Einführung in Neuronale Netze

Thomas Ruland

Contents

Abstract

Im Alltag begegnet jedem Menschen eine riesige Informationsflut. Allein über unsere 5 Sinne werden jede Sekunde Datenmengen in Größenordnungen von mehreren GigaBit registriert aus denen die für uns relevanten Informationen gefiltert werden müssen. Wichtige Entscheidungen werden in Sekundenbruchteilen gefällt. Diese technisch unerreichten Leistungen erbringt ein hochgradig verschaltetes neuronales Netz - unser Gehirn. Bereits in den 50ern des letzten Jahrhunderts gab es erste wissenschaftliche Veröffentlichungen, die ein künstliches neuronales Netz konstruierten. Speziell mit der heutigen Computertechnik sind große Netze realisierbar, die über Beispiele richtige Lösungen trainieren, um sich in komplexen Situationen im Sinne der Aufgabenstellung richtig zu verhalten.

10 May 2004

1 Das menschliche Gehirn - Höchstleistungen im täglichen Leben

Viele alltägliche Handlungen laufen unterbewusst ab. Unser Gehirn scheint im Hintergrund bekannte Problemstellungen automatisch zu bearbeiten. Sogar komplexe motorische Abläufe wie der aufrechte Gang werden mit geringen Reaktionszeiten gesteuert ohne dabei aktiv zu denken. Die Wiedererkennung von Personen gelingt auch unter Einfluss kleinerer "Störungen" wie z.B. einer veränderten Haarfarbe. Aus der Sicht der Computertechnik hochgradig komplexe Leistungen, die für uns jedoch selbstverständlich sind.

2 Die Hardware

2.1 Das Neuron

Betrachtet man das menschliche Gehirn so findet man dort eine sehr große Anzahl von Neuronen, welche die Grundlage der Abläufe bilden.

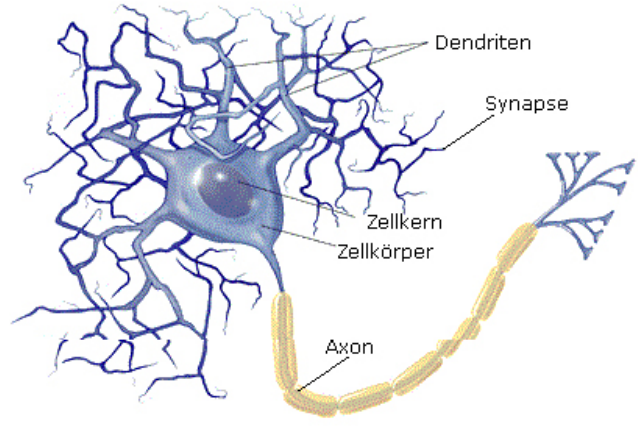

Abbildung 1. Das biologische Neuron, Grafische Darstellung mit Dendriten, Zellkern, Axon und Synapsen

Ein Neuron besteht aus einem **Zellkörper**, mehreren Fasern, genannt **Dendriten**, und einem langen Fortsatz namens **Axon**. Dendriten und Axon spielen dabei die wichtige Rolle der Kommunikation zwischen Neuronen. Fasern unterschiedlicher Zellen interagieren über die so genannten **Synapsen** an ihren Enden. Von den Synapsen abgegebene Botenstoffe lösen komplexe chemische Reaktionen aus, die das elektrische Potential der angeschlossenen Neuronen verändern. Erreicht es einen bestimmten Wert senden diese Zellen ihrerseits

ein elektrisches Signal in Form von Botenstoffen über das Axon aus. Wichtig ist hierbei dass die Signalübetragung in den Synapsen nicht überall identisch abläuft. Hemmende bzw. erregende Synapsen übertragen Impulse unterschiedlich stark.

