

# Wie aktualisieren wir eigentlich die Gewichte?

Bislang haben wir die sehr zentrale Frage, wie die Verknüpfungsgewichte in einem neuronalen Netz aktualisiert werden, noch nicht in Angriff genommen. Wir haben daran gearbeitet, bis zu diesem Punkt zu kommen, und wir sind fast da. Allerdings müssen wir noch einen Kerngedanken verstehen, bevor wir dieses Geheimnis lüften können.

Die Fehler haben wir bis jetzt zu jeder Schicht des Netzes zurückgeführt. Warum? Weil wir uns am Fehler orientieren, wenn es darum geht, die Verknüpfungsgewichte anzupassen, um die vom Netz gegebene Gesamtantwort zu verbessern. Grundsätzlich tun wir damit das Gleiche wie beim linearen Klassifizierer zu Beginn dieses Leitfadens.

Doch diese Knoten sind keine einfachen linearen Klassifizierer. Diese etwas komplexeren Knoten summieren die gewichteten Signale, die in den Knoten eingehen, und wenden die sigmoidale Schwellwertfunktion an. Wie also aktualisieren wir die Gewichte für Verknüpfungen, die diese komplexeren Knoten verbinden?

Warum können wir nicht mit irgendeiner raffinierten Algebra direkt herausfinden, wie groß die Gewichte sein sollten?

Wir können mit einer raffinierten Algebra die Gewichte nicht direkt ermitteln, weil die mathematischen Operationen zu schwierig sind. Es sind einfach zu viele Kombinationen von Gewichten und zu viele Funktionen von Funktionen von Funktionen ... zu kombinieren, wenn wir das Signal in Vorwärtsrichtung durch das Netz leiten. Stellen Sie sich nur einmal ein kleines neuronales Netz mit drei Schichten und drei Neuronen in jeder Schicht vor, wie wir es weiter oben verwendet haben. Wie würden Sie ein Gewicht für eine Verknüpfung zwischen dem ersten Eingabeknoten und dem zweiten versteckten Knoten anpassen, sodass der dritte Ausgabeknoten seine Ausgabe um beispielsweise 0,5 erhöht? Selbst wenn wir erfolgreich wären, könnte der Effekt zunichtegemacht werden, indem ein anderes Gewicht angepasst wird, um einen anderen Ausgabeknoten zu verbessern. Offenbar ist das Ganze überhaupt nicht trivial.

Um ein Bild davon zu bekommen, wie untrivial das ist, sollten Sie sich den furchtbaren Ausdruck in Abbildung 1-76 ansehen, der die Ausgabe eines Ausgabeknotens als Funktion der Eingänge und der Verknüpfungsgewichte für ein einfaches dreischichtiges neuronales Netz mit drei Knoten in jeder Schicht zeigt. Der Eingang bei Knoten  $i$  ist  $x_i$ , die Gewichte für die Verknüpfungen, die den Eingabeknoten  $i$  mit dem versteckten Knoten  $j$  verbinden, sind  $w_{i,j}$ . Analog wird die Ausgabe des versteckten Knotens  $j$  mit  $x_j$  bezeichnet, und die Gewichte für die Verknüpfungen, die den versteckten Knoten  $j$  mit dem Ausgabeknoten  $k$  verbinden, heißen  $w_{j,k}$ . Das Symbol  $\Sigma_a^b$  bedeutet, dass der danach stehende Ausdruck für alle Werte zwischen  $a$  und  $b$  summiert wird.

$$o_k = \frac{1}{1 + e^{-\sum_{j=1}^3 (w_{j,k} \cdot \frac{1}{1 + e^{-\sum_{i=1}^3 (w_{i,j} \cdot x_i)}})}}$$

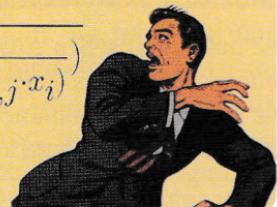


Abbildung 1-76: Ausdruck, der die Ausgabe eines Ausgabeknotens als Funktion der Eingänge und Verknüpfungsgewichte für ein dreischichtiges neuronales Netz mit drei Knoten in jeder Schicht beschreibt

Entsetzlich! Das wollen wir lieber nicht entwirren.

