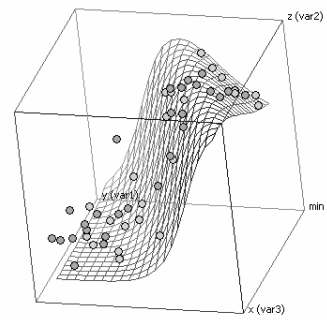
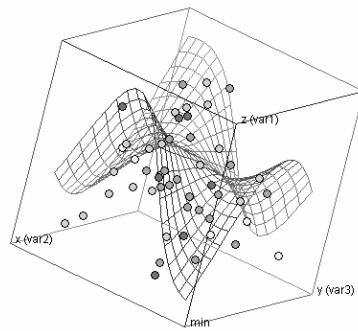
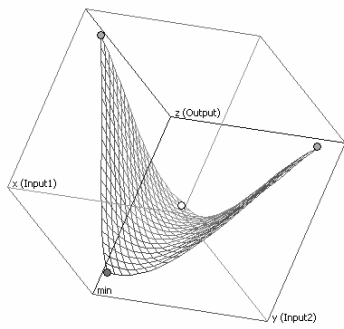
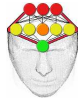


Statistik mit Neuronalen Netzen. Eine anwendungsorientierte Einführung
Andreas Fischer (2011)





Durch das Schätzen erst gibt es den Wert,
und ohne das Schätzen
wäre die Nuss des Daseins hohl.
- Friedrich Nietzsche

Zur Beschreibung von Zusammenhängen zwischen Phänomenen finden in der klassischen Statistik häufig relativ simple Modelle Verwendung: „Je mehr von Prädiktor A, desto mehr von Kriterium B“, oder „Immer wenn A der Fall ist, geschieht B“, oder „Je mehr von Prädiktor A, desto wahrscheinlicher wird es, dass B der Fall ist“, usw.

Für derlei Beschreibungen von Zusammenhängen ist es charakteristisch,

- dass sie nur wenige Variablen enthalten – oft werden z.B. auch komplexe Sachverhalte *monokausal erklärt* bzw. beschrieben, d.h. sie werden auf eine einzige Ursache zurückgeführt;
- dass die Effekte einer jeden erklärenden Variable als unabhängig vom Zustand jeder anderen erklärenden Variable beschrieben werden – oft werden z.B. *Interaktionseffekte vernachlässigt*, wie die Randbedingungen, unter denen die Effekte auftreten;
- dass die Effekte der erklärenden Variablen *linear beschrieben* werden – oft werden mit solchen „je mehr desto mehr“-Aussagen z.B. natürliche Sättigungsgrenzen vernachlässigt;

Um auch komplexe Wirkungsgefüge (in denen zahlreiche Variablen stets mehr oder weniger linear wirken und interagieren können) angemessen zu beschreiben, lässt sich der Formalismus *Neuronaler Netze* heranziehen (zum menschlichen Umgang mit komplexen Systemen vgl. Fischer, Greiff und Funke, 2012).

Schon seit längerem bereichert die Analyse von Netzwerken, bzw. Systemen (Elementen die über diverse Relationen miteinander verbunden sind), ein breites Spektrum an Wissenschaftsdisziplinen sowohl in qualitativen als auch in quantitativen Forschungsarbeiten (so lassen sich Interviewdaten zu Beziehungen zwischen Menschen –vgl. z.B. Schnegg & Lang, 2002– häufig ebenso als Netzwerke darstellen, wie die lineare oder nichtlineare –z.B. logistische– Regression von Kriteriums- auf Prädiktorvariablen. Während netzwerkanalytische Überlegungen in Politologie, Soziologie und Ethnographie weit verbreitet sind um Daten zu *sozialen Netzwerken* systematisch zu erheben und auszuwerten, stellen sie in Psychologie, Biologie und Neurologie einen Zugang zur systematischen Beschreibungen und Beforschung (biologischer) *neuronaler Netzwerke* zur Verfügung. Künstliche Neuronale Netze¹ (KNN) bzw. konnektionistische Modelle erlauben unter anderem die biologisch inspirierte Modellierung psychologisch relevanter Strukturen und Prozesse – wie z.B. Wahrnehmung, Lernen und Gedächtnis (für einen Überblick vgl. Sun, 2008), Spracherwerb (Szagun, 2006) oder das Phänomen der Farbkonstanz (Stanikunas et al., 2004). Sie stellen neben Produktionssystemen und anderen symbolverarbeitenden Ansätzen den zweiten großen Ansatz der künstlichen Intelligenz dar (Lämmel & Cleve, 2008), als welcher sie auch in der Psychologie des Denkens allgemein (Eysenck & Keane, 2005) und speziell in der Psychologie des Problemlösens (Funke, 2003) mit Gewinn rezipiert werden. Für eine gelungene Einführung in den Gegenstandsbereich KNN vgl. v.a. Rey und Wender (2008) sowie Lämmel und Cleve (2008).

Netzwerke lassen sich auf vielfältige Weisen repräsentieren -z.B . kann man ein und dasselbe Netzwerk als Matrix, gerichteten Graph, Liste geordneter Paare oder als Strukturgleichung darstellen (vgl. Abb.1).

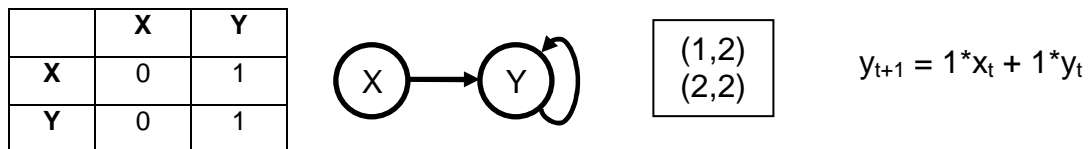
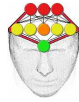


Abbildung 1. Ein Netzwerk aus zwei Elementen und zwei Relationen als Matrix, gerichteter Graph, Liste und als Strukturgleichung

¹ Das Adjektiv „künstlich“ hat sich in der Forschergemeinschaft zur „künstlichen Intelligenz“ (K.I.) eingebürgert, um die formale Beschreibung Neuronaler Netze vom biologischen Vorbild abzugrenzen. Im Folgenden wird auf das Adjektiv nur optional verwendet, wobei mit „Neuronalen Netzen“ –sofern nicht anders angegeben– stets der Formalismus angesprochen ist.



Als quantitative Methoden der statistischen Datenauswertung lassen sich Neuronale Netze *auch* bei der statistischen Datenauswertung nutzbringend einsetzen¹ (vgl. Fischer, 2010):

Viele klassische uni- und multivariate Analysemethoden lassen sich im formalen Rahmen Neuronaler Netze darstellen (z.B. lineare, logistische & polynomiale Regressionsanalysen, Diskriminanzanalysen, Varianzanalysen & Mittelwertvergleiche, Hauptkomponentenanalysen sowie k-means-Clusteranalysen). Neuronale Netze liefern dabei neben einem einfachen und sehr intuitiv verständlichen formalen Rahmen für herkömmliche Verfahren auch die Möglichkeit, über selbige hinaus zu gehen: *Im Rahmen Neuronaler Netze lassen sich beliebig komplexe und potentiell nichtlineare statistische Modelle generieren und als Erweiterungen einfacherer klassischer Verfahren begreifen* (z.B. unterscheidet sich die lineare von der logistischen Regression im Rahmen Neuronaler Netze lediglich in der Wahl einer anderen Aktivitätsfunktion, und komplexere nichtlineare Verfahren ergeben sich schlicht durch die Addition mehrerer solcher Aktivitätsfunktionen). Die Parameter dieser Modelle lassen sich über Algorithmen schätzen, die genau bekannte statistische Eigenschaften haben (z.B. lassen sich die aus der klassischen Statistik bekannten Kleinste-Quadrate-Schätzer und Maximum-Likelihood-Schätzer auch für komplexe Neuronale Netze errechnen). KNN stellen unter gewissen Umständen v.a. dann die Methode der Wahl dar, wenn keine begründeten Hypothesen über die Art der Zusammenhänge aufgestellt werden können (vgl. Backhaus et al., 2006; Rey & Wender, 2008) –insb. wenn man sich im Unklaren über die Art und die Relevanz von Interaktionen zwischen den Prädiktorvariablen ist (vgl. Rumelhart et al., 1995).

Die vorliegende Arbeit sieht sich von der Motivation getragen, die grundlegende Gemeinsamkeiten zwischen den Modellen Neuronaler Netze und einigen gängigen Modellen Klassischer Statistik besonders deutlich herauszuarbeiten, um einen Ausblick darauf zu geben, inwiefern Neuronale Netze in der statistischen Datenauswertung nutzbringend verwendet werden können. Dabei beschränkt sich die vorliegende Arbeit auf eine Zusammenstellung und Ausarbeitung *deskriptivstatistischer* Anwendungen Neuronaler Netze. Inferenzstatistische Aussagen über die ermittelten Parameter lassen sich z.B. auf Basis neuerer rechenaufwändiger Ansätze wie „Bootstrapping“ treffen (hierzu sei verwiesen auf Fox, 2002, oder auch Papadopoulos et al., 2000). In vielen Fällen können – aufgrund formaler Entsprechungen Künstlicher Neuronaler Netze mit Modellen Klassischer Statistik – sogar die klassischen inferenzstatistischen Ansätze genutzt werden (vgl. z.B. Fahrmeir et al.). Als weitere Einführungsliteratur in die Datenauswertung vermittelt Neuronaler Netze sei beispielsweise verwiesen auf Sarle (1994), Rumelhart et al. (1995), Rey & Wender (2008), Rey (2009) oder auch auf das Statistik-Lehrbuch von Backhaus et al. (2006).

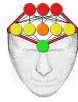
Zur Berechnung Künstlicher Neuronaler Netze existiert eine Reihe empfehlenswerter Programme, allen voran

- MemBrain, Visual-XSel und SPSS (für eine Einführung in die Verwendung Neuronaler Netze mithilfe dieser Programme vgl. Rey&Wender, 2010), oder die gängige kostenfreie Statistikumgebung GNU R (insbesondere die Pakete AMORE, nnet, neural, oder som),
- oder die pädagogisch motivierte Statistikumgebung FISCHER – die speziell dafür konzipiert wurde, Parallelen zwischen klassischer und neuronaler Verfahren in der statistischen Datenauswertung (vgl. oder Sarle, 1994) zu veranschaulichen (vgl. Fischer, 2010; das Programm steht zum kostenlosen Download bereit auf der Seite <http://www.psychologie.uni-heidelberg.de/ae/allg/mitarb/af/index.html>).
- Auch mit herkömmlichen Programmen zur Tabellenkalkulation (namentlich Open Calc oder Microsoft Excel) lassen sich Neuronale Netze auf verschiedene Weise leicht selbst berechnen: Einerseits steht in diesen eine tatkräftige Programmiersprache zur Verfügung (in Excel VisualBasic, in OpenCalc Java), mit der sich auch Neuronale Netze programmieren lassen (ein Code-Beispiel für ein einfaches Neuronales Netz in VisualBasic findet sich z.B. unter <http://www.psychologie.uni-heidelberg.de/ae/allg/mitarb/af/index-Dateien/DeltaregelExcelMakro.txt>), andererseits besteht die Möglichkeit über sog. „solver“ die nonlinearen Gleichungen Neuronaler Netze über die Optimierungsalgorithmen des Programms zu berechnen (für die Berechnung Neuronaler Netze via Excel sei z.B. auf Macho, 2002, verwiesen).

Im Folgenden soll nun das kontinuierliche Grundmodell zur Funktionsweise von Neuronen auseinandergesetzt werden², um darauf aufbauend einige Parallelen und Entsprechungen zwischen Künstlichen Neuronalen Netzen und den gängigen Verfahren klassischer Statistik herauszuarbeiten (vgl. hierzu v.a. Sarle, 1994).

¹ Dabei scheint wenig folgerichtiger zu sein, als die Formalismen, die auch zur Beschreibung des menschlichen Erkenntnisapparates herangezogen werden (der ja aus einer Mannigfaltigkeit einzelner Erfahrungen die Begriffe ableitet mit denen er operiert) auch zu nutzen um aus wissenschaftlichen Daten Erkenntnisse zu gewinnen und statistische Schlüsse zu ziehen.

² Andere Modelle unterscheiden sich von diesem Grundmodell i.d.R. durch einen geringeren Abstraktionsgrad auf Kosten der einfachen und effizienten Berechenbarkeit und finden daher in der statistischen Datenauswertung seltener Verwendung – der Interessierte Leser sei für Modelle, die sich näher am Biologischen Vorbild orientieren, z.B. auf Artikel zu *„Pulsed Neurons“* oder zum *„Hodgkin-Huxley-Modell“* verwiesen.