2.2 Nachahmung in der Computertechnik: Das künstliche Neuron

Bereits 1943 wurde von W. S. McCulloch, und W. H. Pitts ein stark vereinfachtes künstliches Neuron beschrieben. Daraus entwickelte 1957/58 Rosenblatt das Perzeptron zur einfachen Mustererkennung von Ziffern.

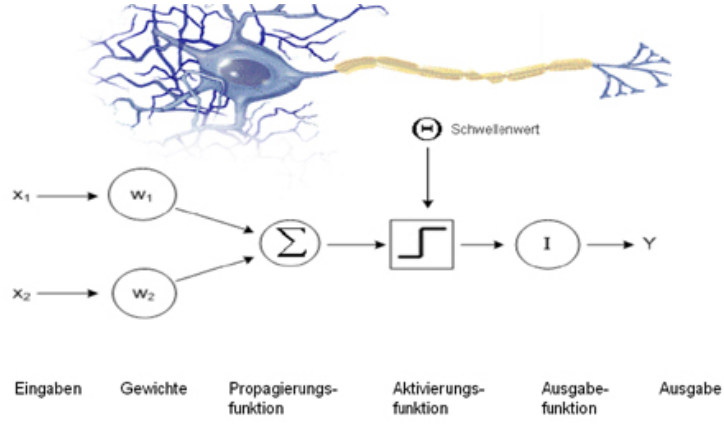

Abbildung 2. Vergleich von biologischem und künstlichen Neuron. Schematische Darstellung des künstlichen Neurons mit den beiden Eingängen x_1 und x_2, ihren spezifischen Gewichten w_1 und w_2 sowie der Ausgabe Y.

Angelehnt an das biologische Neuron besitzt es folgende Struktur:
Eine Propagierungsfunktion verknüpft die mit spezifischen Faktoren gewichteten Eingaben zur so genannten **Netzeingabe**. Die Aktivierungsfunktion bestimmt aus der Netzeingabe unter Berücksichtigung eines Schwellenwertes den internen Zustand des Neurons. Die Ausgabefunktion berechnet dementsprechend den tatsächlichen Ausgabewert. Zusätzlich wird lokal Speicher benötigt um Aktivierungszustand und Schwellenwert zu spezifizieren. Wichtig ist hierbei dass die enthaltenen Zusammenhänge möglichst einfach gehalten werden. Wie dennoch komplexe Abläufe realisierbar sind wird später erläutert.

Die genannten Funktionen werden in der Praxis aus einer kleinen Menge gewählt:

Die **Propagierungsfunktion** stellt meist einfach die Summe aller gewichteten Eingänge als Netzeingabe bereit. Andere theoretische Zusammenhänge sind

möglich, finden jedoch kaum Verwendung.

$$f_{net}(x) = \sum_{i=1}^{n} x_i w_i$$

Die **Aktivierung** bestimmen fast immer um die Stelle $x = 0$ sigmoide Funktionen [1] die für bestimmte Anwendungen konstruiert wurden. Dem biologischen Neuron entsprechend, das von einem charakteristischen Schwellenwert abhängig selbst ein Signal über das Axon ausgibt oder nicht, verhält sich die Schwellwert- oder Stufenfunktion. Übersteigt die Netzeingabe diesen Wert wechselt das Neuron in den Zustand "aktiv". Ein Spezialfall über den Wertebereich $[-1, 1]$ stellt die Signum-Funktion dar. Beide werden als "hard limiter" bezeichnet und in Netzen zur Klassifizierung und Mustererkennung eingesetzt.

$$f_{Schwellwert}(x) = \begin{cases} 1 & : \quad x \geq \Theta \\ 0 & : \quad sonst \end{cases}$$

In der Praxis wird das Zusammenspiel von Propagierungs- und Aktivierungsfunktion oft vereinfacht indem der Schwellenwert Θ bereits beim Zusammenfassen der Eingaben subtrahiert wird und die Schwellwertfunktion ihre Sprungstelle auf $x = 0$ festlegt:

$$f_{net}(x) = \sum_{i=1}^{n} x_i w_i - \Theta$$

$$f_{Schwellwert}(x) = \begin{cases} 1 & : \quad x \geq 0 \\ 0 & : \quad sonst \end{cases}$$