Anstatt nun überschlau sein zu wollen, könnten wir nicht einfach zufällige Kombinationen von Gewichten ausprobieren, bis wir eine gute Kombination gefunden haben?

Diese Idee ist gar nicht so verrückt, vor allem wenn man in einem schwierigen Problem feststeckt. Das Konzept, mit dessen Hilfe man zum Beispiel sogar Passwörter zu knacken versucht, ist auch als *Brute-Force-Methode* bekannt. Das kann durchaus funktionieren, wenn das Passwort ein kurzes Wort oder ein Name ist, weil dann ein schneller Heimcomputer in kurzer Zeit alle Möglichkeiten durchprobieren kann. Stellen Sie sich nun vor, dass jedes Gewicht 1.000 Möglichkeiten zwischen –1 und +1 haben könnte, wie zum Beispiel 0,501, –0,203 und 0,999. Dann gibt es bei einem dreischichtigen neuronalen Netz mit drei Knoten in jeder Schicht 18 Gewichte, sodass 18.000 Möglichkeiten zu testen wären. Wenn wir ein eher typisches neuronales Netz mit 500 Knoten in jeder Schicht betrachten, müssten 500 Millionen Möglichkeiten für die Gewichte getestet werden. Nimmt man für jeden Satz von Kombinationen eine Rechendauer von einer Sekunde an, würde es 16 Jahre dauern, die Gewichte nach nur einem Trainingsbeispiel zu aktualisieren! Und bei 1.000 Trainingsbeispielen wären wir bei 16.000 Jahren!

Die Brute-Force-Methode ist offenbar in der Praxis überhaupt nicht anwendbar. Die Lage wird schnell noch schlimmer, wenn weitere Schichten, Knoten oder mögliche Gewichtswerte hinzukommen.

An diesem Rätsel bissen sich Mathematiker jahrelang die Zähne aus, es wurde erst in den 1960er- bis 1970er-Jahren praktisch gelöst. Es gibt verschiedene Ansichten darüber, wem dies zuerst gelang oder wer den entscheidenden Durchbruch erzielte, doch wichtig ist vor allem, dass diese späte Entdeckung zu einer Explosion moderner neuronaler Netzen führte, die manche sehr eindrucksvolle Aufgaben durchführen können.

Wie lösen wir nun ein derartiges anscheinend schweres Problem? Ob Sie es glauben oder nicht, Sie besitzen bereits die Werkzeuge, um es in eigener Regie zu bewerkstelligen. Wir haben alles dazu Notwendige bereits weiter oben behandelt. Machen wir also weiter!

Zunächst müssen wir aber *Pessimismus* akzeptieren.

Die mathematischen Ausdrücke, die zeigen, wie alle Gewichte zur Ausgabe eines neuronalen Netzes führen, sind zu komplex, um sie leicht entwirren zu können. Die Gewichtskombinationen sind zu zahlreich, um eine nach der anderen testen und die beste finden zu können.

Es gibt sogar noch mehr Gründe, um pessimistisch zu sein. Die Trainingsdaten reichen eventuell nicht aus, um ein Netz ordnungsgemäß anzulernen. Sie enthalten vielleicht sogar Fehler, sodass wir nicht mehr davon ausgehen können, dass sie perfekt wahr sind, um von ihnen lernen zu können. Das Netz selbst enthält möglicherweise nicht genügend Schichten oder Knoten, um die richtige Lösung für das Problem zu modellieren.