1. Neuronale Netze – Ein Definitionsversuch

Neuronale Netze lassen sich verstehen als informationsverarbeitende Systeme aus Neuronen (sog. *Elementen* oder *Units*) und ihren Synapsen (sog. *Relationen* oder *Verbindungen*). Die Stärke bzw. das Gewicht einer Verbindung zwischen zwei Units wird durch eine positive oder negative reelle Zahl ausgedrückt.

1.1 Units

Units befinden sich stets in einem bestimmten Erregungszustand (*Aktivierung*), können über eingehende Verbindungen von anderen Units erregt werden (*Propagierung*) und über ausgehende Verbindungen Erregung an andere Units weitergeben (*Ausgabe*) -vgl. Abbildung 2. Die Aktivierung einer Unit wird über eine positive oder negative reelle Zahl ausgedrückt.

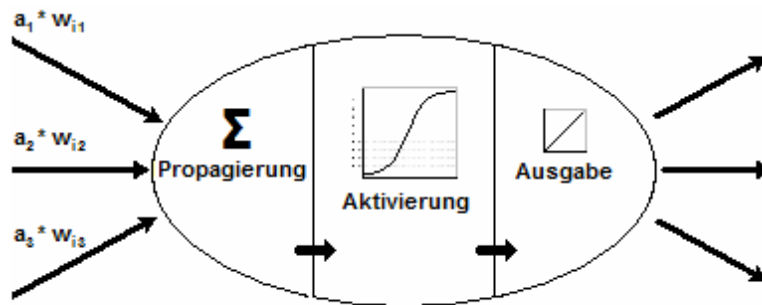


Abbildung 2: Visualisierung der Funktionsweise einer Unit

Wenn eine Unit von einer anderen erregt wird, verarbeitet sie diese Information in drei Schritten:

- 1) **Propagierung:** Die Summe der gewichteten Aktivitäten aller an die empfangende Unit i sendenden Units j ergibt für die empfangende Unit i einen sogenannten *Net-Input* net_i .

$$net_i = \sum(a_j * w_{ij})$$

- 2) **Aktivierung:** aus dem in Schritt 1 errechneten Net-Input ergibt sich über eine Aktivierungsfunktion f_{akt} der sog. *Aktivitätslevel* a_i -vgl. Abbildung 3.

$$a_i = f_{akt}(net_i)$$

- 3) **Ausgabe:** aus dem in Schritt 2 errechneten Aktivitätslevel wird als sog. *Output* o_i diejenige Aktivität errechnet, die die Unit an weitere Units oder an die Umwelt weitergibt. In der Regel besteht der Output o_i einer Unit einfach aus dem Aktivitätslevel a_i .

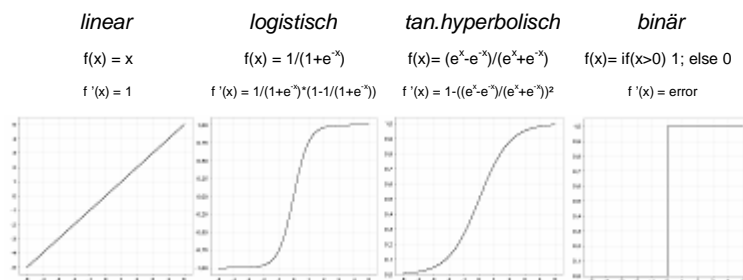
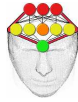


Abbildung 3: Typische Aktivitätsfunktionen



1.2 Verbindungen

Je nachdem, woher eine Unit ihren Input zur Propagierung erhält oder wohin sie ihren Output im Zuge der Ausgabe sendet hat es sich eingebürgert, verschiedene Arten von Units zu unterscheiden (vgl. Abbildung 4):

Inputunits werden von der Umwelt des Systems zur Aktivität angeregt (ihre Ausgabe wird von außen in Form einer Zahl vorgegeben).

Outputunits geben ihren Aktivitätslevel z.B. an Muskelzellen oder sonstige Effektoren ab (ihre Ausgabe lässt sich in der Außenwelt beobachten bzw. als Zahl ablesen).

Hiddenunits befinden sich zwischen Input- und Outputunits.

Biasunits sind Inputunits, die konstant die Ausgabe 1 liefern.

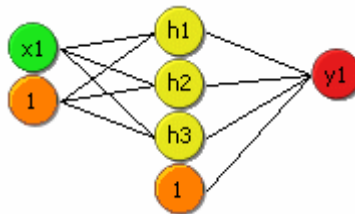


Abbildung 4: Neuronales Netz mit einer Inputunit, zwei Biasunits, drei Hiddenunits und einer Outputunit

Es gibt Netze mit Hiddenunits, aber auch Netze ohne Hiddenunits. Ebenso gibt es Netze mit und Netze ohne Biasunits. In sog. „Vorwärtsgerichteten Netzen“ („feedforward networks“) – und auf solche wird sich die vorliegende Arbeit im wesentlichen Beschränken – sind die Units zu sogenannten Schichten/Layers gruppiert: Die Inputunits werden zur **Inputschicht**, die Hiddenunits zur **Hiddenschicht** und die Outputunits zur **Outputsicht** zusammengefasst, wobei

- Units einer Schicht untereinander nicht verbunden sind, und
- jede Unit einer Schicht mit allen Units der folgenden Schicht verbunden ist.

Die Gewichte der Verbindungen zwischen den Units werden i.d.R. über iterative Schätzverfahren bestimmt, die im Kontext Neuronaler Netze auch „Lernregeln“ genannt werden.

Gebräuchliche Lernregeln sind z.B.

- die Deltaregel (vgl. auch „Generalisierte Deltaregel“, „Backpropagation-Algorithmus“)
- die Hebb'sche Regel (vgl. auch „Generalisierte Hebb'sche Regel“, „Sanger-Algorithmus“)
- der Kohonen-Algorithmus (vgl. auch „Self-Organizing-Maps“)



2. Lernregeln und ihre Parallelen zur klassischen Statistik

2.1 Deltaregel

Die Deltaregel stellt eine Lernregel für Neuronale Netze dar. Sie gibt vor, wie die Gewichte zwischen den Units iterativ zu verändern sind, wenn man erreichen will, dass das Netz bei einem gegebenen Input (sprich: Ausgabe der Inputunits) bestmöglich einen bestimmten Output (sprich: Ausgabe der Outputunits) liefert. Die Art der Fehlerfunktion, die dabei minimiert wird, ist abhängig davon, welche Variante der Deltaregel man heranzieht. Da zu jedem Input ein gewünschter Output vorzugeben ist, spricht man von "supervised training".

In der ursprünglichen Formulierung von Widrow und Hoff (1989) wird für lineare Aktivitätsfunktionen die Summe der quadrierten Abweichungen zwischen vorhergesagten und tatsächlichen Werten (Summe der Abweichungsquadrate, SAQ) minimiert. Dies geschieht, indem man die Gewichte des Netzes, welche die Outputunits i mit den Units j der vorangegangenen Schicht verbinden, iterativ um folgenden Wert Δw_{ij} verändert – die folgende Formel ist auch als „Widrow-Hoff-Regel“ bekannt:

$$\Delta w_{ij} = \varepsilon \cdot \delta_i \cdot a_j, \quad \text{wobei } \delta_i = y_i - a_i;$$

Für die lineare Aktivitätsfunktion stellen die Gewichte, welche die SAQ minimieren (*Kleinste-Quadrate-Schätzer*), zugleich diejenigen Schätzer dar, welche unter Annahme normalverteilter Abweichungen mit der größten Wahrscheinlichkeit den beobachteten Daten zugrunde liegen (*Maximum-Likelihood-Schätzer*). Die sog. "kanonische" (oder „natürliche“) Aktivitätsfunktion bzw. „Linkfunktion“ zur Normalverteilung der Fehler ist die lineare Aktivitätsfunktion (vgl. Fahrmeir et al. 2007). *Auch für andere Aktivitätsfunktionen führt die Widrow-Hoff-Regel zu Maximum-Likelihood-Schätzern: Nimmt man für die Abweichungen diejenige Verteilung an, deren kanonische Linkfunktion die gewählte Aktivitätsfunktion ist, so lässt sich zeigen, dass die aus der Widrow-Hoff-Regel resultierenden Schätzer auch diejenigen mit der größten Wahrscheinlichkeit (d.h. Maximum-Likelihood-Schätzer) sind, sofern sich die resultierende Verteilung der Abweichungen in Form einer einparametrischen Exponentialfamilie (z.B. Normal-, Binomial-, Poisson-, oder Gammaverteilung) beschreiben lässt (vgl. Rumelhart et al., 1995).* Maximum-Likelihood-Schätzer resultieren also nachweislich für eine Reihe ausgewählter Aktivitätsfunktionen:

- lineare Aktivitätsfunktion (unter Annahme der Normalverteilung)
- logistische Aktivitätsfunktion (unter Annahme der Binomialverteilung)
- logarithmische Aktivitätsfunktion (unter Annahme der Poissonverteilung)

Will man für beliebige Aktivitätsfunktionen die SAQ minimieren, bzw. die Wahrscheinlichkeit der Schätzer unter Annahme normalverteilter Abweichungen auch für nonlineare Aktivitätsfunktionen maximieren, dann lässt sich eine weitere Variante der Deltaregel heranziehen. Diese wurde z.B. von Rumelhart et al. (1989) abgeleitet (im Folgenden auch „Rumelhart-Regel“):

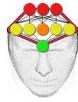
$$\Delta w_{ij} = \varepsilon \cdot \delta_i \cdot a_j \cdot f_{\text{akt}}'(\text{net}_i), \quad \text{wobei } \delta_i = y_i - a_i$$

Auch für andere Fehlerverteilungen lassen sich Formeln ableiten, die für beliebige Aktivitätsfunktionen zu Maximum-Likelihood-Schätzern führen. Für die Binomialverteilung einer binären Outputvariable findet man entsprechende Formeln z.B. bei Spackman (1992) (vgl. auch Rumelhart et al., 1995):

$$\Delta w_{ij} = \varepsilon \cdot \delta_i \cdot a_j \cdot f_{\text{akt}}'(\text{net}_i), \quad \text{wobei } \delta_i = (-a_i)^{-y_i} \cdot (1-a_i)^{-(1-y_i)}$$

Dem findigen Leser wird aufgefallen sein, dass die bisher beschriebenen Formeln sich ausschließlich auf Gewichte beziehen, die mit der Outputunit in Verbindung stehen. Für Neuronale Netze mit Hiddenunits haben Rumelhart et al. (1989) Formeln abgeleitet, mit denen sich die Gewichte der bisher nicht behandelten Verbindungen modifizieren lassen, sofern man die Abweichungen δ_m jeder Unit m der folgenden Schicht kennt. Dieser Ansatz wird auch als *Backpropagation* oder *Generalisierte Deltaregel* bezeichnet. Man modifiziert die Verbindung zwischen jeder Hiddenunit k und der Unit l der vorangegangenen Schicht demnach um:

$$\Delta w_{lk} = \varepsilon \cdot \delta_k \cdot a_l \cdot f_{\text{akt}}'(\text{net}_k), \quad \text{wobei } \delta_k = \sum (w_{km} \cdot \delta_m)$$



2.1.1 Algorithmus

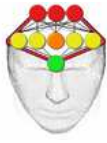
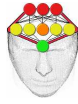
```
Setze alle Gewichte  $w_{ij}$  des Netzes auf 0
FOR EACH [Epoche1] DO {
  FOR EACH [Pattern2] DO {
    • Bestimme für Pattern  $p$  die Ausgabewerte  $a_j$  jeder Inputunit  $j$ ;
    • Berechne Ausgabe  $a_i$  der Outputunit  $i$ ;
    • Bestimme Differenz zwischen  $a_i$  und dem gewünschten Kriterium  $y_i$ ;
    • Verändere für jede sendende Unit  $x$  und jede empfangende Unit  $y$ 
      das Gewicht  $w_{xy}$  um den adäquaten Wert  $\Delta w_{xy}$  (Formeln im Fließtext)
  }
}
```

Anmerkungen zum Algorithmus:

- Der Term ε in der Formel zur Berechnung von Δw_{ij} stellt einen Parameter dar. Durch diesen Parameter wird auch bei Inputs, deren Beträge >1 sind, durch die Gewichtsveränderung tatsächlich eine Verbesserung gewährleistet, sodass sich das Netz nicht kontinuierlich verschlechtert. Sollten die Inputvariablen also Werte aufweisen, deren Betrag >1 ist, ist ein hinreichend kleiner Lernparameter ε von besonderer Wichtigkeit. Ein brauchbarer Wert für ε lässt sich folgendermaßen ermitteln:
 1. Man bilde als Lernparameter ε eine Potenz a^b :
 - a. Als Basis a wähle man den Betrag des größten Input-Wertes,
 - b. als Exponent b wähle man "-2" im Falle der linearen Aktivitätsfunktion (bzw. allg. bei Funktionen deren Ausgabe beliebig groß/klein sein kann), und "-1" im Falle der logistischen oder tanh- Aktivitätsfunktion (bzw. allg. bei Funktionen deren Ausgabe nie jenseits des Intervalls $[-1;1]$ liegt)
- Ein allgemeines Problem der Generalisierten Deltaregel beim Umgang mit Neuronalen Netzen mit Hiddenunits ist, dass es sich um ein Verfahren zur lokalen Optimierung handelt, so dass je nach gewählten Startgewichten unterschiedliche Ergebnisse resultieren können. Die errechneten Gewichte stellen i.d.R. ein lokales Fehler-Minimum dar. Ob es sich dabei jedoch um ein absolutes Minimum handelt lässt sich in vielen Fällen nicht feststellen. Lediglich in manchen Fällen hat die Fehlerfunktion ein einziges Minimum, sodass dieses potentielle Problem nicht besteht. Zwar stehen zahlreiche Variationen des Verfahrens (z.B. resilient-backpropagation), sowie alternative Algorithmen zur Verfügung (z.B. Simulated Annealing und Evolutionäre Algorithmen), die diese und weitere Mängel des Verfahrens zu umgehen helfen. Nichtsdestotrotz bleibt die globale Optimierung in der Mathematik oft ein unsicheres Unterfangen.
- Statt die Gewichte nach jedem Pattern zu verändern – man spricht von „incremental training“ – kann man sich auch entschließen, die Werte Δw_{ij} für jedes Pattern zu berechnen *ohne* die Gewichte gleich anzupassen. In diesem Fall werden die Gewichte erst am Ende jeder Epoche um die Summe aus allen für die einzelnen Patterns berechneten Δw_{ij} erhöht – man spricht in diesem Fall von „batch training“.

