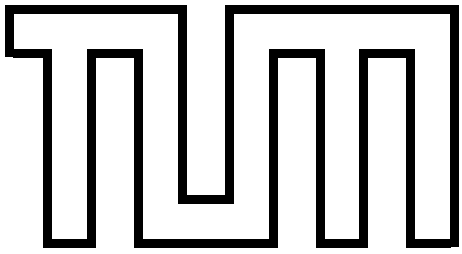

Technische Universität München

Fakultät für Informatik

Forschungs- und Lehreinheit Informatik VI

Neuronale Netze - Supervised Learning

Probleme des Backpropagation-Algorithmus und Alternativen

Seminar Kognitive Robotik (SS12)

Juan Jose Gonzalez

Betreuer: Dr. Florian Röhrbein

Leitung: Prof. Alois Knoll

Abgabetermin: 21. Juli 2012

Inhaltsverzeichnis

1 Vorwort

Wenn man das Thema Supervised Learning kennenlernt, wird man sehr oft mit dem Backpropagation-Algorithmus konfrontiert, entweder in der "standard"-Version, oder in einer der vielen verschiedenen Varianten. Backpropagation hat die Eigenschaft, dass man durch ändern der Parameter, wie Lernrate, Anzahl der Hidden Layers, Verbindungen, etc. sehr viele verschiedene Probleme lösen kann. Man findet auch sehr viel Dokumentation dazu, die dabei hilft, z.B. eine entscheidung zu Treffen, welche Parameter für eine bestimmte Problem-Art benötigt werden oder hilfreich sein könnten. Es ist aber wichtig zu wissen, dass Backpropagation nicht immer optimal ist, und das es Alternativen gibt. Die Flexibilität, die es anbietet, kann oft auch Nachteile mit sich bringen. Je nach Problem, was man lösen möchte, können andere Algorithmen deutlich performanter und einfacher zu implementieren sein.

2 Probleme vom Backpropagation-Algorithmus

Eines der größten Probleme von Backpropagation ist, dass in den meisten Fällen, besonders bei größeren Netzen, die Lernphase sehr langsam abläuft. Üblicherweise sind mehrere tausende Epochen notwendig, um ein gutes Ergebniss zu bekommen (vgl. [5]).

Ein wichtiger Grund dafür liegt in der Art, wie die Gewichte angepasst werden. Dadurch, dass ein Gradientenabstiegsverfahren über $\partial E/\partial \omega$ durchgeführt wird, wird der Lernprozess langsamer, je näher man an ein lokales Minumum kommt, da dort $\partial E/\partial \omega$ selbst gegen 0 geht, und somit die Lernschritte sehr klein sind (siehe Abbildung 1). Wenn man, um das Problem zu umgehen, eine größere Lernrate ε wählen würde, könnte der Lernalgorithmus möglicherweise falsche Ergebnisse liefern, indem es ein Minimum "überspringt".

Weiterhin werden bei Backpropagation alle Neuronen gleichzeitig jeweils für einen Element aus dem Traininsset trainiert. Das hat zur Folge, dass bei jedem Trainingsschritt alle Gewichte für dieses Element trainiert werden. Im nächsten Schritt geschieht das gleiche mit dem nächsten Element aus dem Trainingsset. Da die Neuronen innerhalb einer Schicht unabhängig voneinander aber mit den gleichen Werten trainiert werden, kann das gewünschte Ergebnis, dass jedes Neuron eine bestimmte Eigenschaft eines Inputs erkennt, nicht direkt erreicht werden. Stattdessen wird es nur nach einem sehr langen Wechselspiel zwischen den Gewichten der Verbindungen erreicht. Je mehr Hidden Layers ein Neuronales Netz enthält, desto komplexer ist das Wechselspiel zwischen den verschiedenen Neuronen und desto langsamer ist der Lernprozess[3].