Das bedeutet: Wir müssen einen realistischen Ansatz wählen, der diese Beschränkungen erkennt. Wenn wir das tun, könnten wir ein Konzept finden, das zwar

mathematisch nicht perfekt ist, tatsächlich aber bessere Ergebnisse liefert, weil es keine falschen, idealisierten Annahmen trifft.

Wir wollen veranschaulichen, was damit gemeint ist. Stellen Sie sich eine sehr komplizierte Landschaft vor mit Bergspitzen und Tälern sowie Bergen mit tückischen Unebenheiten und Spalten. Es ist finster, und Sie können nichts sehen. Sie wissen, dass Sie sich auf einer Anhöhe befinden und ganz nach unten gelangen müssen. Von der gesamten Landschaft besitzen Sie keine genaue Karte. Allerdings haben Sie eine Taschenlampe. Was tun Sie jetzt? Wahrscheinlich werden Sie im Schein der Taschenlampe den Boden in der nahen Umgebung inspizieren. Weiter entferntes Gelände ist überhaupt nicht zu sehen und gleich gar nicht die gesamte Landschaft. Sie können erkennen, welcher Teil des Bodens anscheinend nach unten führt, und kleine Schritte in dieser Richtung gehen. Auf diese Weise tasten Sie sich langsam den Berg hinunter, immer Schritt für Schritt, ohne eine vollständige Karte zu besitzen und ohne im Voraus eine Route geplant zu haben.



Abbildung 1-77: Beispiel für ein bergiges Gelände, in dem Sie sich fast blind bewegen müssen

Die mathematische Version dieses Konzepts heißt *Gradientenverfahren* oder auch *Verfahren des steilsten Abstiegs* – es dürfte klar sein, warum. Nachdem Sie einen Schritt gegangen sind, untersuchen Sie wieder die nahe Umgebung, um festzustellen, in welcher Richtung Sie Ihrem Ziel näher kommen. Dann wagen Sie erneut einen Schritt in diese Richtung. Das setzen Sie so lange fort, bis Sie glücklich am Fuß der Berge angekommen sind. Der Gradient entspricht der Bodenneigung. Sie gehen in die Richtung, wo die Neigung am steilsten nach unten führt.

Stellen Sie sich nun diese komplexe Landschaft als mathematische Funktion vor. Das Gradientenverfahren erlaubt es uns, das Minimum zu finden, ohne diese komplexe Funktion so weit zu verstehen, um sie mathematisch beschreiben zu können. Wenn eine Funktion so schwierig ist, dass sich das Minimum nicht ohne Weiteres mithilfe der Algebra finden lässt, können wir stattdessen diese Methode verwenden. Zweifellos dürfen wir keine genaue Antwort erwarten, weil wir uns der Antwort schrittweise nähern und unsere Position dabei Stück für Stück verbessern.

sern. Doch das ist besser, als überhaupt keine Antwort zu bekommen. Immerhin können wir die Antwort mit immer kleineren Schritten in Richtung tatsächliches Minimum immer weiter verfeinern, bis wir mit der erreichten Genauigkeit zufrieden sind.

Wie hängt nun dieses wirklich coole Gradientenverfahren mit neuronalen Netzen zusammen? Wenn die komplexe schwierige Funktion den Fehler des Netzes darstellt, bedeutet ein Hinabsteigen, um das Minimum zu finden, dass wir den Fehler minimieren. Wir verbessern die Ausgabe des Netzes. Das ist genau das, was wir wollen!

Die Idee des Gradientenabstiegs wollen wir an einem äußerst einfachen Beispiel veranschaulichen, um sie richtig verstehen zu können.

Der Graph in Abbildung 1-78 zeigt eine einfache Funktion  $y = (x - 1)^2 + 1$ . Wäre es eine Funktion, bei der  $y$  den Fehler bedeutet, würden wir das  $x$  suchen, das den Fehler minimiert. Für einen Moment wollen wir so tun, als wäre dies keine einfache Funktion, sondern eine komplizierte, schwierige Funktion.