¹ Eine Epoche zeichnet sich dadurch aus, dass jeder Fall (d.h. die Daten jeder Person oder jedes Lernbeispiels) einmal präsentiert wurde

² Als Pattern bezeichnet man die Ausprägungen der verschiedenen Variablen eines Falls



Exkurs: Zeitreihenanalyse mit Neuronalen Netzen

Neuronale Netze lassen sich auch zur Analyse von Zeitreihen verwenden. Im Folgenden soll kurz darauf eingegangen werden, wie man sich Neuronale Netze in der Zeitreihenanalyse vorstellen kann. Das Grundprinzip ist schnell erklärt: Hat man zu vielen Zeitpunkten t eine Variable y erhoben, dann kann man versuchen jeden Wert $y(t)$ durch eine Reihe vergangener Werte $y(t-1), y(t-2), \dots$ vorherzusagen.

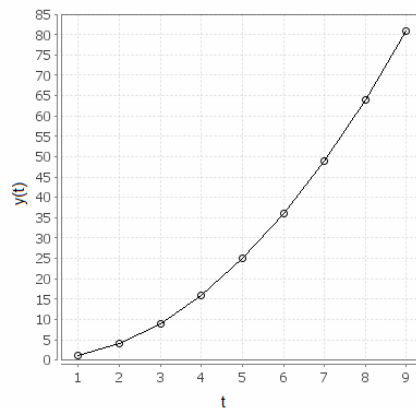


Abbildung 5: Eine simple Zeitreihe.

Angenommen z.B. man habe die in der Abbildung dargestellte Zahlenreihe $y = \{1, 4, 9, 16, 25, 36, 49, 64, 81, \dots\}$ und entschieße sich dazu, jeden Wert $y(t)$ durch die zwei vorangegangenen Werte $y(t-1)$ und $y(t-2)$ vorherzusagen, dann geht man wie folgt vor: Die ersten beiden Werte der Zeitreihe $y(t)$ (also 1 und 4) haben naturgemäß keine zwei vorangegangenen Werte, daher besteht der vorherzusagende Output (das sog. Kriterium) lediglich aus folgenden Werten:

- $y(t) = \{9, 16, 25, 36, 49, 64, 81, \dots\}$

Zur Vorhersage von $y(t)$ bildet man folgende zwei Input-Variablen (sog. Prädiktoren):

- $y(t-1) = \{4, 9, 16, 25, 36, 49, 64, \dots\}$
- $y(t-2) = \{1, 4, 9, 16, 25, 36, 49, \dots\}$

Diese Variablen speist man nun in ein Neuronales Netz mit folgenden Komponenten ein:

- zwei Inputunits für die beiden Variablen $y(t-1)$ und $y(t-2)$,
- eine Outputunit und
- evtl. noch eine Biasunit.
- Was die Anzahl oder das Vorhandensein der Hiddenunits anbelangt sind der Phantasie des Netzwerkarchitekten wie so oft keine Grenzen gesetzt.

Dieses Netz trainiert man nach der Deltaregel (als gewünschte Ausgabe der Outputunits zieht man $y(t)$ heran) ...Fertig ist ein einfaches (lineares oder nonlineares) *autoregressives Modell* bzw. ein prädiktives Neuronales Netz, das jeden Wert $y(t)$ auf Basis der zwei vorangegangenen Werte unserer Zeitreihe perfekt vorhersagt:

$$y(t) = -1 * y(t-2) + 2 * y(t-1) + 2$$

(Vorhersagefehler $RMSE=0.00$. Hätte man lediglich einen der beiden Prädiktoren verwendet, wäre die Vorhersage schlechter ausgefallen: Für den Prädiktor $y(t-1)$ läge der $RMSE$ bei 0.68, für den Prädiktor $y(t-2)$ sogar bei 1.69. Hätte man statt der Prädiktoren den Messzeitpunkt t als Prädiktor verwendet läge der $RMSE$ bei 3.46.)



2.1.2 Die Deltaregel und die klassische Statistik

Neuronale Netze *ohne* Hiddenunits entsprechen klassischen Regressionsmodellen: So entspricht z.B. die Formel zur Bestimmung der Ausgabe o einer Outputunit angesichts der Aktivitäten a einer oder mehrerer Inputunits im Falle einer linearen Aktivitätsfunktion, exakt der Regressionsgleichung einer linearen Regression der Kriteriumsvariable y auf einen oder mehrere Prädiktorvariablen x (vergleiche „ $o_i = \sum(a_j * w_{ij})$ “ und „ $y_i = \sum(x_j * b_{ij})$ “). Im Falle einer logistischen Aktivitätsfunktion hingegen entspricht die Ausgabe o der Regressionsgleichung einer logistischen Regression (vergleiche „ $o_i = \text{logistic}(\sum(a_j * w_{ij}))$ “¹ mit „ $y_i = \text{logistic}(\sum(x_j * b_{ij}))$ “²) – für eine sehr anschauliche Beschreibung der Zusammenhänge vgl. Sarle, 1994). Netze ohne Hiddenunits lassen sich insofern als *Generalisierte Lineare Modelle* auffassen. Das Generalisierte Linear Modell (GLM) der klassischen Statistik ist eine Generalisierung des Allgemeinen Linearen Modells (ALM) –welches sich auf lineare Zusammenhänge beschränkt²– und bietet einen einheitlichen Rahmen für eine Vielzahl linearer und nichtlinearer Regressionsansätze:

- 1) der Erwartungswert der Zielvariablen wird mit dem linearen Prädiktor $\eta = x' \beta$ durch eine (lineare oder nichtlineare) Responsefunktion/Aktivitätsfunktion h (bzw. eine Linkfunktion $g = h^{-1}$) verknüpft
- 2) Die Verteilung der Zielvariablen lässt sich in Form einer einparametrischen Exponentialfamilie darstellen (z.B. Normal-, Binomial-, Poisson-, oder Gamma-Verteilung).

Das GLM umfasst damit sowohl die *lineare Regression* (wo im Zuge der Maximum-Likelihood-Schätzung eine lineare Regressionsfunktion und normalverteilte Fehler angenommen werden³), als auch Regressionsmodelle für Daten die nicht normalverteilt und/oder nichtlinear zu beschreiben sind -wie z.B. die *logistische Regression* für ein binäres Kriterium (wo im Zuge der Maximum-Likelihood-Schätzung eine logistische Regressionsfunktion und binomialverteilte Fehler angenommen werden⁴). GLMs, deren Koeffizienten mittels Maximum-Likelihood-Schätzung bestimmt werden, stellen ein Kriterium in einem linearen oder nichtlinearen Zusammenhang mit einer gewichteten Linearkombination von Prädiktoren (wobei sich die Art des Zusammenhangs über die sog. "Link-Funktion" bestimmt). Neuronale Netze, die nach der Deltaregel trainiert werden, stellen eine Outputvariable in einem linearen oder nichtlinearen Zusammenhang mit einer gewichteten Linearkombination von Inputvariablen, nämlich dem Net-Input. Hier bestimmt sich die Art des Zusammenhangs über die sog. „Aktivitätsfunktion“. Die Koeffizienten in GLMs werden über Maximum-Likelihood-Schätzung errechnet, wobei hierfür eine bestimmte Verteilung der Fehler angenommen wird. Auch im Rahmen Neuronaler Netze lassen sich für beliebige Aktivitätsfunktionen und beliebige Fehlerverteilungen völlig äquivalente Schätzer ermitteln.

- Für herkömmliche lineare Modelle erbringt die Widrow-Hoff-Regel die Kleinste-Quadrate-Schätzer⁵ (die unter der Annahme eines normalverteilten Kriteriums auch Maximum-Likelihood-Schätzer sind). Für logistische Modelle ergeben sich die üblichen Maximum-Likelihood-Schätzer unter der impliziten Annahme eines binomialverteilten binären Kriteriums.
- Will man auch für nichtlineare Regressionsmodelle im Rahmen Neuronaler Netze die SAQ minimieren bzw. Kleinste-Quadrate-Schätzer erhalten, dann ist die Rumelhart-Regel die Lernregel der Wahl.

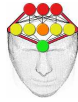
¹ Hier und im Folgenden steht $\text{logistic}(x)$ für $1/(1+e^{-x})$

² Wobei sich über nichtlineare Transformationen der Prädiktoren auch nichtlineare Zusammenhänge modellieren lassen, s.u.

³ Die Annahme normalverteilter Störgrößen trifft man allein deswegen, weil die Maximum-Likelihood-Schätzung nur unter dieser Annahme die Schätzer, welche die SAQ minimieren, auch als diejenigen Schätzer ausweist, welche den gegebenen Daten angesichts des Regressionsmodells mit höchster Wahrscheinlichkeit zugrunde liegen (vgl. Fahrmeir et al., 2007) – wie man sich auf Seite 6 des vorliegenden Artikels leicht vergewissern kann, entsprechen in diesem Fall die Formeln der Widrow-Hoff-Regel auch den Formeln nach Rumelhart et al. (1989)

⁴ Für eine binäre Variable normalverteilte Fehler anzunehmen gibt naturgemäß wenig Sinn (vgl. Backhaus et al., 2006)

⁵ Besonders interessante statistische Eigenschaften lassen sich für Kleinste-Quadrate-Schätzer unter den Annahmen nachweisen dass (a) der Erwartungswert der Störgrößen gleich Null ist, dass (b) die Störgrößen homoskedastisch und unkorreliert sind, und dass (c) die Designmatrix vollen Spaltenrang besitzt (für einen tieferen Einblick sei auf Fahrmeir et al., 2007, verwiesen)



Neuronale Netze mit Hiddenunits ("Multilayer-Perceptrons", MLPs) stellen eine besonders flexible Methode für *komplexe*¹ (nicht)lineare Regression eines Kriteriums auf beliebig viele Prädiktoren dar – wobei analog zu Neuronalen Netzen ohne Hiddenunits entweder die SAQ minimiert wird, oder die negative Likelihood der Schätzung für ausgewählte Fehlerverteilungen (Sarle, 1994). Im Gegensatz zu Neuronalen Netzen ohne Hiddenunits sind Neuronale Netze mit Hiddenunits dazu in der Lage, über eine Funktion der gewichteten Linearkombination der Funktionen von gewichteten Linearkombinationen der Prädiktoren beliebige Zusammenhänge anzunähern. Statt also nur diejenige Linearkombination der Prädiktoren zu ermitteln, für welche z.B. eine logistische Kurve optimal in den Daten liegt, ist ein MLP mit mehreren Hiddenunits im Falle nonlinearer Aktivitätsfunktionen in der Lage, eine Vielzahl nonlinearer Funktionen auf eine Weise zu addieren, die gewährleistet dass die gewichtete Linearkombination dieser nonlinearen Funktionen – eine i.d.R. sehr komplexe nonlineare Kurve – optimal zu den Daten passt. Da die Anzahl der Hidden-Units variieren kann (und die Regressionsgleichung bzw. -parameter von der Anzahl der Hiddenunits abhängen), lassen sich Neuronale Netze mit Hiddenunits als *nichtparametrische Regression* auffassen. Die Anzahl an Hiddenunits mit nonlinearer Aktivitätsfunktion bestimmt, wie komplex der angenäherte Zusammenhang maximal sein kann. *Prinzipiell lassen sich mit Neuronalen Netzen mit Hiddenunits und nonlinearen Aktivitätsfunktionen beliebige Zusammenhänge annähern – man spricht von solchen Netzen auch als „universelle Approximatoren“ (Hornik et al, 1989).*

Prinzipiell ist es auch im Rahmen der linearen Regression – und damit für Neuronale Netze ohne Hiddenunits – möglich, Interaktionen mehrerer Variablen oder Potenzen von Variablen in das Regressionsmodell mit aufzunehmen – und auf diese Weise komplexe nonlineare Zusammenhänge anzunähern - man nennt dieses Vorgehen auch *polynomiale Regression* (für ein beispielhaftes Regressionsmodell, das die Interaktion zweier Variablen enthält, vgl. Abb.4).

