lässt sich das Gewicht w_0 in einer Iteration korrigieren. Dasselbe Verfahren wird für die Gewichte w_1 sowie w_2 angewendet, es resultiert:

$$\frac{\delta P_{err}}{\delta w_1} = (d - z) \cdot z \cdot (1 - z) \cdot x_1 \tag{2.17}$$

$$\frac{\delta P_{err}}{\delta w_2} = (d - z) \cdot z \cdot (1 - z) \cdot x_2 \tag{2.18}$$

2.1.2 Das Problem mit XOR und nichtlinearen Funktionen

Mittels eines Perceptrons kann also eine Funktion implementiert werden, die ab einer gewissen Schwelle den Ausgabewert 1 liefert. Geoemtrisch kann dies als eine lineare Separation angesehen werden. Es können z.B. logische Operatoren wie „AND“, „OR“ oder auch „NAND“ abgebildet werden. In der Abbildung 2.2 werden die genannten Operatoren gezeigt. Auf den X-Y-Achsen sind jeweils die Input-Variablen angegeben. Die Output-Dimension wird als Kreis dargestellt. Da der Input auf die Paare $(x, y) \longrightarrow (00), (10), (01), (11)$ beschränkt ist, befinden sich die Output-Werte ebenfalls nur an diesen Positionen. Wird an einer Stelle als Ausgabe eine 1 erwartet, so ist der Kreis rot ausgefüllt. Wird eine 0 erwartet, ist der Kreis nicht ausgefüllt. In

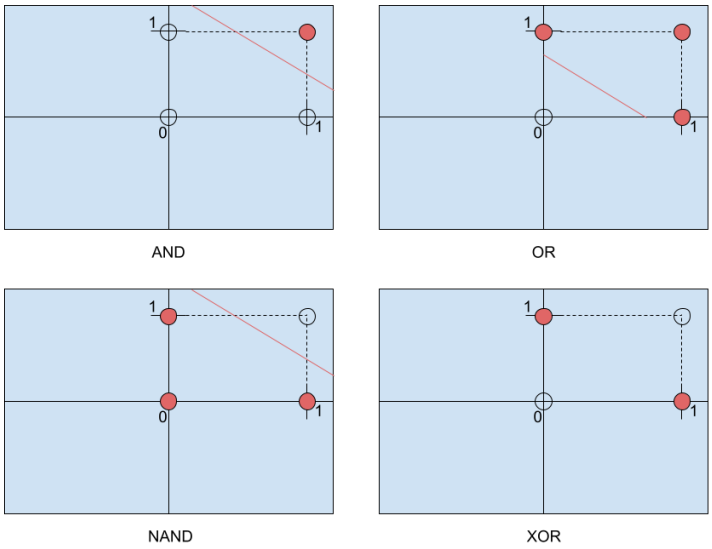


Abbildung 2.2: Logische Operatoren und ihre Separierbarkeit

Abbildung 2.2 wird nun gezeigt, was mit einem Perceptron möglich ist. Es können nur Funktionen implementiert werden, die die einzelnen Klassen voneinander linear separieren können (durch die eingezeichnete rote Gerade). Dies ist im Falle von „XOR“ nicht möglich. Der Leser kann sich selbst überlegen, wie eine solche Gerade auszusehen vermag, um eine Grenze zu ziehen. Diese Problematik wird nun durch ein neuronales Netz gelöst, wo viele dieser Perceptronen miteinander verbunden werden. Dadurch ergeben sich weitere Möglichkeiten, um die Werte in bestimmten Clustern zu klassifizieren. Es können also auch nichtlineare Funktionen abgebildet werden.

2.2 Neuronale Netze

Ein neuronales Netz ist ein Zusammenschluss aus mehreren Perceptronen in verschiedenen Layern. Als Beispiel sei an dieser Stelle die Abbildung 2.3 gegeben. Dieses wird ebenso verwendet, um die Ableitungskette für ein Gewicht in diesem Fall zu zeigen.