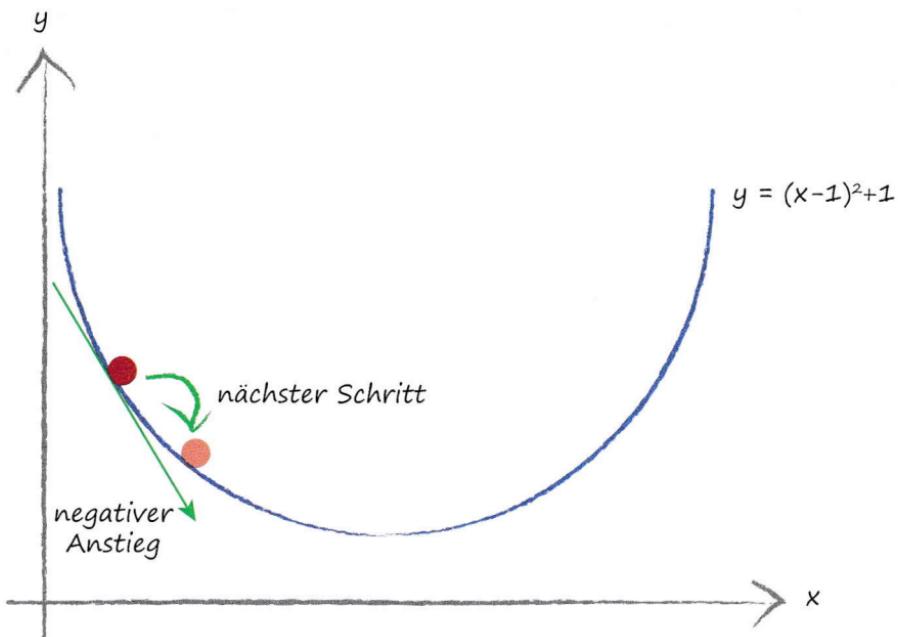


Abbildung 1-78: Beispiel einer Funktion, um das Gradientenverfahren zu veranschaulichen

Um das Gradientenverfahren durchzuführen, müssen wir irgendwo beginnen. Der Graph zeigt unseren zufällig gewählten Ausgangspunkt. Wie der Bergsteiger sehen wir uns die Umgebung unserer aktuellen Position an und stellen fest, in welche Richtung es nach unten geht. Der Anstieg ist auf dem Graphen markiert und in diesem Fall ein negativer Gradient. Wir folgen der Abwärtsrichtung, sodass wir

uns entlang der x-Achse nach rechts bewegen. Das heißt, wir erhöhen x ein wenig. Das ist der erste Schritt unseres Bergsteigers. Es ist zu erkennen, dass wir unsere Position verbessert haben und näher an das tatsächliche Minimum herangekommen sind.

Nehmen wir jetzt an, dass wir an einer anderen Stelle gestartet sind, wie es der Graph in Abbildung 1-79 zeigt.