In diesem Fall wächst jedoch die Zahl möglicher Interaktionen und Potenzen – und damit sowohl der Rechenaufwand als auch die Anzahl der zu schätzenden Parameter – exponentiell mit der Anzahl der Inputvariablen. Überdies kann es bei Polynomen hoher Ordnung zu Problemen mit der numerischen Präzision kommen.

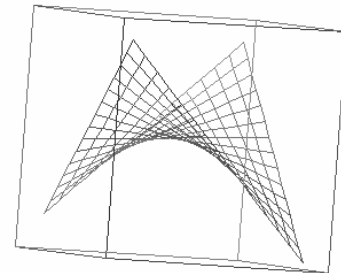


Abbildung 6: Dreidimensionaler Graph der Regressionsfunktion einer Interaktion zweier Prädiktoren: $y = x_1 \cdot x_2$

Neuronale Netze mit Hiddenunits und nonlinearen Aktivitätsfunktionen sind im Gegensatz zu polynomialen Regressionen *nonlinear in den Parametern*, was das Berechnen der Gewichte ebenfalls sehr rechenaufwändig werden lässt. Allerdings sind die Ergebnisse numerisch stabiler als bei der polynomialen Regression. Außerdem führt die Extrapolation zu anderen, häufig realistischeren, Vorhersagen (Polynome gehen für große Prädiktorbeträge z.B. stets gegen unendlich, während die Regressionsfunktionen von Neuronalen Netzen mit sigmoiden Aktivitätsfunktionen sich für große Prädiktorbeträge asymptotisch einem endlichen Grenzwert nähern) und es lassen sich leicht neue Inputvariablen einspeisen ohne dass dies einen exponentiellen Zuwachs zu schätzender Parameter veranlassen würde (vgl. Sarle, 1994).

Eine besonders interessante Beschaffenheit Neuronaler Netze liegt darin, dass die zahlreichen Interaktionen und Potenzen der Inputvariablen nicht als weitere Inputvariablen ins Modell eingespeist werden müssen, um alle relevanten Interaktionen und Nonlinearitäten zu modellieren. Vorzugeben ist lediglich die Anzahl an Hiddenunits und damit das gewünschte Ausmaß an modellierter Komplexität (z.B. lässt sich die Interaktion in Abb.4 mit drei Hiddenunits modellieren. Für das logische Entweder-Oder „XOR“ genügen bereits zwei Hiddenunits). Die Ermittlung der optimalen Kombination von Inputvariablen und die Berücksichtigung relevanter Wechselwirkungen wird vom Algorithmus selbst geleistet (vgl. Rumelhart et al., 1995 oder auch Rey&Wender, 2005).

¹ „Komplex“ im Sinne von „mehrere vernetzte Einheiten umfassend“. Das Adjektiv soll betonen, dass hier –im Gegensatz zu Neuronalen Netzen ohne Hiddenunits– mehrere nonlineare Funktionen der gewichteten Summe aller Prädiktoren zur Vorhersage je einer Kriteriumsvariable gewichtet aufsummiert werden.



Neuronale Netze mit Hiddenunits sind den gängigen Verfahren nichtparametrischer Regression ähnlich: Wie beispielsweise Klinko und Grassman (1998) herausgearbeitet haben weisen Neuronale Netze weitreichende Übereinstimmungen mit dem Verfahren *Projection Pursuit Regression* auf. Dieses Verfahren funktioniert auf eine sehr ähnliche Weise, jedoch werden statt vorgegebener Aktivitätsfunktionen verschiedene Basisfunktionen anhand der Daten geschätzt. Für einen tieferen Einblick sei verwiesen auf Klinge und Grassman (1998). Auch die Modellierung nichtlinearer Zusammenhänge über *Polynom-Spline-Regression* (vgl. Fahrmeir et al., 2007) lässt sich im Rahmen Neuronaler Netze realisieren (indem man als Aktivitätsfunktion der Hiddenunits z.B. eine lineare Funktion mit Schwelle wählt), vgl. Abbildung 7.

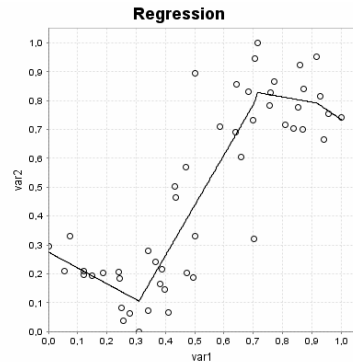


Abbildung 7: Neuronales Netz mit 4 Hiddenunits, deren Aktivitätsfunktion linear mit Schwelle ist

Eine Eigenart der komplexen nichtlinearen Regression ist es, dass sich die Stärke des Einflusses einzelner Variablen nicht mehr an einzelnen zugehörigen Koeffizienten ablesen lässt wie dies noch bei Regressionsmodellen im Rahmen des GLM der Fall war (ein interessanter Sonderfall ergibt sich bei lediglich einer Hiddenunit).

Der Einfluss einer Prädiktorvariable auf die Kriteriumsvariable zeigt sich nur mehr in Hinblick auf eine Vielzahl an Parametern, wobei die Bedeutung einzelner Parameter synergetisch hinter die Bedeutung jener Vielzahl zurücktritt: Einzelne Parameter sind häufig wenig aussagekräftig und austauschbar (sodass z.B. dieselbe Kurve auch bei anderer Parameterkonstellation emergieren könnte). Signifikanztestung bzw. die Berechnung von Vertrauensintervallen für einzelne Parameter erweist sich vor diesem Hintergrund häufig als sehr schwierig oder als unnötig. Nichtsdestotrotz lassen sich auch für Neuronale Netze mit Hiddenunits Maße für den Einfluss einzelner Prädiktorvariablen finden: Für einen Ansatz, der den Einfluss einer Inputunit über die Gesamtheit aller Gewichte dieser Unit zu statistisch relevanten Hiddenunits (vgl. White, 1989b) mithilfe eines (Chi²-verteilten) Wald-Tests auf statistische Signifikanz prüft sei verwiesen auf Anders (1996).

Die Wirkung einer bestimmten Variation einer Prädiktorvariable kann aber ohnehin –sofern bedeutsame Wechselwirkungen vorliegen– optimal nur im Kontext der Ausprägung der übrigen Prädiktorvariablen betrachtet werden.

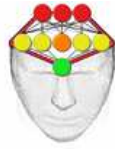
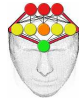
Neben dem Einfluss von Inputvariablen stellt sich beim Umgang mit MLPs auch die Fragen nach der Relevanz von bzw. einer möglichst optimalen Anzahl an Hiddenunits. Bei zu vielen Hiddenunits kann es zu sog. „overfitting“ kommen -von „overfitting“ spricht man, wenn zufällige unsystematische Schwankungen um einen zugrunde liegenden Zusammenhang aufgrund eines zu komplexen Modells für systematisch erklärt werden (worunter die Vorhersage neuer Daten leidet). Wählt man demgegenüber zu wenige Hiddenunits, können bedeutsame Nonlinearitäten u.U. nicht genügend angenähert werden.

In ihrem grundlegenden Artikel „Backpropagation: The Basic Theory“ erörtern Rumelhart et al. (1995) diverse interessante Blickpunkte auf das Phänomen Hidden-Unit:

1. “Sigmoidal hidden units can be viewed as approximations to linear threshold functions which divide the space into regions which can then be combined to approximate the desired function.
2. Hidden units may be viewed as a set of basis functions, linear combinations of which can be used to approximate the desired output function.
3. Sigmoidal hidden units can be viewed probabilistically as representing the probability that certain „hidden features“ are present in the input.
4. Layers of hidden units can be viewed as a mechanism for transforming stimuli from one representation to another from layer to layer until those stimuli which are functionally similar are near one another in hidden-unit space.”

Eine interessante Anwendung Neuronaler Netze liegt darin, zu errechnen, welche nichtlineare Transformation eines Prädiktors oder mehrerer Prädiktoren vorzunehmen ist, um die lineare Korrelation zwischen dem/den transformierten Prädiktoren und dem Kriterium zu optimieren (wobei dieses linearisierende Vorgehen nur in solchen Fällen zu empfehlen ist, wo augenscheinlich systematische Nonlinearität zwischen den Variablen vorliegt und auch angemessen, d.h. ohne sog. „overfitting“, modelliert wird).

Die lineare Korrelation zwischen transformierten Prädiktoren und Kriterium lässt sich wiederum interpretierbar mit Vertrauensintervallen belegen und über Signifikanztests von zufälligen Schwankungen um 0 abgrenzen.



Exkurs: Varianzanalyse im Rahmen des ALM

Über das "Allgemeine Lineare Modell" (ALM) lässt sich auch das weite Feld varianzanalytischer Verfahren in den begrifflichen Rahmen der multiplen linearen Regression integrieren. Ein zentrales Konzept für das Verständnis dieser Integration ist dasjenige der „Indikatorvariablen“: Um den Einfluss einer nominalskalierten Variable x mit regressionsanalytischen Methoden zu modellieren, erzeugt man zu jeder nominalskalierten Variablen mit p verschiedenen Ausprägungen $p-1$ Indikatorvariablen. Dies geschieht häufig auf eine der folgenden Weisen:

- **Dummycodierung:** Jede der $p-1$ Indikatorvariablen ist für jeweils eine Ausprägung der nominalskalierten Variable x gleich 1, und für alle anderen Ausprägungen 0. Beispiel: Angenommen man habe drei Personen, die drei verschiedenen Parteien angehören, dann ergäben zur Indikatorcodierung der Parteizugehörigkeit zwei Indikatorvariablen ($i1=\{1,0,0\}$; $i2=\{0,1,0\}$). An den Merkmalsausprägungen in diesen zwei Variablen kann man die Parteizugehörigkeit zu Partei1 ($i1=1,i2=0$), in Partei2 ($i1=0,i2=1$) oder Partei3 ($i1=0,i2=0$) ablesen.
- **Effektcodierung:** Jede der $p-1$ Indikatorvariablen wird so codiert, wie es für die Dummycodierung beschrieben ist. Danach wird noch diejenige Ausprägung, die auf allen Indikatorvariablen mit "0" codiert wurde (im o.g. Beispiel die Zugehörigkeit zu Partei3), durch "-1" ersetzt (in Bezug auf o.g. Beispiel ergäben sich also die zwei Indikatorvariablen $i1=\{1,0,-1\}$; $i2=\{0,1,-1\}$).

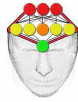
In das Regressionsmodell nimmt man als Prädiktoren diese $p-1$ Indikatorvariablen auf und nicht die ursprüngliche nominalskalierte Variable x . Für eine einfaktorielle Varianzanalyse (in welche die Daten von N Personen eingehen) erzeugt man für eine UV mit p Ausprägungen $p-1$ Indikatorvariablen via Effektcodierung (für weitere Verfahren -z.B. ANOVAS für zwei- und mehrfaktorielle Designs, sowie Designs mit Messwiederholung, ANCOVAs, t-Tests und Chi²-Tests,...- sei auf das Lehrbuch "Statistik" von Bortz (2005) verwiesen). Das Quadrat der multiplen Korrelation zwischen den $p-1$ Indikatorvariablen und der abhängigen Variablen entspricht dem Varianzanteil der abhängigen Variablen, der durch die $p-1$ Indikatorvariablen erklärt wird ($\text{Var}_{\text{real}}=R^2$). Der nicht erklärte Varianzanteil entspricht dem Fehlervarianzanteil ($\text{Var}_{\text{Fehler}}=1-R^2$). Der F-Wert ergibt sich wie folgt:

$$F = [R^2 * (N - p)] / [(1 - R^2) * (p - 1)]$$

Da sich im Rahmen des ALM varianzanalytische Fragestellungen mit Methoden der linearen Regression oder entsprechender Neuronaler Netze angehen lassen, ist zu überlegen, ob sich die die Leistungsfähigkeit der ANOVA durch Neuronale Netze (mit nichtlinearen Aktivitätsfunktionen) noch übertreffen ließe. Dieses Vorhaben stößt durch die Natur der Indikatorvariablen schnell an Grenzen:

- Hat eine Indikatorvariable nur zwei Stufen, ist eine Gerade bereits völlig ausreichend (da man ja nur zwei Punkte hat, die man verbinden will)
- Hat man mehrere Indikatorvariablen, die jeweils nur über die Ausprägungen -1,0 und 1 verfügen (wobei die "-1" in allen Variablen an den gleichen Stellen steht), hat man scheinbar ein ähnliches Problem (auch in diesem Fall stellt eine (Hyper-) Ebene bereits die beste Anpassung dar)

Das volle Potential Neuronaler Netze zum flexiblen Annähern nonlinearer Zusammenhänge könnte man u.U. nutzen, um den Einfluss von Kovariaten aus einer Abhängigen Variablen herauszurechnen.