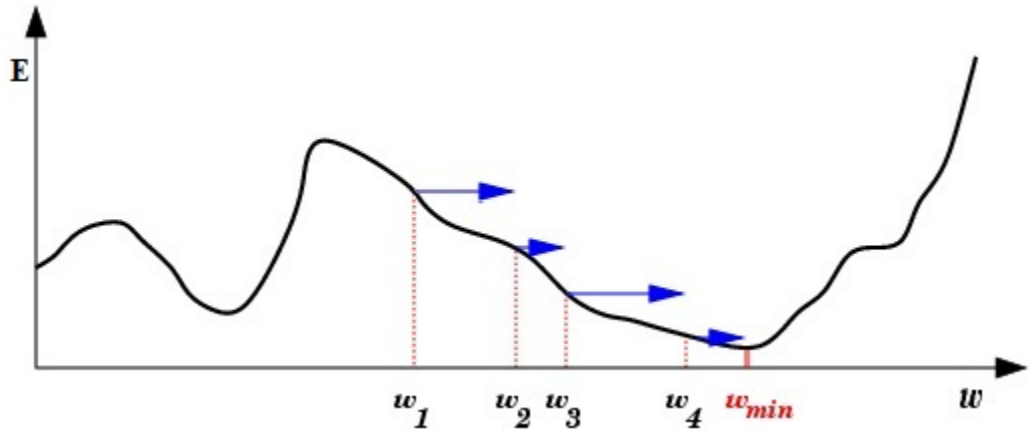

Abbildung 1: Beispiel für ein Gradientenabstiegsverfahren: Je mehr man sich an das Minimum nähert, desto kleiner werden die schritte, da die Ableitung der Kurve gegen Null konvergiert.

Quelle: `http://cs.uni-muenster.de/Studieren/Scripten/Lippe/wwwnnscript/index.html`

2.1 Beschleunigte Backpropagation

Um das Problem der langsamen Konvergenz des Backpropagation-Algorithmus zu lösen wurden mehrere Ansätze entwickelt. Viele davon basieren auf das Hinzufügen einer Art Moment (im physikalischem Sinne), wodurch $\Delta\omega$ nicht nur auf den aktuellen Fehlerwert E basiert, sondern auch auf die vorherigen Werte (siehe Abbildung 2). Ein solcher Algorithmus wurde von Rumelhart et al.[1] in einer einfachen Form vorgeschlagen:

$$\Delta\omega_{ij}(t) = -\varepsilon(\delta_j V_i) + \alpha\Delta\omega_{ij}(t-1)$$

wobei α eine Konstante bezeichnet, die die Auswirkung der vorigen Änderung von ω in der aktuellen Trainingsphase bestimmt.

Weitere Methoden, wie z.B die Newton'sche Methode oder das davon abgeleitete "Pseudo-Newton Algorithmus"[4], benutzen Informationen aus der zweiten Ableitung der Fehlerfunktion, um eine schnellere Approximation des Minumums zu erreichen.

3 Cascade-Correlation

1991 veröffentlichten Fahlmann und Lebiere einen Lernalgorithmus für Neuronale Netze, das die erwähnten Probleme von Backpropagation größtenteils löst.
Einer der größten Vorteile für den Benutzer vom Cascade-Correlation-Algorithmus ist, dass man die Struktur des Netztes nicht im Voraus definieren muss; der Algorithmus baut nach und nach die richtige Struktur durch wiederholtem Hinzufügen von Neuronen auf. Dabei

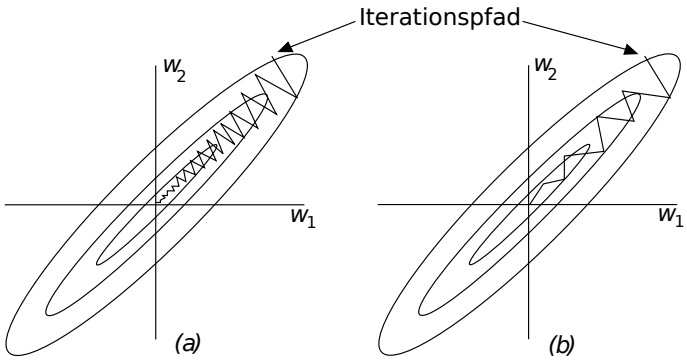

Abbildung 2: Konvergenz des Gradientenabstiegsverfahren in Backpropagation (a) und Backpropagation with Momentum (b). Während die Trainingsschritte bei Backpropagation immer kleiner werden, bleiben sie bei Backpropagation with Momentum auch in der nähe des Minimums groß.

Quelle: [2]

wird in jedem Schritt ein Neuron hinzugefügt, was den bisherigen Output des Netzes so gut wie möglich korrigiert.

3.1 Aufbau des Netzes

Der Algorithmus beginnt mit einem Neuronalen Netz ohne Hidden-Layers, das heißt nur mit direkten Verbindungen zwischen Inputs und Outputs. Die Anzahl und Art der Inputs und Outputs ist durch die Aufgabe, die das Neuronale Netz lösen soll, bestimmt. Die Gewichte ω dieser Verbindungen können mit Zufallswerten initialisiert, und dann mit der Deltaregel oder einem beliebigen Trainingsalgorithmus optimiert werden.