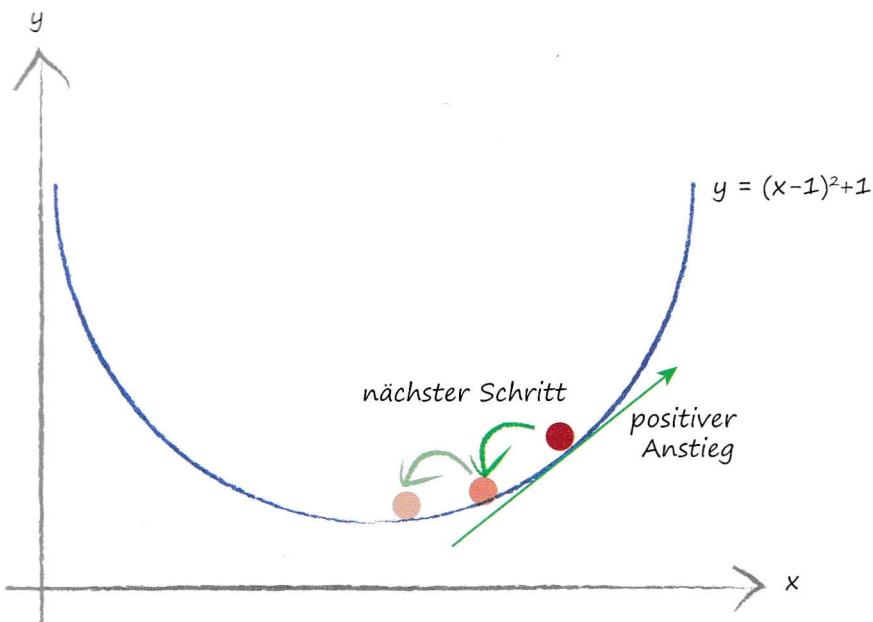


Abbildung 1-79: Das Gradientenverfahren mit einem anderen zufällig gewählten Ausgangspunkt

Dieses Mal ist der Anstieg unter unseren Füßen positiv, sodass wir nach links gehen. Das heißt, wir verringern x ein wenig. Auch hier ist zu erkennen, dass wir unsere Position etwas verbessert haben und dem tatsächlichen Minimum näher gekommen sind. Wir können auf diese Weise weitermachen, bis die Verbesserungen nur noch so klein sind, dass wir zufrieden feststellen können, beim Minimum angekommen zu sein.

Als notwendige Verfeinerung ändern wir die Größe der ausgeführten Schritte, um zu vermeiden, über das Minimum hinauszuschießen und ewig um das Minimum herumzupendeln. Wenn man nur Zweimeterschritte machen kann, aber bereits 0,5 Meter entfernt vom wahren Minimum sind, ist leicht einzusehen, dass uns jeder Schritt in Richtung Minimum über das Minimum hinaus bringt. Passen wir nun die Schrittgröße so an, dass sie proportional zur Größe des Gradienten ist, führen wir kleinere Schritte aus, wenn wir uns dem Minimum nähern. Das setzt aber voraus, dass der Anstieg bei der Annäherung an das Minimum tatsächlich flach

cher wird. Für die meisten sanften stetigen Funktionen ist diese Annahme durchaus zutreffend. Für wilde Zickzackfunktionen mit Sprüngen und Lücken, die der Mathematiker als *Unstetigkeiten* bezeichnet, ist die Annahme allerdings nicht brauchbar.

Abbildung 1-80 veranschaulicht dieses Konzept der Schrittgrößenanpassung bei kleiner werdendem Funktionsgradienten, der ein guter Indikator dafür ist, wie nahe wir einem Minimum sind.