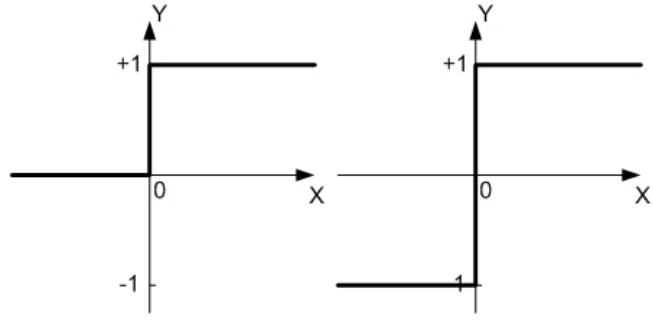

Abbildung 3. Grafische Darstellung der Schwellwertfunktion,

Abbildung 4. Signum-Funktion

[1] Funktion besitzt eine S-Form mit zur x-Achse parallelen Tangenten für $x \longrightarrow \pm\infty$

Eine weitere zur Bestimmung der Aktivierung des Neurons verwendete Funktion ist die lineare. Sie gibt am Ausgang direkt die Netzeingabe aus und wird oft für lineare Funktionsapproximationen verwendet.

$$f_{linear}(x) = id_x$$

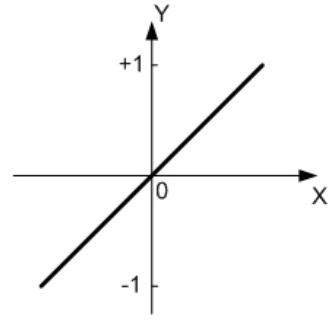

Abbildung 5. Lineare Aktivierungsfunktion

Für die Entwicklung neuer Lernverfahren welche die Differenzierbarkeit der Aktivierungsfunktion voraussetzten (z.B. Backpropagation) entstanden neue Neuronenmodelle. Ein Beispiel für diese Klasse von Funktionen ist die logistische:

$$f_{logistic}(x) = \frac{1}{1 + e^{-c \cdot x}} \quad mit \quad c > 0$$

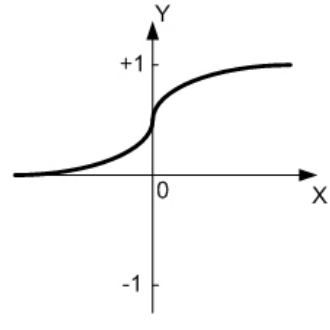

Abbildung 6. Logistische Aktivierungsfunktion

Die Berechnung der **Ausgabe** übernimmt fast ausschließlich die Identität obwohl andere Zusammenhänge theoretisch möglich sind. Aus diesem Grund wird oft die Aktivierung direkt als Ausgabe des Neurons bezeichnet.

Der Begriff **Perzeptron** bündelt meist folgende Spezifikationen:

- Summe der gewichteten Eingaben als Propagierungsfunktion
- Aktivierungszustand wird durch Schwellwertfunktion bestimmt
- Ausgabefunktion ist die Identität

Im weiteren Verlauf betrachten wir, wenn nicht anders gekennzeichnet, ein solches Perzeptron als Beispiel für künstliche Neuronen.