2.1.3 Deltaregel-Zahlenbeispiel

Wir erstellen ein Neuronales Netz *ohne Hiddenunits* mit einer Inputunit, einer Biasunit sowie einer Outputunit und speisen aus Tabelle 1 (siehe Anhang) die Variable „UV“ als Prädiktorvariable und die Variable „AV“ als Kriteriumsvariable ein. Wir wählen eine lineare Aktivitätsfunktion und sind daran interessiert, für normalverteilte Fehler die Summe der quadrierten Abweichungen zwischen den vorhergesagten Werten und den tatsächlichen Ausprägungen von AV zu minimieren.

Hierfür verändern wir die Gewichte w gemäß dem angegebenen Algorithmus viele Epochen lang für jedes Pattern um

$$\Delta w_{ij} = \epsilon \cdot \delta_i \cdot a_j \cdot f_{\text{akt}}'(\text{net}_i).$$

Das Gewicht zwischen Inputunit und Outputunit konvergiert gegen 0.905, das Gewicht zwischen Biasunit und Outputunit konvergiert gegen 0.004. Die Ausgabe y der Outputunit lässt sich also errechnen als

$$y = 0.905 \cdot x + 0.004$$

wobei diese Gleichung (vgl. Abbildung 8) exakt der Regressionsgleichung entspricht, die auch eine herkömmlich gerechnete lineare Regression von Variable AV auf Variable UV zum Ergebnis hat ($R=0.900$; $R^2=0.809$; $RMSE=0.183$)

Ersetzt man im gerade beschriebenen Prozedere die lineare Aktivitätsfunktion durch eine logistische Aktivitätsfunktion resultieren andere Gewichte. Die Ausgabe y der Outputunit lässt sich in diesem Fall errechnen als

$$y = \text{logistic}(5.293 \cdot x - 2.918)$$

wobei auch für diese logistische Kurve (vgl. Abbildung 9) gilt, dass die Summe der quadrierten Abweichungen zwischen vorhergesagten Werten und tatsächlichen Ausprägungen von Variable AV näherungsweise minimal ist ($R = 0.921$; $R^2 = 0.847$; $RMSE = 0.167$).

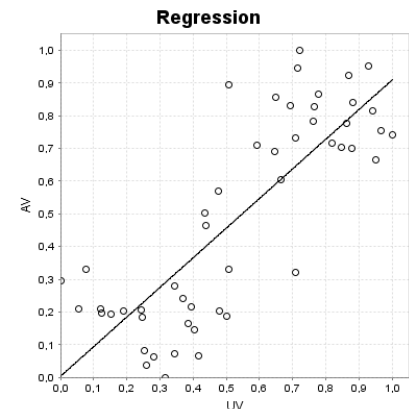


Abbildung 8: Regressionsfunktion eines Neuronales Netzes ohne Hiddenunits, mit linearer Aktivitätsfunktion

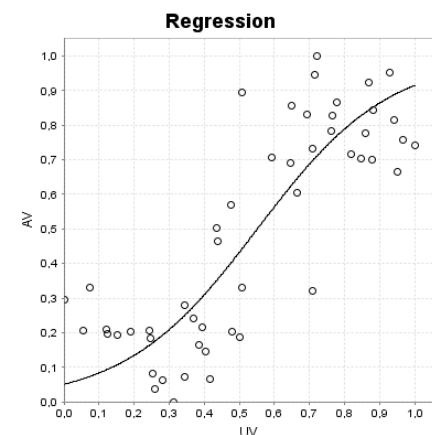
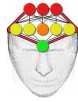


Abbildung 9: Regressionsfunktion eines Neuronales Netzes ohne Hiddenunits, mit logistischer Aktivitätsfunktion



Fügt man in die eben vorgestellte Architektur noch Hiddenunits mit nonlinearer Aktivitätsfunktion ein, lassen sich wie angesprochen beliebig komplexe Zusammenhänge annähern. Wir wählen vorerst vier Hiddenunits und für jede Hiddenunit die logistische Aktivitätsfunktion.

Da wir weiterhin daran interessiert sind, für normalverteilte Fehler die Summe der quadrierten Abweichungen zwischen den vorhergesagten Werten und den tatsächlichen Ausprägungen von AV zu minimieren, verändern wir die Gewichte w gemäß dem angegebenen Algorithmus viele Epochen lang für jedes Pattern um $\Delta w_{ij} = \varepsilon \cdot \delta_i \cdot a_j \cdot f'_{akt}(\text{net}_{ij})$ (für Gewichte zwischen einer Hiddenunit j und Outputunit i) bzw. um $\Delta w_{lk} = \varepsilon \cdot \delta_k \cdot a_l \cdot f'_{akt}(\text{net}_{lk})$ (für Gewichte zwischen einer Inputunit l und einer Hiddenunit k).

Das Ergebnis ist folgende Gleichung zur Berechnung der Ausgabe y der Outputunit

$$y = 1.935 \cdot \text{logistic}(7.845 \cdot x - 3.768) + \\ -1.839 \cdot \text{logistic}(2.528 \cdot x - 0.409) + \\ 0.461 \cdot \text{logistic}(0.067 \cdot x + 0.320) + \\ -1.464 \cdot \text{logistic}(2.127 \cdot x + 0.029) + 1.523$$

wobei auch für diese komplexe Kurve (vgl. Abbildung 10) gilt, dass die Summe der quadrierten Abweichungen zwischen vorhergesagten Werten und tatsächlichen Ausprägungen von Variable AV näherungsweise minimal ist ($R = 0.946$; $R^2 = 0.895$; $RMSE = 0.139$)

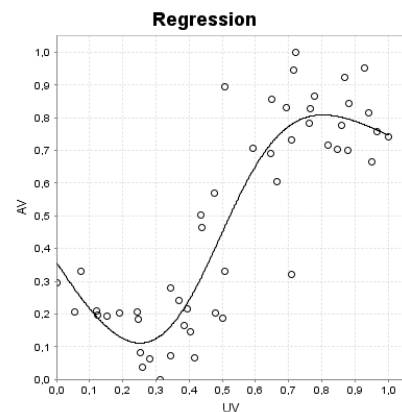


Abbildung 10: Regressionsfunktion eines Neuronalen Netzes mit 4 Hiddenunits deren Aktivitätsfunktion

Wählt man statt vier Hiddenunits lediglich eine Hiddenunit, ergeben sich Ergebnisse wie sie in der Abbildung (rechts) zu sehen sind.

$$y = 0.637 \cdot \text{logistic}(15.983 \cdot x - 8.123) + 0.156$$

wobei auch für diese Kurve (vgl. Abbildung 11) gilt, dass die Summe der quadrierten Abweichungen zwischen vorhergesagten Werten und tatsächlichen Ausprägungen von Variable AV näherungsweise minimal ist ($R = 0.941$; $R^2 = 0.886$; $RMSE = 0.144$)

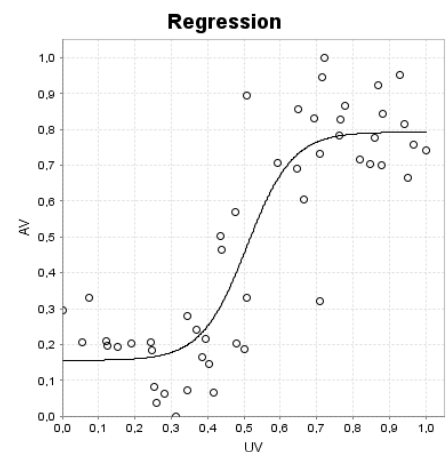


Abbildung 11: Regressionsfunktion eines Neuronalen Netzes mit einer Hiddenunit, deren Aktivitätsfunktion logistisch ist



2.2 Hebb'sche Regel

Die Hebb'sche Regel stellt eine Lernregel für Neuronale Netze ohne Hiddenunits dar. Sie fordert in ihrer einfachsten Form lediglich die Präsentation von Inputpatterns. Da keinerlei Kriteriumsvariable oder gewünschte Aktivität der Outputunits vorzugeben ist, spricht man von „unsupervised learning“ (ersetzt man in der Formel jedoch z.B. die Aktivität der empfangenden Unit durch die gewünschte Aktivität der empfangenden Unit, lässt sich die Hebb'sche Regel auch als „supervised training“ realisieren).

Die Hebb'sche Regel fordert zwischen zwei Units i und j eine Gewichtsveränderung um folgenden Wert Δw_{ij} :

$$\Delta w_{ij} = \epsilon * a_i * a_j$$

Es existiert eine Verallgemeinerung der Hebb'schen Regel – die sog. „Sanger-Regel“ oder auch einfach „Generalisierte Hebb'sche Regel“ – die bei linearen Aktivitätsfunktionen und standardisiert eingespeisten Inputdaten zu Gewichten führt, welche man als Elemente der Eigenvektor-Matrix V der Korrelationsmatrix R der Inputdaten interpretieren kann (Sanger, 1989a/b).

Die Formel für Δw_{ij} lautet für die Sanger-Regel wie folgt:

$$\Delta w_{ij} = \epsilon * a_j - \sum a_k * w_{kj}$$

(wobei die Summe von der ersten Outputunit bis zur i -ten Outputunit geht)

2.2.1 Algorithmus

Setze die Gewichte des Netzes randomisiert auf Werte nahe 0

FOR EACH (Epoche) **DO** {

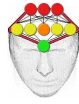
FOR EACH (Pattern) **DO** {

- Setze die Ausgabe a_j jeder Inputunit j auf den Wert der entsprechenden Prädiktorvariable x_j
- Berechne die Ausgabe a_i jeder Outputunit i
- Verändere jedes Gewicht zwischen einer Inputunit j und einer Outputunit i um den Wert Δw_{ij}

}}

Anmerkungen zum Algorithmus:

- Wie schon bei den übrigen Algorithmen lässt sich auch die Hebb'sche Regel als „batch training“ und als „incremental training“ realisieren.
- Es sei noch erwähnt, dass die angegebene Formel der ursprünglichen Formulierung von Donald Hebb („what fires together wires together“) nur in dem Fall entspricht, dass als Aktivitäten der Neuronen lediglich positive Werte zugelassen werden – wie es z.B. unter Verwendung der binären oder der logistischen Aktivitätsfunktion der Fall ist. Bezieht man auch negative Aktivitätswerte in die Rechnung mit ein entfernt man sich von Hebb's ursprünglicher Formulierung, was der Nützlichkeit der Regel jedoch keinen Abbruch tut.



2.2.2 Die Hebb'sche Regel und die klassische Statistik

- **Hauptkomponenten- und Hauptachsenanalyse (Multivariate Zusammenhänge)**

Die Hauptkomponentenanalyse (*Principal Component Analysis*, PCA) ist ein statistisches Verfahren, das den Versuch unternimmt, eine Vielzahl korrelierter Variablen durch möglichst wenige unkorrelierte Faktoren hinreichend gut abzubilden. Es wird davon ausgegangen, dass die gesamte Varianz der Variablen vollständig aufgeklärt wäre, wenn man nur ebenso viele Faktoren wie Variablen in seine Analyse mit einbeziehen würde. In der Praxis nimmt man jedoch geringe Verluste an Varianzaufklärung zugunsten einer geringeren Faktorzahl in Kauf. Praktisch zeigt sich diese Annahme darin dass – im Zuge der weiter unten angeführten Kommunalitätsschätzung (unter der Kommunalität einer Variable versteht man die durch die Faktoren aufklärbare Varianz) – die Hauptdiagonale der Korrelationsmatrix R aller Variablen auf 1 gesetzt wird.