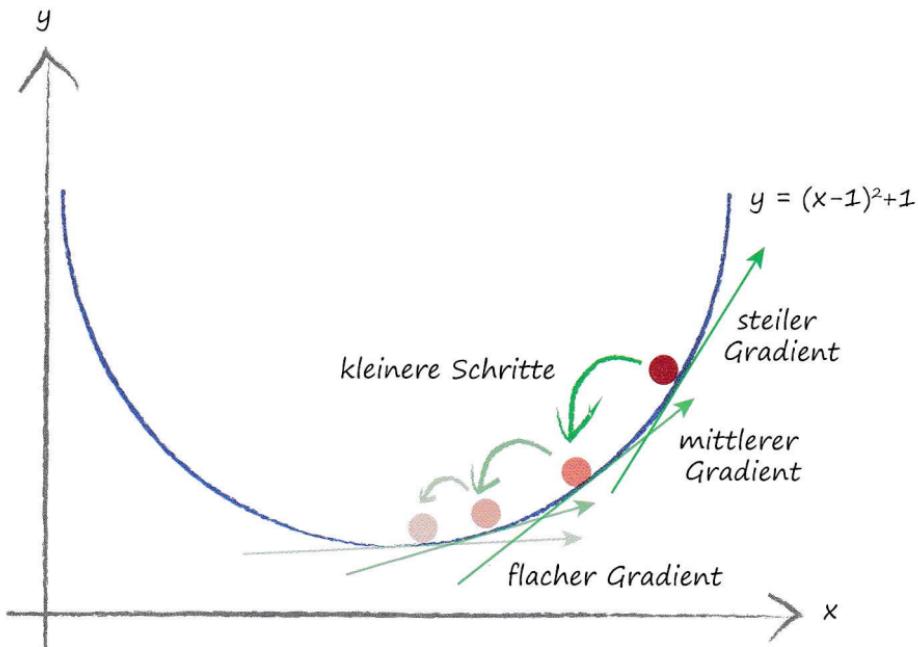


Abbildung 1-80: Gradientenverfahren mit kleiner werdenden Schritten

Haben Sie übrigens bemerkt, dass wir  $x$  in der Gegenrichtung zum Gradienten erhöht haben? Ein positiver Gradient bedeutet, wir verringern  $x$ . Ein negativer Gradient bedeutet, wir erhöhen  $x$ . Die Graphen zeigen das zwar deutlich, doch man vergisst es leicht und schlägt die falsche Richtung ein.

Wir haben bei diesem Gradientenabstieg das wahre Minimum nicht mittels Algebra ermittelt, weil wir so getan haben, als wäre die Funktion  $y = (x - 1)^2 + 1$  zu komplex und zu schwierig. Selbst wenn wir nicht in der Lage sind, den Anstieg mit mathematischer Genauigkeit zu bestimmen, können wir ihn schätzen. Und wie Sie sehen, funktioniert das trotzdem ziemlich gut, um uns in die generell richtige Richtung zu bringen.

Diese Methode überzeugt vor allem bei Funktionen mit vielen Parametern. Das heißt,  $y$  hängt nicht nur von  $x$  ab, sondern vielleicht auch von  $a$ ,  $b$ ,  $c$ ,  $d$ ,  $e$  und  $f$ .

Wie bereits erwähnt, hängen die Ausgabefunktion und demzufolge die Fehlerfunktion eines neuronalen Netzes von vielen, vielen weiteren Gewichtsparametern ab. Oftmals von Hunderten davon!

Abbildung 1-81 veranschaulicht ebenfalls das Gradientenverfahren, doch mit einer etwas komplexeren Funktion, die von zwei Parametern abhängt. Dies lässt sich in drei Dimensionen darstellen, wobei die Höhe den Wert der Funktion angibt.