3 Der Lernvorgang

3.1 Langfristige Speicherung von Informationen im Gehirn

Die Neuronen im Gehirn des Menschen sind wie bereits beschrieben miteinander vernetzt. Doch all diese Kommunikationswege sind nicht statisch. Die Stärke einer Verbindung zwischen Neuronen variiert, neue werden geknüpft, bestehende umgeleitet - die Struktur des Netzes aus Nervenzellen verändert sich. Diese Vorgänge bezeichnet man allgemein als **Lernen**. Verbindungen die zu korrekten Antworten führen werden gestärkt, andere geschwächt. Bei diesem so genannten "reinforcement learning" bedarf es einer Vorbildsperson (z.B. Eltern, Lehrmeister) die vorgibt ob die Reaktion auf eine gegebene Situation richtig oder falsch war. Positive Verstärkung (Lob) oder Zurechtweisung (Tadel) führen zu einer Verhaltensmodifikation. Den meisten Menschen ist dabei gemein, dass ein Sacherverhalt nicht gleich bei der ersten Konfrontation dauerhaft gespeichert wird. Langfristige Veränderungen im Gehirn treten erst ein wenn eine Situation viele Male wiederholt wurde. Nur derartig eintrainierte Informationen werden ins Langzeitgedächtnis übernommen.

3.2 Übertragung auf künstliche Neuronen am Beispiel der Lernstrategie des Perzeptrons

Vom biologischen Lernvorgang inspiriert entwickelten sich auch die ersten Lernstrategien künstlicher Neuronen. Beim überwachten Lernen (supervised learning) liegen in der Lernphase zu jedem Beispiel eines Problems die erwarteten Ausgabedaten vor. Werden dem Netz die Eingaben übergeben, ist seine Ausgabe mit der erwarteten zu vergleichen um den genauen Fehler zu bestimmen. In weiteren Beispielen wird danach versucht diesen Fehler durch Anpassung der Gewichtungen zu minimieren.

$$e(i) = A_d(i) - A(i)$$

Die **Fehlerfunktion** liefert die Abweichung der Ausgabe $A(i)$ von der erwarteten $A_d(i)$ in der i-ten Iteration. Ist die Abweichung positiv, so war der Ausgabewert des Neurons zu gering und muss erhöht werden. War sie dagegen negativ, versucht man sie durch justieren der Gewichte zu verringern.

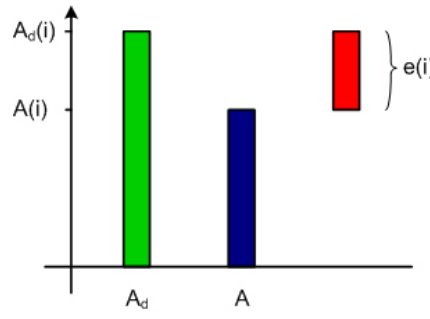

Abbildung 7. Darstellung der resultierenden Abweichung $e(i)$ der Ausgabe $A(i)$ von der erwarteten $A_d(i)$ in der i-ten Iteration

Dabei ist zu beachten dass die Eingaben dem reellen Zahlenbereich entnommen sein können, multipliziert mit den Gewichten dementsprechend additiv oder subtraktiv auf die Ausgabe wirken ($Eingabe_j = w_j \cdot x_j$). Ist eine Eingabe positiv so bewirkt eine Erhöhung seines Gewichtes ebenfalls ein Anwachsen der Ausgabe, ist sie dagegen negativ schrumpft der Ausgabewert des Neurons.

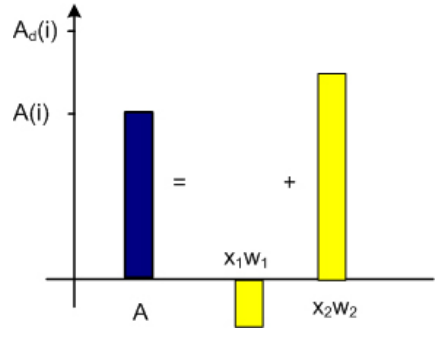

Abbildung 8. Additive und subtraktive Auswirkungen von Eingaben auf die Propagierung und Ausgabe

Bezieht man diesen Sachverhalt mit ein ergibt sich folgende Formel zur Anpassung des j-ten Gewichts eines Perzeptrons für die nächste (i+1 te) Iteration:

$$w_j(i+1) = w_j(i) + \alpha \cdot x_j(i) \cdot e(i)$$

Der konstante positive Faktor α wird hierbei als Lernrate bezeichnet.