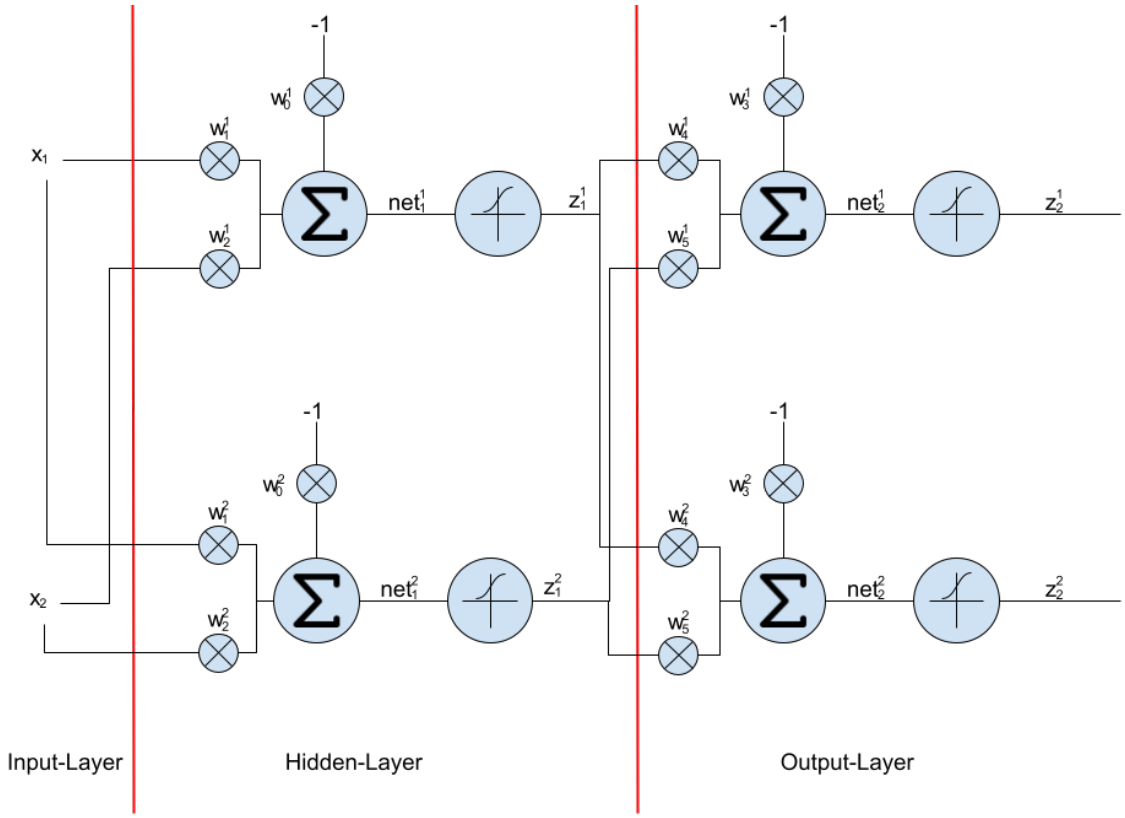


Abbildung 2.3: Ein neuronales Netz mit einem Hidden-Layer

2.2.1 Lernverfahren mit Gradientenabstieg

2.2.2 Lernverfahren mit Backpropagation

2.2.3 XOR und die Lösung

Abbildungsverzeichnis

1.1 Annäherung an Datenpunkte über lineare Funktion 2

2.1 Das Perceptron 3
2.2 Logische Operatoren und ihre Separierbarkeit 6
2.3 Ein neuronales Netz mit einem Hidden-Layer 7

3.1 Steigung an einem bestimmten Punkt der Funktion $f(x)$ 10
3.2 Die Sigmoid-Funktion 11
3.3 Die Ableitung der Sigmoid-Funktion 13
3.4 Gradient an der Position $(1, 1)$ 13

Glossar

Kapitel 3

Anhang

3.1 Die Ableitung 1. Grades

Die Ableitung 1. Grades beschreibt die Steigung an einem bestimmten Punkt der Funktion. In der Abbildung 3.1 wird eine Funktion $f(x)$ (in grün) gegeben. Die Steigung $f'(x)$ an einem bestimmten Punkt x ist rot markiert. Die Ableitung selbst ist wiederum eine Funktion und kann über diverse Ableitungsregeln aufgrund der gegebenen Funktion selbst gebildet werden.