In jedem Trainingsschritt k werden mehrere "Kandidaten"-Neuronen erzeugt. Diese bekommen die Eingänge des Netzes V_x und die Ausgänge aller vorrigen Hidden-Layers V_{h_1} bis $V_{h_{k-1}}$ als Eingänge, mit beliebigen gewichten. Diese Kandidaten werden trainiert, d.h. ihre gewichte werden optimiert, und der Kandidat mit den besten Ergebnissen wird als neues Hidden Layer eingebaut (siehe Abbildung 3).

Der Wert V_{h_k} der Hidden Unit h_k in einem Neuronalen Netz mit n Eingängen ist also

$$V_{h_k} = f(\sum_{i=1}^{n}(\omega_{x_i h_k} V_{x_i}) + \sum_{i=1}^{k-1}(\omega_{h_i h_k} V_{h_i}))$$

für eine Aktivierungsfunktion f.

Dieser Prozess wird widerholt, bis das Ergebnis des Netzes "gut genug" ist. Hierbei ist die Bedeutung von "gut genug" abhängig vom Problem, das man lösen will: während für

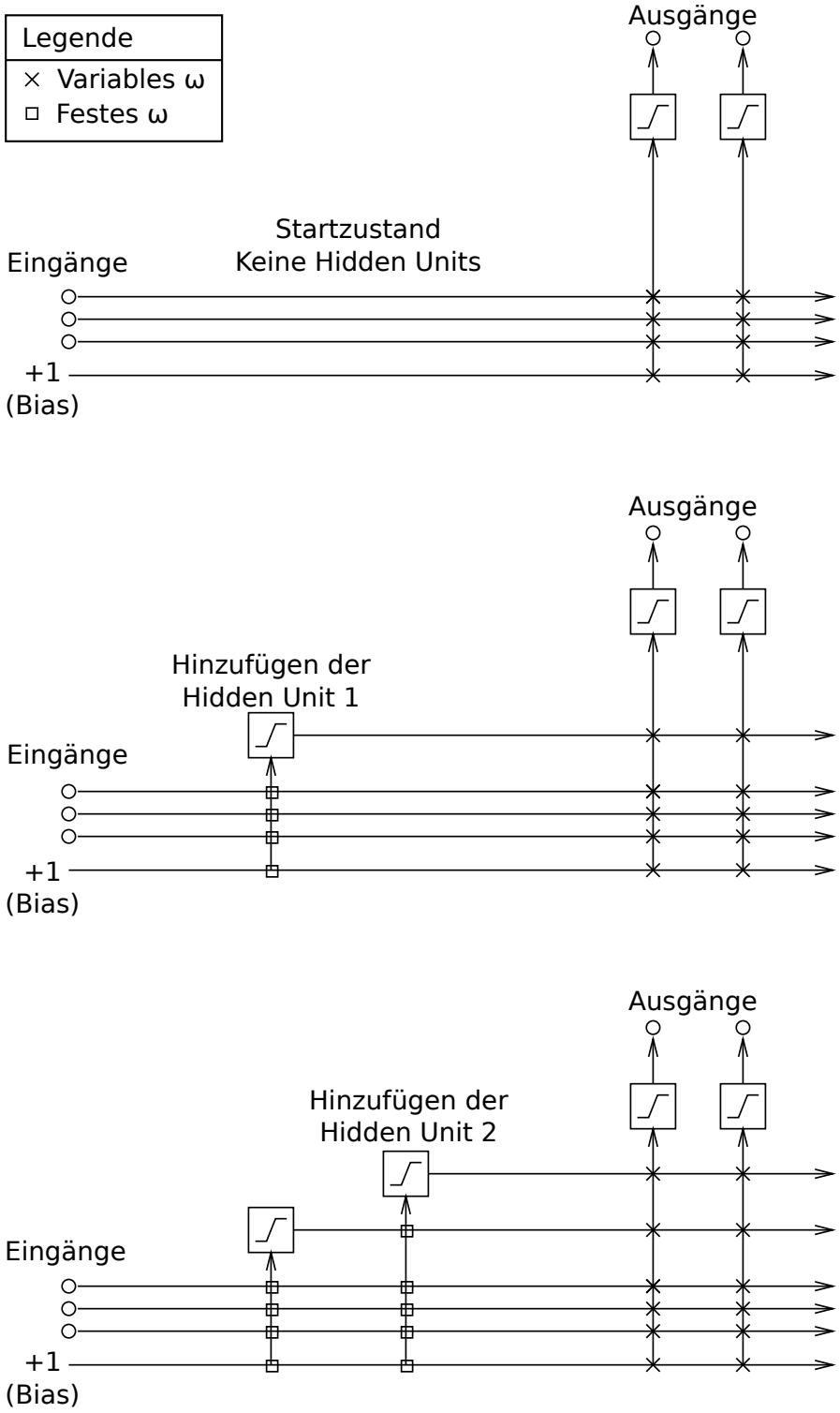

Abbildung 3: Die ersten zwei Schritte im Aufbau eines Neuronalen Netztes mit Cascade Correlation. Die vertikalen Linien zeigen die Eingänge der Neuronen, die aus der gewichteten Summe der damit verbundenen Ausgänge anderer Einheiten (horizontale Linien) berechnet werden.

Quelle: [3]

manche Probleme das Netz 100 Prozent der Trainingsdaten korrekt wiedergeben soll, kann man bei anderen Problemen auch leichte Abweichungen tolerieren.

3.2 Training der Kandidaten-Neuronen

Ziel des Trainings ist es, die Korrelation zwischen dem Ausgang des Kandidates und den Fehlern der Ausgangsneuronen zu maximieren, um beim Einbauen mit einer geeigneten Gewichtung maximal gegen diese Fehler wirken zu können. Sei für Kandidat k diese Korrelation definiert als

$$S = \sum_o \left| \sum_p (V_{p,k} - \bar{V}_k)(E_{p,o} - \bar{E}_o) \right|$$

wobei $\bar{V}_k$ und $\bar{E}_o$ jeweils der durchschnittliche Wert des Kandidats und der durchschnittliche Fehler des Ausgangsneuron o über alle Elemente und $V_{p,k}$ und $E_{p,o}$ der Wert und der Fehler für das Element p des Trainingssets darstellen.