Insgesamt ergibt sich als Lernstrategie für Rosenblatts Perzeptron folgender Algorithmus:

(1) Gewichte und Schwellenwert werden zufällig initialisiert
(2) Das Neuron wird mit dem ersten Beispiel konfrontiert
(3) Aktivierungszustand und Ausgabe werden berechnet
(4) Berechnung des Fehlers und Anpassung der Gewichte
(5) Auswahl des nächsten (i+1 ten) Testbeispiels und fortfahren mit (3)

Dieses Verfahren wird so lange ausgeführt bis die Fehlabweichung die gewünschte Toleranz (z.B. =0) erreicht.

3.3 Lernvorgang eines Perzeptrons für die logische Funktion UND

l Lernphase
x_1, x_2 Eingaben
w_1, w_2 Ausgangsgewichte
A Ausgabe
e Fehler
w_{n1}, w_{n2} Neue angepasste Gewichte
Die Konstante Θ wurde = 0.2, $\alpha = 0.1$ gewählt.

l	x_1	x_2	A_d	w_1	w_2	A	e	w_{n1}	w_{n2}
1	0	0	0	0.3	-0.1	0	0	0.3	-0.1
	0	1	0	0.3	-0.1	0	0	0.3	-0,1
	1	0	0	0,3	-0,1	1	-1	0,2	-0,1
	1	1	1	0,2	-0,1	0	1	0,3	0,0
2	0	0	0	0,3	0,0	0	0	0,0	0,0
	0	1	0	0,3	0,0	0	0	0,3	0,0
	1	0	0	0,3	0,0	1	-1	0,2	0,0
	1	1	1	0,2	0,0	1	0	0,2	0,0
3	0	0	0	0,2	0,0	0	0	0,2	0,0
	0	1	0	0,2	0,0	0	0	0,2	0,0
	1	0	0	0,2	0,0	1	-1	0,1	0,0
	1	1	1	0,1	0,0	0	1	0,2	0,1
4	0	0	0	0,2	0,1	0	0	0,2	0,1
	0	1	0	0,2	0,1	0	0	0,2	0,1
	1	0	0	0,2	0,1	1	-1	0,1	0,1
	1	1	1	0,1	0,1	1	0	0,1	0,1
5	0	0	0	0,1	0,1	0	0	0,1	0,1
	0	1	0	0,1	0,1	0	0	0,1	0,1
	1	0	0	0,1	0,1	0	0	0,1	0,1
	1	1	1	0,1	0,1	1	0	0,1	0,1

Anhand dieser Tabelle erkennt man deutlich dass der berechnete Fehler nicht unbedingt monoton gegen 0 konvergieren muss. Bereits richtig gelöste Beispiele können im Verlauf der Lernphasen auch wieder falsch beantwortet werden. Hier werden erst im letzten Abschnitt 5 der Lernreihe alle Operationen vom Perzeptron korrekt "berechnet".

Stellt man die Ergebnisse dieser logischen Operation in einem Diagramm dar ergibt sich folgendes Bild:

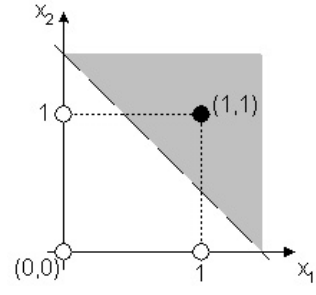

Abbildung 9. Argumentraum und Ergebnis der logischen Operation UND

Schwarze Punkte stellen dabei Punkte dar deren Ergebnis 1, weiße deren Ergebnis 0 ist. Die Gerade deutet den Sprung in der zur Aktivierung verwendeten Schwellwert- oder Stufenfunktion bei $x_{net} = 0$ an und teilt den Argumentraum in zwei Halbebenen, deren enthaltene Ergebnisse in der einen Halbebene 1, in der anderen 0 sind.