Auf rein deskriptiver Ebene hat die Annahme, eine Variable korreliere mit sich selbst zu 1, ihre Berechtigung. Sobald man jedoch inferenzstatistische Aussagen treffen möchte, die über die aktuelle Stichprobe hinausgehen, ist eben jene Annahme mit größerer Vorsicht zu genießen, da es durchaus oft absehbar ist, dass zwei Erhebungen der gleichen Variable nicht zu 1 miteinander korrelieren (man denke zum Vergleich etwa an Retest-Reliabilitäten kleiner 1 in der Testdiagnostik).

Die Hauptachsenanalyse (*Factor Analysis*, FA) trägt genau dieser Problematik Rechnung, indem sie den Variablen unsystematische Varianzanteile wie spezifische und Fehlervarianz zugesteht, insofern als sie – im Gegensatz zur PCA – nur die Kovarianz der Variablen zur Analyse heranzieht. Die FA ist damit besonders geeignet in mit Messfehlern behafteten Daten nach zugrundeliegenden linearen Strukturen zu suchen.

Exkurs: Die Philosophie hinter der FA ähnelt derjenigen der klassischen Testtheorie insofern, als hier wie dort davon ausgegangen wird, dass konkrete Itemausprägungen von wahren Werten (Faktorwerten) und einer Messfehlerkomponente abhängig sind.

Fazit:

Die Hauptkomponentenanalyse stellt ein Hilfsmittel dazu dar, Datensätze auf Stichprobenebene durch wenige Faktoren möglichst gut zu beschreiben, während die Hauptachsenanalyse zu Rate gezogen wird wo es darum geht, Datensätze über die Stichprobe hinausgehend durch wenige Faktoren möglichst gut zu erklären

Bedenkenswert:

Die Lösungen einer Faktorenanalyse verbessern sich mit der Größe der Stichprobe, mit der Linearität zwischen den Variablen (wobei Multikollinearität zwischen den Variablen in manchen Varianten der Faktorenanalyse zu Problemen führen kann wo Matrizen invertiert werden müssen) und mit der multivariaten Normalverteilung der Items.

Die Kommunalitätsschätzungen (s.u.) der Hauptachsenanalyse werden überdies mit der Anzahl aufgenommener Items genauer (da das Verhältnis von zu schätzenden Diagonalelementen der Korrelationsmatrix R zu den restlichen –der Schätzung zugrunde liegenden- Elementen von R kleiner wird).

Aus der Korrelationsmatrix R der Daten lässt sich mathematisch das Fundamentaltheorem der Faktorenanalyse für unkorrelierte Faktoren ableiten:

$$R=AA'$$



Nun müssen die Kommunalitäten der Variablen geschätzt werden. Für die PCA sind sie per Definition 1. Bei der FA behilft man sich über ein iteratives Schätzverfahren, das die Diagonale der Korrelationsmatrix R mit den quadrierten multiplen Korrelationen (SMCs) der Items füllt. Danach wird nach gewissen Kriterien die Anzahl bedeutsamer Faktoren bestimmt, ausgehend von dieser Faktorenzahl die Kommunalitäten errechnet und diese wiederum in die Diagonale der Korrelationsmatrix einträgt. Daraufhin wird wieder die Faktorenzahl bestimmt usw., bis die Kommunalitäten gegen einen stabilen Wert konvergiert sind. Andernfalls wird das Schätzverfahren mit anderen Kriterien für die Faktorenzahlauswahl wiederholt. Je mehr Variablen man hat, desto besser werden die Schätzungen bzw. desto unwichtiger werden die Startwerte des iterativen Schätzverfahrens.

Mit der (evtl. durch das Schätzverfahren erhaltenen modifizierten) Korrelationsmatrix R wird nun weitergerechnet.

Zuerst werden die Eigenwerte (in der Diagonalmatrix L) und Eigenvektoren (in der Matrix V) von R errechnet. Es gilt:

$$V'V=I$$
$$L=V'RV$$

Daraus folgt für die Faktorladungsmatrix A der Korrelationen einer jeden Variable (in den Zeilen) mit allen Faktoren (in den Spalten):

$$A=V\sqrt{L}$$

I.d.R. werden die interessierenden Spalten von A (d.h. die Ladungen aller Items zu den interessierenden Faktoren) noch einem Rotationsverfahren nach Wahl (zur Varianzmaximierung der Faktoren o.ä.) unterzogen.

Die Eigenwert (EW) eines Faktors ist $\sum a^2$ vertikal
Die Kommunalität (h^2) eines Items ist $\sum a^2$ horizontal

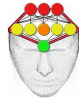
Die durch einen Faktor aufgeklärte Varianz beträgt $EW/\text{Variablenanzahl}$
Die durch einen Faktor aufgeklärte Kovarianz beträgt $EW/\sum (EW)$.

Die Sanger-Regel führt bei linearen Aktivitätsfunktionen und standardisiert eingespeisten Inputdaten zu Gewichten, welche man als Elemente der Eigenvektor-Matrix V der Korrelationsmatrix R der Inputdaten interpretieren kann (Sanger, 1989a/b).

Die Gewichte entsprechen damit dem Ergebnis einer Hauptkomponentenanalyse (PCA).

Die angeführte Formel von Sanger findet auch bei nonlinearen Aktivitätsfunktionen Verwendung wo sie bei normalverteiltem Input zu unkorrelierten Outputaktivitäten mit maximaler Varianz führt (vgl. Sanger, 1989b). Für weitere Einblicke in nonlineare PCA über „unsupervised training“ sei auf die Arbeiten der Forschergruppe um Oja verwiesen (vgl. Karhunen, 1994).

Am Rande sei auch erwähnt, dass es Ansätze gibt, die vergleichbare Ergebnisse mit der Generalisierten Deltaregel (über sog. „Autoassoziative Netze“) hervorbringen, indem dieselben Variablen als Prädiktoren und als Kriterien eingespeist werden (vgl. „Self-Supervised-Backpropagation“ z.B. bei Sanger, 1989a; oder „Nonlinear PCA“ bei Kramer, 1991).



2.2.3 Hebb'sche Regel Zahlenbeispiel

Wir erstellen zuerst ein Neuronales Netz *ohne Hiddenunits* mit zwei Inputunits sowie einer Outputunit und speisen aus Tabelle 1 die Variablen UV und AV als Prädiktorvariablen ein. Wir wählen eine lineare Aktivitätsfunktion und sind am ersten Eigenvektor der Korrelationsmatrix R interessiert.

Hierfür verändern wir die Gewichte w gemäß dem angegebenen Algorithmus viele Epochen lang für jedes Pattern um

$$\Delta w_{ij} = \epsilon \cdot a_j - \sum_k a_k \cdot w_{kj}$$

(wobei die Summe von der ersten Outputunit bis zur i -ten Outputunit geht)

Die Gewichte beider Variablen konvergieren gegen 0.707, sodass sich die Aktivität der Outputunit (im Folgenden auch bezeichnet als Eigenvektor) errechnen lässt als

$$\text{Eigenvektor} = 0.707 \cdot x1[UV] + 0.707 \cdot x2[AV]$$

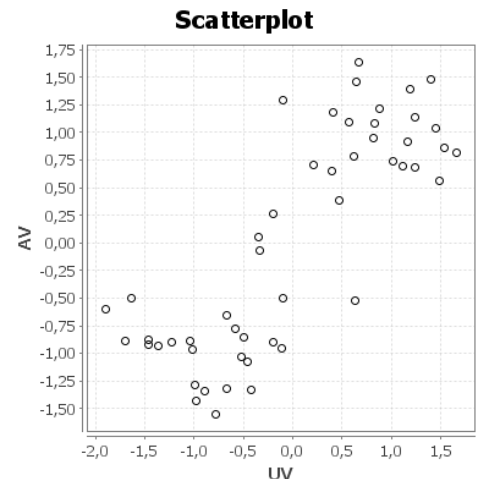


Abbildung 12: Punktwolke.

Über Matrizenmultiplikation ergeben sich aus dem dargestellten Eigenvektor die folgenden Ergebnisse einer herkömmlichen Hauptkomponentenanalyse (vgl. zu den Formeln Bortz, 2005):

Eigenvektormatrix V

| |
|-------|
| 0.707 |
| 0.707 |

V'V = I

| |
|-------|
| 1.000 |
|-------|

Quadratische Iteminterkorrelationsmatrix R

| |
|---------------|
| 1.000 0.809 |
| 0.809 1.000 |

Diagonalmatrix der Eigenwerte L = V'RV

| |
|-------|
| 1.809 |
|-------|

Faktorladungsmatrix A = V (L^{1/2})

| |
|-------|
| 0.951 |
| 0.951 |

R_reproduziert1 = AA'

| |
|---------------|
| 0.905 0.905 |
| 0.905 0.905 |

R_residual1 = R - R_reproduziert1

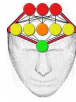
| |
|----------------|
| 0.095 -0.095 |
| -0.095 0.095 |

R_reproduziert2 = VLV'

| |
|---------------|
| 0.905 0.905 |
| 0.905 0.905 |

R_residual2 = R - R_reproduziert2

| |
|----------------|
| 0.095 -0.095 |
| -0.095 0.095 |



2.3 Kohonen-Algorithmus

Netze, die nach einer Variante des Kohonen-Algorithmus trainiert werden, bezeichnet man auch als Kohonennetze oder auch "Self Organizing Maps" (SOMs).

Die wesentliche Besonderheit an diesem Algorithmus stellt die Verwendung einer sog. „Nachbarschaftsfunktion“ $neigh(x,y)$ dar, welche für zwei Output-Units x und y einen Wert ausgibt, der i.d.R. umso kleiner ist, je weiter die beiden Units voneinander entfernt sind. Durch diese Funktion kommt also – anders als bei den bisher vorgestellten Algorithmen – erstmals auch die Distanz zwischen Outputunits ins Spiel bzw. die räumliche Anordnung der Outputunits.

Kohonennetze sind in der Lage, ohne vorgegebenen Output niedrigdimensionale topologische Karten eines u.U. hochdimensionalen Inputraums zu erstellen. Sie sind in besonderem Ausmaß geeignet für die Repräsentation und Klassifikation von Inputmustern, da ähnliche Inputmuster i.d.R. nahe bei einander liegende (im Extremfall identische) Outputunits aktivieren. SOMs finden daher u.a. Verwendung bei der Mustererkennung, wie z.B. Fingerabdruck-, Gesichter-, oder Sprach-Erkennung. Da man keine gewünschte Ausgabe der empfangenden Units vorgeben muss, handelt es sich um sogenanntes „unsupervised training“. Die Outputschicht eines Kohonennetzes (in folgender Abbildung die Schicht der zweidimensional angeordneten roten Units) wird auch "feature map" genannt.

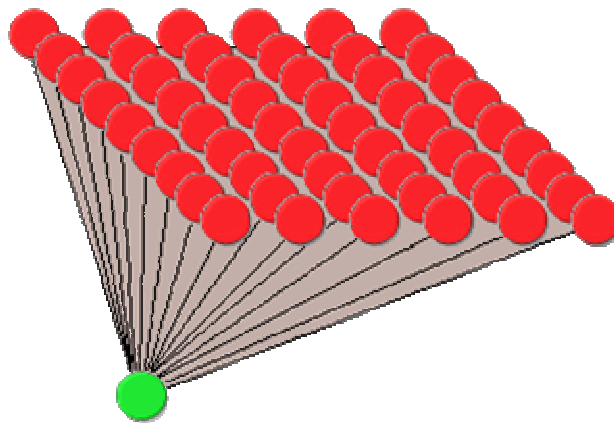


Abbildung 13:
Kohonennetz

2.3.1 Algorithmus

1. Setze die Gewichte des Netzes randomisiert auf Werte nahe 0 und normalisiere die Inputvariablen
2. **FOR EACH** (Epochen) **DO** { **FOR EACH** (Pattern) **DO**{

- Bestimme für jede Outputunit i die euklidische Distanz $euklid_i$ zum Inputpattern:

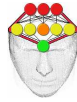
$$euklid_i = \sqrt{\sum (w_{ij} - x_j)^2}$$

- bestimme die Gewinnerunit z , für die $euklid_i$ am kleinsten ausfiel
- modifiziere die Gewichte jeder Outputunit i um folgenden Wert Δw_{ij} :

$$\Delta w_{ij} = \varepsilon * neigh(i,z) * (x_j - w_{ij})$$

- verringere Lernparameter ε und der Radius der Nachbarschaftsfunktion $neigh(x,y)$

}}



Anmerkungen zum Algorithmus:

- Da die Gewichte der Outputunits umso stärker verändert werden, je näher eine Outputunit der Gewinner-Unit ist (gegen Ende des Trainings – wenn der Radius der Nachbarschaftsfunktion gegen 0 geht – werden sogar nur mehr die Gewichte der Gewinnerunit verändert), spricht man bei diesem Lernprozess auch von "competitive learning" bzw. von "Wettbewerbslernen".
- Will man die Aktivität einer Unit der "feature map" mit einem sinnvollen Output assoziieren, bietet es sich an, als eine Erweiterung von SOMs z.B. "Counterpropagation"-Netze zu verwenden (vgl. Vracko, 2005).
- Während die Nachbarschaftsfunktion bei Kohonennetzen u.a. auch vorgibt welches Neuron welche Nachbarn hat (verbindet man jede Unit mit der ihr nächstgelegenen ergibt sich z.B. eine zweidimensionale Gitterstruktur), verzichtet man beim sogenannten "Neuronalen Gas" auf die Vorgabe eines solchen Gitters. Stattdessen wird die Nachbarschaft zur Gewinnerunit anhand der Ähnlichkeit der Reaktion auf den Input ermittelt.