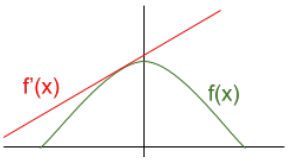


Abbildung 3.1: Steigung an einem bestimmten Punkt der Funktion $f(x)$

Die Ableitungsregeln, welche im Zuge der Erklärung des Lernprozesses eines neuronalen Netzwerks benötigt werden, sind nachfolgend ersichtlich.

Potenzregel $f(x) = x^n \longrightarrow f'(x) = n \cdot x^{n-1}$

Kettenregel $f(x) = u(v(x)) \longrightarrow f'(x) = u'(v(x)) \cdot v'(x)$

Es existieren viele weitere Ableitungsregeln, auf die hier nicht weiter eingegangen wird. Die Schreibweise der Ableitung einer Funktion nach einer Variablen lautet $f'(x) = \frac{\delta f(x)}{\delta x}$.

3.2 Die partielle Ableitung

Ist der Input einer Funktion mehrdimensional, das heisst, die Funktion f ist abhängig von mehreren Variablen, dann kann die Ableitung jeweils lediglich nach einer Variablen gebildet werden. Die übrigen Variablen werden als konstant angesehen. In dem Fall beschreibt die partielle Ableitung die Steigung an einem bestimmten Punkt der abgeleiteten Dimension. Es sei als Beispiel die Funktion $f(x,y) = x^2 + y^2 + 10$ gegeben. Diese wird nun partiell nach x sowie

nach z abgeleitet.

$$f^x(x,y) = 2x \tag{3.1}$$
$$f^y(x,y) = 2y \tag{3.2}$$

Die hierbei angewendete Ableitungsregel ist die Potenzregel, welche bereits im vorangegangenen Kapitel erwähnt wurde.

3.3 Die Sigmoide

Als Aktivierungsfunktion wird die Sigmoidfunktion benutzt. Diese wird heutzutage meist nicht mehr eingesetzt aufgrund des schlechten Lernverhaltens in einigen Bereichen der Funktion. Die erwähnte Eigenschaft wird bei der Betrachtung der Ableitung ersichtlich.

$$sig(t) = \frac{1}{1+e^{-t}} \tag{3.3}$$

Geometrisch lässt sich die Funktion wie in Abbildung 3.2[1] so interpretieren, dass eine gewisse Schwelle existiert, ab der die Funktion den Eingabewert auf 1 abbildet, in dem Jargon der Neuronen also „feuert“. Das Resultat lautet entweder 0 oder 1.

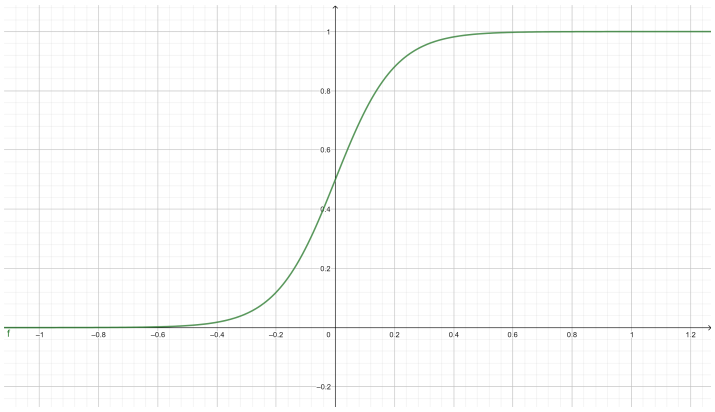


Abbildung 3.2: Die Sigmoid-Funktion

Für die weitere Verwendung ist nun vor allem die Ableitung der Sigmoide interessant, welche in den nachfolgenden Zeilen behandelt wird. Zuerst wird der Bruch durch eine andere Schreibweise (Exponent -1) dargestellt. Weiterhin lässt sich nun die Sigmoide als Verkettung von zwei Funktionen f und y schreiben.