Um S zu maximieren muss man, ähnlich wie im Backpropagation-Algorithmus, die partielle Ableitungen

$$\frac{\partial S}{\partial \omega_{ik}} = \sum_o \sum_p \sigma_o (E_{p,o} - \bar{E}_o) f'_p V_{p,i}$$

berechnen, wobei

$$\sigma_o = sgn(\sum_p (V_{p,k} - \bar{V}_k)(E_{p,o} - \bar{E}_o))$$

das Signum der Korrelation zwischen dem Wert von k und den Fehler von o, f' die Ableitung der Aktivierungsfunktion f des Kandidats und $V_{p,i}$ der Ausgangswert von Neuron i für das Trainingselement p symbolisieren.

Analog zu Backpropagation wendet man num einen Gradientenverfahren an, um S zu maximieren. Hierbei sind Verbesserungen des Backpropagation-Algorithmus wie zum Beispiel "Backpropagation with Momentum" teilweise auch anwendbar, um den Prozess zu beschleunigen.

Da alle Kandidaten voneinander unabhängig sind, können sie auch unterschiedliche Aktivierungsfunktionen besitzen, um einen möglichst breit verteilten Pool zu generieren, aus dem der beste Kandidat in das Netzwerk eingebaut werden kann.

3.3 Performanz des Algorithmus

In jedem Traininsschritt muss beim Cascade-Correlation-Algorithmus jeweils ein Pool von Kandidaten trainiert werden, aus dem am Ende nur eines als Hidden layer eingefügt wird, und der Rest verworfen wird. Allerdings korrigiert dieses Neuron den Output des Netzes viel gezielter als z.B ein Trainingsschritt des Backpropagation-Algorithmus, in dem alle

Gewichte auf einmal geändert werden, und somit möglicherweise die Änderung von einem Gewicht der Änderung eines anderen Gewichtes entgegenwirkt. Jedes Neuron wird bei Cascade-Correlation so trainiert, dass es eine bestimmte Eigenschaft der Eingangswerte erkennt. Dieses Neuron wird nach dem Einbauen immer nur diese Eigenschaft erkennen, sodass es für alle weiteren Neuronen wie ein weiterer fester Teil der Eingangswerte erscheint.