$$x_{net} = 0 \Leftrightarrow f_{net}(x) = \sum_{i=1}^{n} x_i w_i - \Theta = 0 \Leftrightarrow x_1 w_1 + x_2 w_2 - \Theta = 0$$

Das hier verwendete Perzeptron hätte ohne Veränderung der verwendeten Funktionen und Konstanten auch die logische Operation ODER lernen können. Die grafische Darstellung hierfür wäre dieses Diagramm:

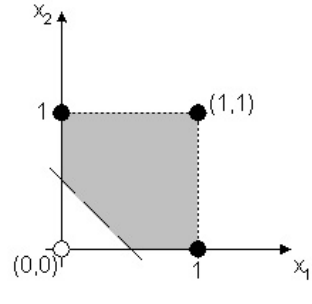

Abbildung 10. Argumentraum und Ergebnis der logischen Operation ODER

Aufgrund der veränderten Gewichte befindet sich die Trennlinie an einer anderen Stelle im Koordinatesystem teilt aber wieder den Argumentraum in die Halbebene der *wahren*, und die der $falschen$ Ergebnisse.

Aus diesen Diagrammen erkennt man leicht warum ein Perzeptron auch als LTU - "Linear Treshold Unit" bezeichnet werden. Die zu lernende Funktion muss **linear separierbar**, ihr Argumentraum also mit einer Hyperebene der Dimension $n-1$ in eine Halbebene der wahren und eine der falschen Ergebnisse zerlegbar sein.

Konvergenz-Theorem von Rosenblatt: Ein einstufiges Perzeptron kann jede linear separierbare Funktion in endlicher Zeit erlernen.

3.5 Eingeschränkte Leistungsfähigkeit des einschichtigen Perzeptrons

Hier zeigen sich die Grenzen eines einzelnen Perzeptrons welche Minsky und Papert 1969 in ihrem Buch "Perceptrons" veröffentlichten und bewiesen. Nicht linear separierbare Funktionen wie z.B. XOR können nicht von einem einzelnen Perzeptron erlernt werden:

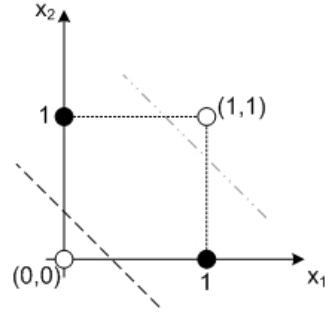

Abbildung 11. Argumentraum und Ergebnis der logischen Operation XOR

Beweis nach Minsky/Papert: Perceptrons, MIT-Press, 1969:
Angenommen, es gibt Koeffizienten $w_1, w_2, \Theta \in R$, so dass für alle $x_1, x_2 \in \{0, 1\}$ gilt:

$$x_1 \oplus x_2 = 1 \Leftrightarrow w_1 x_1 + w_2 x_2 - \Theta \geq 0$$

Da $x_1 \oplus x_2 = x_2 \oplus x_1$ gilt, folgt:

$$x_1 \oplus x_2 = 1 \Leftrightarrow w_1 x_2 + w_2 x_1 - \Theta \geq 0$$

Nach addieren der beiden rechten Seiten erhalten wir:

$$x_1 \oplus x_2 = 1 \Leftrightarrow (w_1 + w_2)\, x_1 + (w_1 + w_2)\, x_2 - 2\Theta \geq 0 \Leftrightarrow (w_1 + w_2)\, (x_1 + x_2) - 2\Theta \geq 0$$

Fassen wir den auf der rechten Seite vorkommenden Term als Funktion von $z := x_1 + x_2$ auf:

$$f(z) := (w_1 + w_2)\, z - 2\Theta$$

Eine lineare Funktion mit höchstens einer Nullstelle.