$$sig(t) = (1+e^{-t})^{-1} \longrightarrow y(t) = 1+e^{-t}, f(t) = y(t)^{-1} \tag{3.4}$$

[1] Die Sigmoid-Funktion kann über einen Parameter modifiziert werden. Die eigentliche Funktion lautet $sig(t) = \frac{1}{1+e^{-\alpha \cdot t}}$. Für die Abbildungen der Sigmoide wie deren Ableitung wurde ein Parameter $\alpha = 10$ gewählt. Für die Herleitung der Ableitungskette der neuronalen Netze wie auch bei der weiteren Erklärung in diesem Kapitel wird der Wert auf 1 belassen. Für die Abbildung wurde ein höherer Wert gewählt, um den Effekt der Ableitung nahe 0 zu verdeutlichen.

Es handelt sich also um eine äussere und innere Funktion, wobei nun die Ableitungsregel 3.1 angewendet wird. Für die innere Ableitung wird nun noch die Regel $f(x) = e^x \longrightarrow f'(x) = e^x$ verwendet.

$$\frac{\delta f(t)}{\delta t} = (-1) \cdot (y(t))^{-2} \tag{3.5}$$

$$\frac{\delta y(t)}{\delta t} = (e^{-t}) \cdot (-1) \tag{3.6}$$

Nach der Kettenregel resultiert:

$$\frac{\delta sig(t)}{\delta t} = (-1) \cdot (1 + e^{-t})^{-2} \cdot (e^{-}t) \cdot (-1) \tag{3.7}$$

$$\frac{\delta sig(t)}{\delta t} = \frac{e^{-t}}{(1 + e^{-t})^2} \tag{3.8}$$

Nun wird $\frac{1}{1+e^{-t}}$ ausgeklammert.

$$\frac{\delta sig(t)}{\delta t} = \frac{1}{1 + e^{-t}} \cdot \frac{e^{-t}}{1 + e^{-t}} \tag{3.9}$$

Beim Zähler wird 1 dazuaddiert und abgezogen, damit der Faktor umgeformt werden kann.

$$\frac{\delta sig(t)}{\delta t} = \frac{1}{1 + e^{-t}} \cdot \frac{e^{-t} + 1 - 1}{1 + e^{-t}} \tag{3.10}$$

$$\frac{\delta sig(t)}{\delta t} = \frac{1}{1 + e^{-t}} \cdot (\frac{1 + e^{-t}}{1 + e^{-t}} - \frac{1}{1 + e^{-t}}) \tag{3.11}$$

Es resultiert die Ableitung in bekannter Form 3.12

$$\frac{\delta sig(t)}{\delta t} = sig(t) \cdot (1 - sig(t)) \tag{3.12}$$

Auch hier kann eine geometrische Interpretation in Abbildung 3.3 erfolgen. Ersichtlich wird nun, dass die Steigung nur in einem sehr kleinen Intervall stark ungleich 0 ist. Dies bedeutet, dass sich im Falle eines solchen Variablenwerts, wo die Steigung fast 0 ist, kaum eine Korrektur durch den Gradienten[2] durchzuführen ist. Daher werden heutzutage Aktivierungsfunktionen wie die „ReLU“ oder „LeakyReLU“ verwendet, welche dieses Problem beheben.