2.3.2 Der Kohonen-Algorithmus und die klassische Statistik

„Clusteranalyse“ ist ein Sammelbegriff, welches Verfahren umfasst, die Objekte mit diversen Merkmalen nach ihrer Ähnlichkeit gruppieren (zu sogenannten Clustern). Unterschiede innerhalb eines Clusters sollen möglichst gering und Unterschiede zwischen den Clustern möglichst groß ausfallen (vgl. Bortz, 2005).

Zur Bestimmung der Ähnlichkeit zwischen Objekten mit intervallskalierten Merkmalen (zu Ähnlichkeitsmaßen für nominal-, ordinal- oder gemischt-skalierte Merkmale vgl. Bortz, 2005) wird meist die euklidische Metrik herangezogen:

$$d_{ij} = [\sum (x_{ij} - x_{ij})^2]^{1/2}$$

Im Falle korrelierter Merkmale kann es sich anbieten, stattdessen mit der sog. Mahalanobis-Distanz ein euklidisches Distanzmaß heranzuziehen, das um Korrelationen zwischen den Merkmalen bereinigt ist:

$$d_{ij} = [\sum \sum c^{jk} (x_{ij} - x_{ij}) (x_{ik} - x_{ik})]^{1/2}$$

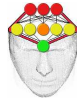
wobei c = Element jk aus der Inversen der Varianz-Kovarianz-Matrix der p Variablen.

Wenn weniger die einzelnen Abweichungen/Übereinstimmungen zwischen den Merkmalen von Objekten interessieren, sondern eher ähnliche Profilverläufe, kann es auch sinnvoll sein die Objektähnlichkeiten über Produkt-Moment-Korrelationen zu bestimmen.

Auf der Basis eines der oben genannten Ähnlichkeitsmaße werden die Objekte im Rahmen einer Clusteranalyse also zu Clustern gruppiert.

Man unterscheidet hierbei

- a) hierarchische Verfahren wie *agglomerative Algorithmen*, die mit ebenso vielen Clustern wie Objekten beginnt und in jedem Schritt die jeweils ähnlichsten Cluster/Objekte zu einem Cluster fusioniert (vgl. single-linkage-, complete-linkage-, average-linkage-Clustering, Ward-Clustering, ...), oder *divisive Algorithmen*, die mit einem Cluster beginnen das in zunehmend viele Teilcluster zerlegt wird



- b) nicht-hierarchische Verfahren, bei denen die Zugehörigkeit der Objekte zu einer vorgegebenen Anzahl an Clustern schrittweise optimiert wird (vgl. k-means-Clustering).

Wählt man für ein Kohonennetz die Nachbarschaftsfunktion so, dass ein Wert 1 nur für die Gewinnerunit zurückgegeben wird, und sonst ein Wert von 0, oder entfernt man die Nachbarschaftsfunktion $neigh(i,z)$ in der Formel für Δw_{ij} , dann ergibt sich exakt die Formel für k-means-Clustering:

$$\Delta w_{ij} = \varepsilon * (x_j - w_{ij})$$

Auch bei k-means-Clusteranalyse werden k Cluster/Outputunits angenommen, wobei für jede Outputunit i die Gewichte zu den Merkmalen/Inputunits iterativ so verändert werden, dass die Gewichte dem Mittelwert derjenigen Inputpatterns ähnlicher werden, die der Outputunit i ähnlicher sind als jeder anderen Outputunit.

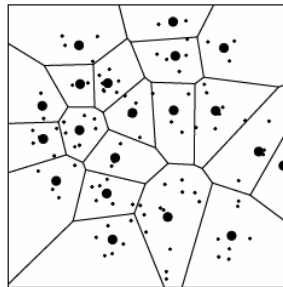
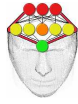


Abbildung 14: Cluster (Punkte) und ihre Einzugsbereiche

Der Vorteil von Kohonennetzen im Vergleich zur k-means-Clusteranalyse liegt in der Verwendung der Nachbarschaftsfunktion, bzw. in dem Resultat, dass gegen Ende des Trainings numerisch ähnliche Inputpatterns entweder identischen oder räumlich nahe beieinander liegenden Outputunits zugewiesen werden. Auf diese Weise ist sichergestellt, dass z.B. bei neuartigen oder verrauschten Inputs auch im Falle einer Fehlklassifikation zumindest eine der richtigen Lösung ähnliche Klassifikation vorgenommen wird.



2.3.3 Kohonen-Algorithmus-Zahlenbeispiel

Wir erstellen zuerst ein Neuronales Netz *ohne Hiddenunits* mit zwei Inputunits sowie 25 Outputunits (die zu einem Quadrat aus 5x5 Units angeordnet sind) und speisen aus Tabelle 1 die Variablen UV und AV als Prädiktorvariablen ein. Wir wählen eine lineare Aktivitätsfunktion und sind daran interessiert, die 50 Inputpatterns den 25 Outputunits zuzuweisen und zwar derart, dass numerisch ähnliche Inputpatterns derselben oder einer räumlich nahe gelegenen Outputunit zugewiesen werden. Hierfür verändern wir die Gewichte w gemäß dem angegebenen Algorithmus mehrere Epochen lang für jedes Pattern um

$$\Delta w_{ij} = \varepsilon * neigh(i,z) * (x_j - w_{ij})$$

Folgendes waren die Gewichte der 25 Outputunits zu den beiden Inputunits UV und AV:

Für UV:

0.383 | 0.239 | 0.185 | 0.068 | 0.000 | 0.460 | 0.359 | 0.300 | 0.198 | 0.164 | 0.722 | 0.665 | 0.578 | 0.520 | 0.427 | 0.900
| 0.917 | 0.797 | 0.669 | 0.609 | 1.000 | 0.986 | 0.779 | 0.681 | 0.514

Für AV:

0.004 | 0.029 | 0.149 | 0.151 | 0.271 | 0.133 | 0.114 | 0.216 | 0.213 | 0.335 | 0.373 | 0.454 | 0.547 | 0.592 | 0.465 | 0.724
| 0.803 | 0.816 | 0.726 | 0.659 | 0.717 | 0.863 | 0.947 | 0.858 | 0.932

Jedes der 50 Pattern (bunte Punkte in Abbildung 15) lässt sich nun derjenigen Outputunit (in Abbildung 15 dargestellt als Knotenpunkt im Gitternetz) zuweisen, der es am nächsten liegt bzw. am ähnlichsten ist (wobei durchaus mehrere Patterns derselben Outputunit zugeordnet werden kann).

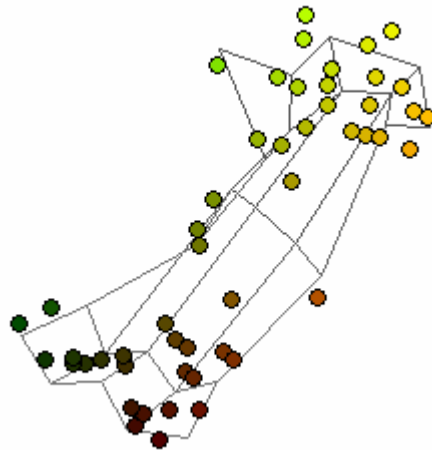
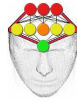
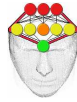


Abbildung 15: Kohonennetz in einer Punktwolke (Die Knotenpunkte des Gitters repräsentieren Outputunits).

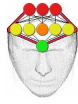


Literaturverzeichnis

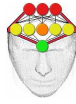
- Anders, U., 1996. Statistical model building for neural networks. Sixth AFIR Colloquium, 963–975.
- Backhaus, K., Erichson, B., Plinke, W. & Weiber, R. (2006). *Multivariate Analysemethoden. Eine anwendungsorientierte Einführung* (11. Auflage). Berlin & Heidelberg: Springer.
- Bortz, J. (2005). *Statistik für Human- und Sozialwissenschaftler* (6. Auflage). Berlin: Springer.
- Eysenck, M.W., Keane, M.T. (2005). *Cognitive Psychology*. Psychology Press: East Sussex.
- Fahrmeir, L., Kneib, T. & Lang, S. (2007). *Regression. Modelle, Methoden und Anwendungen*. Berlin & Heidelberg: Springer.
- Fischer, A., Greiff, S., & Funke, J. (2012). The Process of Solving Complex Problems. In *Journal of Problem Solving*, 4(1), 19-41.
- Fischer, A. (2010). *Instruktionelle Erklärungen zur Verbesserung der Verständlichkeit von Statistiksoftware*. Universität Trier: Diplomarbeit.
- Fox, J. (2002). *Bootstrapping Regression Models. Appendix to An R and S-PLUS Companion to Applied Regression*. In: <http://cran.r-project.org/doc/contrib/Fox-Companion/appendix-bootstrapping.pdf> Zugriff: 7.11.10
- Funke, J. (2003). *Problemlösendes Denken*.
- Heaton, J. (2008). Introduction to neural networks for Java.
- Hornik, K., Stinchcombe, M. & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2, 359-366.
- Karhunen, J., "Optimization Criteria and Nonlinear PCA Neural Networks", Proc. Of the 1994 IEEE International Conference on Neural Networks (a part of World Congress on Computational Intelligence), Orlando, Florida, June 26 - July 2, 1994, vol. II, pp.1241-1246. (Invited paper in the special session on Nonlinear PCA Neural Networks.)
- Klinke, S., Grassmann, J. (1998): Projection Pursuit Regression and Neural Network, DP 9817, SFB 373, Humboldt-University of Berlin. In: <http://edoc.hu-berlin.de/series/sfb-373-papers/1998-17/PDF/17.pdf> Zugriff: 22.11.2010
- Kramer, M. A., 1991: Nonlinear principal component analysis using autoassociative neural networks. *AIChE J.*, 37, 233–243.
- Lämmel, U., Cleve, J. (2008). *Künstliche Intelligenz*.
- Macho, S. (2002). Kognitive Modellierung mit Neuronalen Netzen.
- Papadopoulos, G., Edwards, P.J., Murray, A.F. (2000). *Confidence Estimation Methods for Neural Networks: A Practical Comparison*. In: <http://www.dice.ucl.ac.be/Proceedings/esann/esannpdf/es2000-27.pdf> Zugriff: 7.11.10
- Rumelhart, D.E., Durbin, R., Golden, R. & Chauvin, Y. (1995). Backpropagation: The Basic Theory. In: Y.Chauvin, D.E.Rumelhart (Hrsg.), *Backpropagation: theory, architectures, and applications*, 1-34. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Rumelhart, D.E., Hinton, G.E., Williams, R.J. (1989). Learning internal representation by error propagation. In: Anderson, J.A., Rosenfeld, E. *Neurocomputing. Foundation of Research*.



- Rey, G.D. (2009). *E-Learning - Theorien, Gestaltungsempfehlungen und Forschung*. Bern: Huber.
- Rey, G.D. & Wender, K.F. (2008). *Neuronale Netze – Eine Einführung in die Grundlagen, Anwendungen und Datenauswertung*. Bern: Huber.
- Sanger, T.D. (1989a). Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network. *Neural Networks*, Vol. 2, pp. 459-473.
- Sanger, T.D. (1989b). An Optimality Principle for unsupervised learning. In: Touretzky, D.S. *Advances in Neural Information Processing Systems I*.
- Sarle, W.S. (1994). Neural Networks and Statistical Models. In: <ftp://ftp.sas.com/pub/neural/neural1.ps> Zugriff: 6.7.2010
- Schnegg, M. & Lang, H. (2002). Netzwerkanalyse. Eine praxisorientierte Einführung. In: Lang & Schnegg (Hrsg.), *Methoden der Ethnographie*.
- Stanikunas, R. Vaitkevicius, H. & Kulikowski, J.J. (2004). Investigation of color constancy with a neural network. *Neural Networks*, 12, 13-28.
- Spackman, K.A. (1992). Maximum likelihood training of connectionist models: comparison with least squares back-propagation and logistic regression. In: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2247540> Zugriff: 22.11.2010-11-22
- Sun, R. (2008). *The Cambridge Handbook of Computational Psychology*.
- Szagun, G. (2006). *Sprachentwicklung beim Kind: Ein Lehrbuch*. Weinheim: Beltz Verlags Union. Kapitel 10, S. 267-292.
- Vracko, M. (2005). Kohonen Artificial Neural Network and Counter Propagation Neural Network in Molecular Structure-Toxicity Studies. In *Current Computer-Aided Drug Design*, 1, 73-78.
- White, H. (1989b). An Additional Hidden Unit Test for neglected Non-linearity in Multilayer Feedforward Networks. Proceedings of the International Joint Conferenc on Neural Networks. Washington, DC. San Diego: SOS Printing, II, 451-455
- Widrow, B., Hoff, M.E. (1989). Adaptive switching circuits. In: Anderson, J.A., Rosenfeld, E. *Neurocomputing. Foundation of Research*.