Aus dem Diagramm ersichtlich müsste die Funktion folgender Werte liefern:

$f(0) < 0$, da für z = 0 (entspricht $x_1 = x_2 = 0$) $x_1 \oplus x_2 = 0$ ergibt.

$f(1) > 0$, da für z = 1 (entspricht $x_1 = 1$ und $x_2 = 0$, oder umgekehrt) $x_1 \oplus x_2 = 1$ ergibt.

$f(0) < 0$, da für z = 2 (entspricht $x_1 = x_2 = 1$) $x_1 \oplus x_2 = 0$ ergibt.

Dies würde jedoch eine Kurve mit 2 Nullstellen ergeben.

$\Rightarrow Widerspruch.$

Diese Veröffentlichung von Minsky und Papert bremste die Euphorie der damaligen Zeit, neuronale Netzte als Lösung aller Probleme zu sehen, und hemmte die Weiterentwicklung viele Jahre.

4 Leistungsverbesserung durch Aufbau eines Netzes

Ende der 80er Jahre machten es die Fortschritte in der Computertechnik möglich größere neuronale Netze zu betrachten. Die damalige Entwicklung des Backpropagation-Algorithmus gab den neuronalen Netzen zusätzlich neuen Aufschwung. Mehrstufigen Perzeptronen war es damit möglich jede boolesche Funktion zu realisieren.

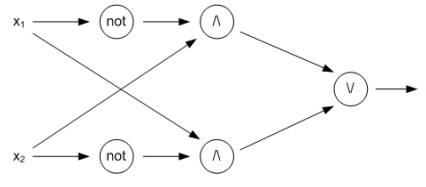

Abbildung 12. Schematischer Aufbau von XOR über der Basis (Und, Negation)

4.1 Verschiedene Architekturen

Vergleichbar mit den menschlichen Neuronen deren Anzahl in Größenordnugen von $10^{10} - 10^{11}$ liegt und dabei $10^{13} - 10^{14}$ Verbindungen untereinander eingehen wurden jetzt auch künstliche neuronale Netze vergrößert und neue Architekturen entwickelt.

Vorwärts verkettetes Netz:

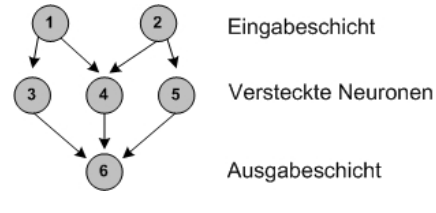

Abbildung 13. Schematische Darstellung, vorwärts verkettetes Netz

Diese Architektur spiegelt zum ersten Mal das Prinzip mehrerer Schichten wieder. In der **Eingabe-Schicht** befinden sich Neuronen zu denen keine Verbindungen führen. Die **Ausgabe-Schicht** beinhaltet in vorwärts gerichteten Netzen die Neuronen von denen keine Verbindungen ausgehen. Neuronen die zu keiner der genannten Schichten gehören werden als **versteckt** bezeichnet.

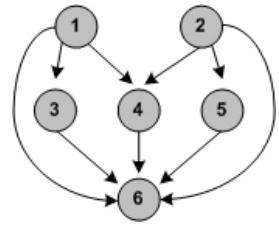

Abbildung 14. Schematische Darstellung, vorwärts verkettetes Netz mit Shortcuts

Wie in Abbildung 14 sind auch Verbindungen möglich die eine oder mehrere Schichten überspringen.

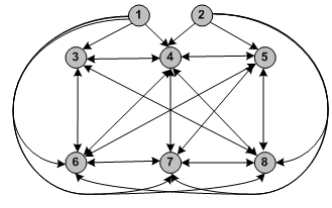

Abbildung 15. Schematische Darstellung, vorwärts verkettetes Netz mit voll vernetzter Schicht

Die Architektur in Abbildung 15 beinhaltet nun auch Rückkopplungen. Die Eingabeschicht (Neuronen 1 und 2) ist weiterhin vorhanden und nach Definition nicht rückgekoppelt.