Weiterhin benötigt die Träiningsroutine eines Kandidates nur die Outputs des Netzwerks aus dem vorigen Schritt, und kann dann eigenständig ablaufen. Dadurch kann man das Trainieren der Kandidaten komplett parallelisieren, was bei den meisten Algorithmen aufgrund von gegenseitigen Abhängigkeiten nicht möglich ist.

3.3.1 Das "Two Spirals"-Problem

Um die Performanz des Algorithmus mit anderen bekannten Algorithmen wie z.B Backpropagation zu vergleichen, haben Fahlmann und Lebiere[3] eine Aufgabe gewählt, zu der es bereits mehere Testberichte gab: das "Two Spirals"-Problem. Es handelt sich um ein Trainingsset mit 194 X-Y-Koordinaten, die sich vom Ursprung aus als zwei ineinander liegende Spiralen ausbreiten. Als gewünschten Ausgang erhalten alle Punkte einer Spirale den Wert 1, die Punkte der Anderen den Wert -1 (siehe Abbildung 4)

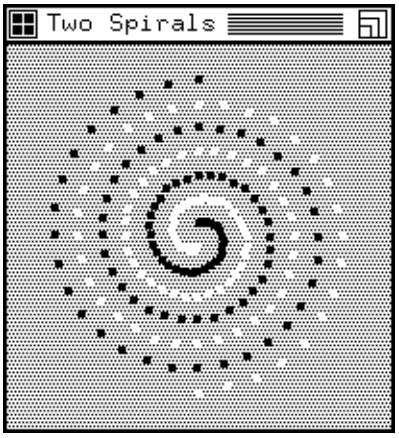

Abbildung 4: Das "Two Spirals"-Problem. Die Punkte stellen die Elemente des Trainingssets dar, mit ihren Koordinaten als Eingänge und die Farbe als Ausgang (1 bzw. -1).

Quelle: [3]

3.3.2 Ergebnisse des Tests

Mit einer leicht modifizierten Version von Backpropagation war es Wieland (nicht veröffentlicht) bisher gelungen, das "Two Spirals"-Problem in 150.000 bis 200.000 Epochen zu lösen. Mit einer besonderen Netzstruktur, in der jedes Neuron alle Ausgänge der vorigen Layers

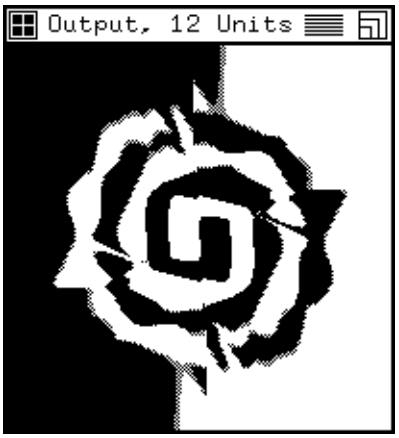

Abbildung 5: Ergebnis eines zum "Two Spirals"-Problem aufgebautem Neuronalen Netzes mit Cascade Correlation.

Quelle: [3]

bekommen hat, konnte das Problem unter Benutzung des Quickprop-Algorithmus mit dem bis zu dem Zeitpunkt bestem Ergebnis von 8000 Epochen gelöst werden[3].

Fahlmann und Lebiere haben das Cascade Correlation-Algorithmus 100 Mal mit dem Problem ausgeführt, bis das Neuronale Netz alle 194 Punkte vom Netz korrekt wiedergegeben wurden. Sie haben dabei einen Pool von 8 Kandidaten-Neuronen benutzt. Die Versuche haben durchschnittlich nur 1700 Epochen benötigt, mit einem Durchschnitt von 15,2 hinzugefügte Neuronen. Unter den Epochen sind dabei sowohl die Trainingsschritte aller Kandidaten als auch die Trainingsschritte der Ausgangs-Neuronen einbezogen.

Der Unterschied ist noch größer, wenn man nicht nur die Anzahl der Epochen betrachtet, sondern auch die benötigte Rechenleistung, um eine Epoche abzuarbeiten. Bei Backpropagation muss das gesamte Netz zwei Mal durchlaufen werden: zuerst von den Eingängen zu den Ausgängen, um den Ausgangswert zu berechnen, und dann von den Ausgängen zu den Eingängen, um $\Delta\omega$ zu berechnen. Da bei Cascade Correlation die Gewichte der eingebauten Hidden-Layers fest sind, muss nur eine "Schicht" von $\Delta\omega$-Werten berechnet werden.