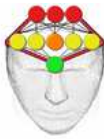
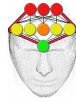


Appendix



| UV | AV |
|---------------------|--------------------|
| 0.07518115942028984 | 0.3305555555555556 |
| 0.25362318840579706 | 0.0833333333333333 |
| 0.2418478260869565 | 0.2055555555555556 |
| 0.1213768115942029 | 0.2111111111111111 |
| 0.05525362318840579 | 0.2083333333333337 |
| 0.12318840579710144 | 0.1972222222222224 |
| 0.0000000000000000 | 0.2972222222222223 |
| 0.1902173913043478 | 0.2027777777777778 |
| 0.15126811594202896 | 0.1944444444444444 |
| 0.3931159420289855 | 0.2166666666666667 |
| 0.3143115942028985 | 0.0000000000000000 |
| 0.5009057971014492 | 0.1861111111111112 |
| 0.47826086956521735 | 0.2027777777777778 |
| 0.2817028985507246 | 0.0638888888888889 |
| 0.3442028985507246 | 0.0722222222222226 |
| 0.24728260869565216 | 0.1833333333333335 |
| 0.25996376811594196 | 0.0388888888888889 |
| 0.41485507246376807 | 0.0666666666666667 |
| 0.5063405797101449 | 0.3305555555555556 |
| 0.4030797101449275 | 0.1472222222222225 |
| 0.38586956521739124 | 0.1638888888888892 |
| 0.36865942028985504 | 0.2416666666666667 |
| 0.7092391304347826 | 0.3222222222222224 |
| 0.3451086956521739 | 0.2805555555555556 |
| 0.4393115942028985 | 0.4638888888888889 |
| 0.4356884057971014 | 0.5027777777777778 |
| 0.47554347826086957 | 0.5694444444444445 |
| 0.6648550724637681 | 0.6055555555555556 |
| 0.7626811594202898 | 0.7833333333333334 |
| 0.6458333333333333 | 0.6916666666666667 |
| 0.5063405797101449 | 0.8944444444444445 |
| 0.648550724637681 | 0.8555555555555556 |
| 0.7219202898550724 | 1.0000000000000000 |
| 0.5932971014492754 | 0.7083333333333334 |
| 0.7083333333333333 | 0.7305555555555556 |
| 0.868659420289855 | 0.9222222222222223 |
| 0.7789855072463767 | 0.8666666666666667 |
| 0.7137681159420289 | 0.9444444444444444 |
| 0.6920289855072463 | 0.8305555555555557 |
| 0.7663043478260869 | 0.8277777777777778 |
| 0.8179347826086956 | 0.7166666666666667 |
| 0.8795289855072462 | 0.7000000000000001 |
| 0.8813405797101448 | 0.8416666666666667 |
| 0.8478260869565216 | 0.7027777777777778 |
| 0.8614130434782608 | 0.7750000000000001 |
| 0.9510869565217391 | 0.6638888888888889 |
| 0.9402173913043478 | 0.8138888888888889 |
| 0.9664855072463767 | 0.7555555555555556 |
| 0.9275362318840579 | 0.9527777777777777 |
| 1.0000000000000000 | 0.7416666666666667 |

Tabelle 1: Daten für Zahlenbeispiele



Neuronale Netze In Der Statistischen Datenauswertung?

Mit **F-I-S-C-H-E-R** sei an dieser Stelle ein Programm vorgestellt, das speziell zur statistischen Datenauswertung mittels herkömmlicher Verfahren **klassischer Statistik** und **Neuronaler Netze** konzipiert wurde und im Rahmen eines Forschungspraktikums an der Universität Würzburg sowie einer Diplomarbeit an der Universität Trier evaluiert wurde. Es wurde versucht, für klassisch-statistische Verfahren und für Neuronale Algorithmen eine einheitliche Form der Handhabung und Ergebnispräsentation zu schaffen, so dass es dem Anwender besonders leicht fallen sollte, Ergebnisse aus beiden Bereichen zueinander in Beziehung zu setzen und Neuronale Netze zur statistischen Datenauswertung zu nutzen.

Bisher stehen dem Nutzer des Programms folgende Analyse-Verfahren zur Verfügung:

- Lineare Regression & Polynomiale Regression
- ANOVA & t-Test
- Chi²-Test
- *Neuronale Netze* mit/ohne Hiddenunits (Generalized-Delta-Algorithm, Maximum-Likelihood-Backpropagation, Simulated Annealing, ...)
- Neuronale Netze für PCA (Generalized-Hebbian-Algorithm)
- Hopfield-Netze

Die neueste Version des Programms kann man kostenfrei unter der Adresse <http://www.psychologie.uni-heidelberg.de/ae/allg/mitarb/af/index.html> herunterladen (sollte jemand Anregungen haben oder gar Fehler im Programm finden wäre ich dankbar davon zu erfahren - andreasfischer1985@web.de).

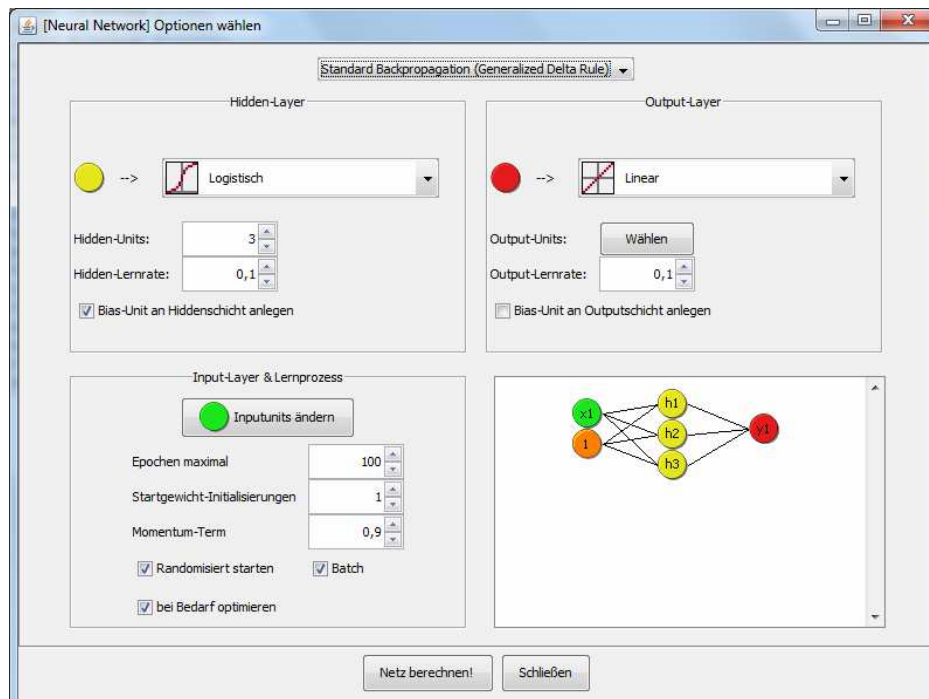
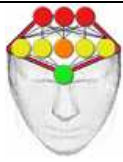
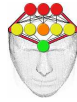


Abbildung: Screenshot aus der „Neuronale Netze Werkstatt“



Klassische Konditionierung In Der Sprache Neuronaler Netze?

Die Hebb'sche Regel wird häufig herangezogen, um zu veranschaulichen inwiefern ein konditionierter Stimulus (CS), der häufig gemeinsam mit einem unkonditionierten Stimulus (US) und einer darauf folgenden unkonditionierten Reaktion (UR) auftritt, zunehmend selbst in die Lage versetzt wird, besagte Reaktion auszulösen (Stimulus-Substitutions-Hypothese). Hierfür denkt man sich für CS und US jeweils eine Inputunit, und für die Reaktion eine Outputunit, und bestimmt die Gewichte nach folgender Lernregel:

$$\Delta w_{ij} = \epsilon * a_i * a_j,$$

Die Gewichtsveränderung zwischen zwei Einheiten ergibt sich also aus einem Lernparameter ϵ , der Aktivität a_j der sendenden Einheit j und der Aktivität $a_i = f(\sum (w_{ij} * a_j))$ der empfangenden Einheit i , wobei als Funktion f idR eine lineare oder sigmoide Funktion zum Tragen kommt. Einige Details der zeitlichen Zusammenhänge zwischen CS und US werden von diesem Modell jedoch nicht erklärt (so wirkt ein CS z.B. ausschließlich, wenn er zeitlich vor dem US präsentiert wird –für ein Modell, welches dieser Tatsache Rechnung trägt, in seinen zentralen Folgerungen jedoch dem im Folgenden dargestellten Rescorla-Wagner-Modell nahesteht, vgl. z.B. Sutton & Barto (1981) „Toward a Modern Theory of Adaptive Networks: Expectation and Prediction“). Auch geraten Hebb'sche Modelle in Erklärungsnot, wo die konditionierte Reaktion (CR) nicht identisch mit der unkonditionierten Reaktion (UR) ist, sondern lediglich ähnlich. Ein äußerst einflussreiches und auch heute noch gängiges Modell der Lerntheorie ist das **Rescorla-Wagner-Modell** der klassischen Konditionierung: Die Stärke V_j eines CS j (von der Assozierbarkeit/Salienz α_j) verändert sich angesichts eines US von der Stärke A , im Kontext aller CS V_{sum} in einem Konditionierungsdurchgang –sofern der CS j vorliegt und nur dann– um:

$$\Delta V_j = \alpha_j * (A - V_{\text{sum}})$$

In der Klammer steht also die Differenz zwischen der tatsächlichen Stärke des US und der (anhand der CS) vorhergesagten/erwarteten Stärke des US. Eben diese Differenz zwischen tatsächlichen und vorhergesagten Werten, entspricht in der Deltaregel neuronaler Netze dem namensgebenden Vorhersagefehler Delta. Die Deltaregel lässt sich folgendermaßen formulieren:

$$\Delta w_{ij} = \epsilon * (a_{i(\text{ideal})} - a_{i(\text{actual})}) * a_j,$$

D.h. die Gewichtsveränderung zwischen zwei Einheiten ergibt sich aus einem Lernparameter ϵ , der Aktivität a_j der sendenden Einheit j und der gewünschten Aktivität $a_{i(\text{ideal})}$ der empfangenden Einheit i , sowie der tatsächlichen Aktivität $a_{i(\text{actual})} = f(\sum (w_{ij} * a_j))$ der empfangenden Einheit i , wobei als Funktion f idR eine lineare oder sigmoide Funktion zum Tragen kommt. *Wie Sutton & Barto (1981) bemerkten, ist die Widrow-Hoff-Regel (also die Deltaregel) mit der Rescorla-Wagner-Gleichung im Kern identisch.* Die Gewichte w zwischen den Inputunits und einer Outputunit entsprechen der Stärke V eines CS bzgl. einer Reaktion y ,

- wenn man für jeden CS eine Inputunit j annimmt ($a_j=1$, wenn der CS vorliegt, $a_j=0$, wenn der CS nicht vorliegt),
- wenn man für den gewünschten Output $a_{i(\text{ideal})}$ die Stärke des US A annimmt (wobei $a_{i(\text{ideal})}=0$, wenn kein US vorliegt),
- wenn man $a_{i(\text{actual})}$ wie V_{sum} als die gewichtete Summe der CS-/Inputaktivitäten bestimmt, und
- wenn der Lernparameter ϵ dem Lernparameter α_j entspricht.

Aufgrund ihrer Parallelen zum (bzw. ihrer Übereinstimmung mit dem) Rescorla-Wagner-Modell bietet die Deltaregel eine adäquatere Beschreibung klassischer Konditionierung, als die Hebb'sche Regel (wobei angemerkt sei, dass Sutton und Barto in o.g. Artikel eine Erweiterung der Rescorla-Wagner/Widrow-Hoff-Regel vorschlugen, die auch noch dem zeitlichen Aspekt klassischer Konditionierung Rechnung trägt).