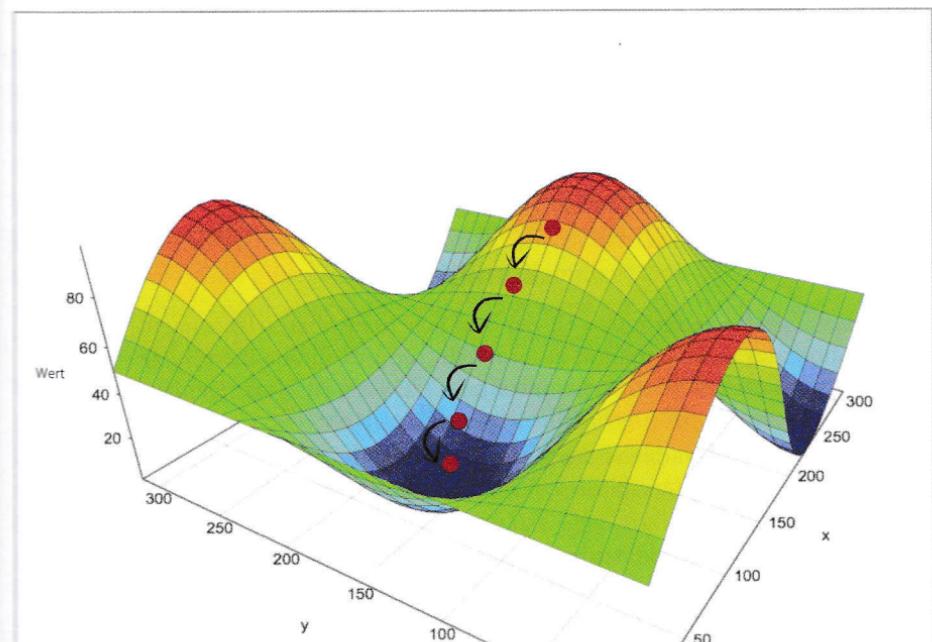


Abbildung 1-81: Gradientenverfahren bei einer komplexeren Funktion

Angesichts dieser dreidimensionalen Oberfläche fragen Sie sich möglicherweise, ob der Gradientenabstieg auch in dem anderen Tal auf der rechten Seite enden kann. Allgemeiner gedacht, stellt sich die Frage: Bleibt das Gradientenverfahren manchmal im falschen Tal stecken, weil bestimmte komplexe Funktionen viele Täler haben? Welches ist das falsche Tal? Es ist ein Tal, das nicht am tiefsten liegt. Die Antwort lautet: Ja, das kann passieren.

Um zu vermeiden, im falschen Tal – oder Funktionsminimum – zu landen, trainieren wir neuronale Netze mehrere Male, wobei jeweils verschiedene Ausgangspunkte auf dem Berg sicherstellen, dass wir nicht immer wieder im falschen Tal landen. Verschiedene Ausgangspunkte heißt, verschiedene Startparameter auszuwählen. Und bei neuronalen Netzen bedeutet das, mit verschiedenen Verknüpfungsgewichten zu beginnen.

Abbildung 1-82 veranschaulicht das Gradientenverfahren mit drei verschiedenen Ausgangspunkten, wobei ein Abstieg im falschen Tal gefangen wird.



Abbildung 1-82: Verschiedene Ausgangspunkte beim Gradientenverfahren

Legen wir eine Pause ein und sammeln wir unsere Gedanken.

### Kernideen

- Das *Gradientenverfahren* ist ein wirklich brauchbares Instrument, um das Minimum einer Funktion zu ermitteln, und es funktioniert auch sehr gut, wenn diese Funktion so komplex und schwierig ist, dass sich das Minimum mathematisch per Algebra nur schwer finden lässt.
- Darüber hinaus arbeitet die Methode auch dann, wenn viele Parameter in das Ergebnis einfließen. In diesem Punkt scheitern andere Methoden oder sind unbrauchbar.
- Diese Methode ist außerdem robust gegenüber unvollkommenen Daten. Wir werden nicht gänzlich fehlgeleitet, wenn die Funktion nicht vollkommen perfekt beschrieben ist oder wir aus Versehen gelegentlich einen falschen Schritt unternehmen.

Die Ausgabe eines neuronalen Netzes ist eine komplexe und schwierige Funktion, deren Rückgabewert von vielen Parametern, den Verknüpfungsgewichten, beeinflusst wird. Können wir überhaupt mit dem Gradientenverfahren die richtigen Gewichte ermitteln? Ja, sofern wir uns für die richtige Fehlerfunktion entscheiden.

Die Ausgabefunktion eines neuronalen Netzes ist selbst keine Fehlerfunktion. Doch wir wissen, dass wir sie leicht in eine Fehlerfunktion umwandeln können, weil sich der Fehler aus der Differenz zwischen den Solltrainingswerten und den tatsächlichen Ausgabewerten berechnet.

Hierbei ist etwas zu beachten. Sehen Sie sich Tabelle 1-6 mit Trainingswerten (Sollwerten) und tatsächlichen Werten (Istwerten) für drei Ausgabeknoten und den Kandidaten für eine Fehlerfunktion an.