[2] Siehe Kapitel 3.4

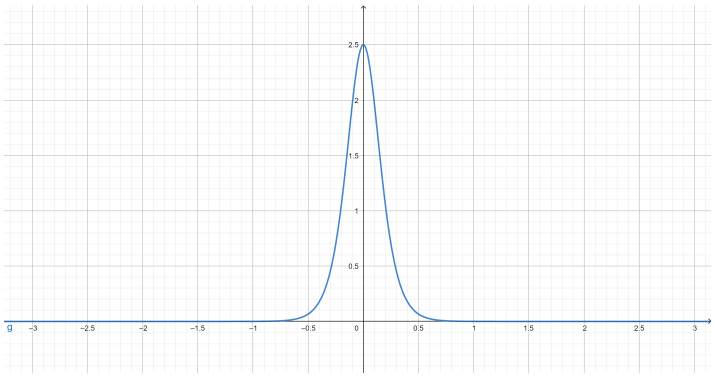


Abbildung 3.3: Die Ableitung der Sigmoid-Funktion

3.4 Der Gradient

Der Gradient beschreibt einen Vektor, welcher in Richtung des steilsten Anstiegs einer Funktion zeigt. Die Komponenten des Gradientenvektors bestehen aus den partiellen Ableitungen der Funktion an der jeweiligen Variablen.

$$\nabla f(x,y) \longrightarrow \left(\frac{\delta f(x,y)}{\delta x} \quad \frac{\delta f(x,y)}{\delta y}\right) \tag{3.13}$$

Geometrisch kann dies an der Position $(x = 1, y = 1)$ für die Funktion $f(x,y) = x^2 + y^2$ wie in Abbildung 3.4 aussehen. Zu beachten sei hier, dass der eigentliche Gradient in der XY-Ebene liegt (blau). Der schwarze Vektor soll lediglich anzeigen, was eine Verschiebung in dieser Richtung bei der Eingabe der Variablen für den Ausgabewert der Funktion bedeutet. Die Länge des Gradienten entspricht der Stärke der Steigung an diesem Punkt.

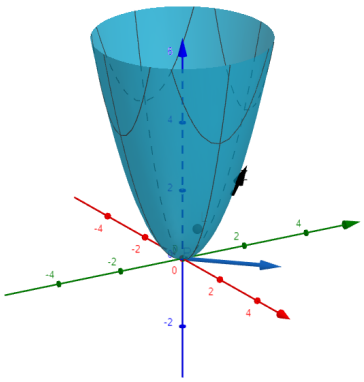


Abbildung 3.4: Gradient an der Position $(1,1)$

3.5 Das Gradientenabstiegsverfahren

Hierbei handelt es sich um ein Optimierungsverfahren, um einen Maximal- oder Minimalwert[3] einer gegebenen Zielfunktion zu finden. Es wird in dem Fall der Gradient, wie in Kapitel 3.4 besprochen, verwendet. Die Idee ist, dass man bei einer Maximierung in kleinen Schritten in Richtung des Gradienten folgt. Als Beispiel wird eine Funktion angegeben, die von zwei Variablen x, y abhängig ist. Beim Wert λ handelt es sich um die Lernrate, welche die Länge der zu gehenden Schritte beeinflusst. Der $\vec{\nabla}$ steht hierbei für den Gradienten.

$$\begin{pmatrix} x_{neu} & y_{neu} \end{pmatrix} = \begin{pmatrix} x_{alt} & y_{alt} \end{pmatrix} + \lambda \cdot \vec{\nabla} \tag{3.14}$$

Geometrisch kann wiederum die Abbildung 3.4 hinzugezogen werden. Dort würde man in einem ersten Schritt in Richtung des schwarzen Vektors gehen, respektive in Richtung des Blauen, wenn man nur die Variablen beachtet.

[3] Obwohl das Verfahren Gradientenabstieg heisst, kann ebenso ein Gradientenaufstieg durchgeführt werden. Dies ist auch der Ansatz, der in diesem Kapitel erwähnt wird. Beim Gradientenabstieg wird lediglich der Gradientenvektor in die umgekehrte Richtung verfolgt, man sucht also ein Minimum. Die Formel lautet in dem Fall $\vec{w_{neu}} = \vec{w_{alt}} - \lambda \cdot \vec{\nabla}$, wobei der Vektor $\vec{w}$ für die anzupassenden Variablen steht.