4.2 Entwicklung eines neuronalen Netzes

Bisher existiert noch kein Algorithmus zur Entwicklung eines erfolgreichen neuronalen Netzes für ein gegebenes Problem. Es wird eine ausreichende Anzahl von Beispieldaten benötigt um das Netz "einzulernen". Weiterhin ist oft eine Menge von Testdaten erforderlich um die Generalisierungsfähigkeit eines entwickelten Netzes zu prüfen. Die Fähigkeit aus gegebenen Beispielen auf Lösungen für bis dahin unbekannte Eingaben zu schließen. Um ein neuronales Netz an die Aufgabenstellung anzupassen kann folgender Ablauf ein möglicher Weg sein:

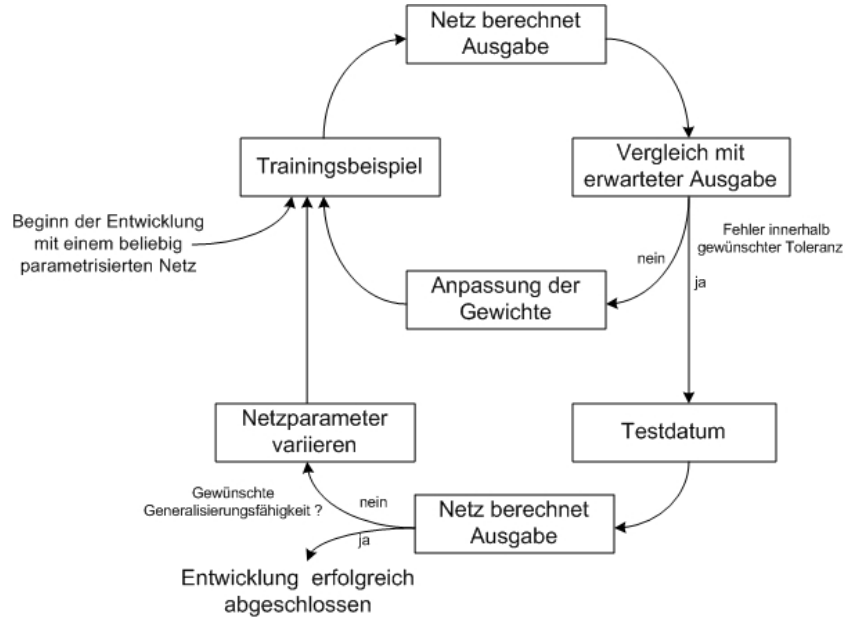

Abbildung 16. Entwicklung eines neuronalen Netzes

Die Basis der Leistungsfähigkeit eines neuronalen Netzes ist dabei nicht in komplexen Funktionen innerhalb der Neuronen zu suchen, sondern besteht darin, mit einer großen Anzahl einfacher Verarbeitungseinheiten durch massive Verschaltung und Parallelisierung ein Problem zu lösen, das nicht von einem Algorithmus beschrieben werden kann. Die in vielen Beispielen antrainierten "Fertigkeiten" des Netzes erlauben es ihm auch auf möglicherweise nicht vorhersehbare Situationen angemessen zu reagieren.

References

[1] Michael Negnevitsky: Artificial Intelligence: A Guide to Intelligent Systems, Addison Wesley, Band 1, 2001, ISBN: 0-201-71159-1

[2] Uwe Lämmel, Jürgen Cleve: Lehr- und Übungsbuch, Künstliche Intelligenz, Carl Hanser Verlag, 2001

[3] Prof. Dr. Uwe Schöning: Boolsche Funktionen, Logik, Grammatiken und Automaten, (Theoretische Informatik II, Vorlesungsskript)

[4] Holger Gläßel: Neuronale Netze, `http://www.isd.uni-stuttgart.de/arbeitsgruppen/pigroup/vorlesung_ki/`