Weiterhin wird ein Neuronales Netz mit Cascade Correlation nach und nach aufgebaut, sodass das Durchlaufen des Netzes am Anfang sehr schnell ist, während bei normaler Backpropagation alle Neuronen bereits existieren und durchlaufen werden müssen.

Ein weiterer Vorteil von Cascade Correlation ist, dass man die Ausgangswerte der vorigen Hidden-Layers für alle Elemente des Trainingssets in einem Cache ablegen kann, da sich diese nie ändern werden. Das ermöglicht ein sehr schnelles Training der Kandidaten und der Ausgangs-Neuronen, da bei genügend Speicher nur noch die neuen Werte berechnet werden müssen.

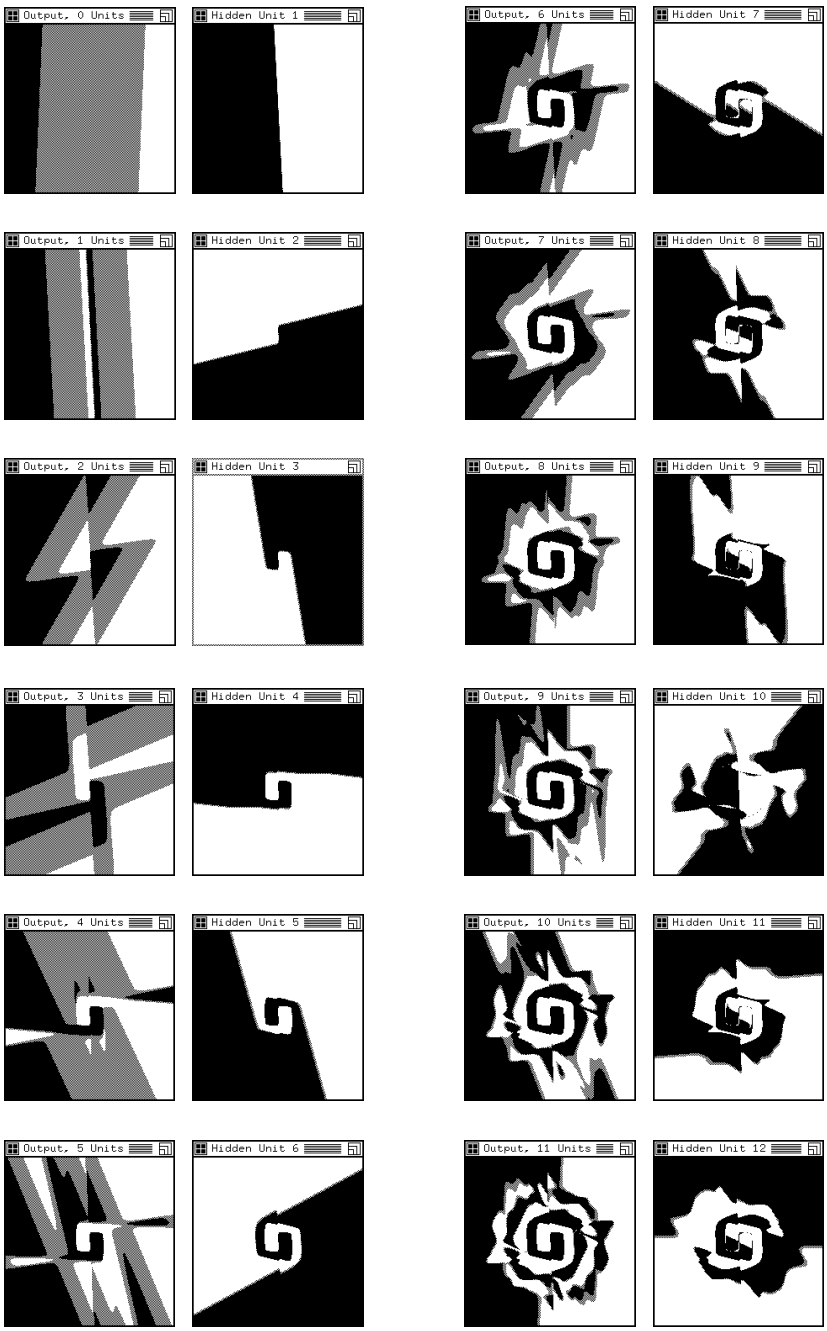

Abbildung 6: Aufbauschritte eines Neuronalen Netzes mit Cascade Correlation für das "Two Spirals"-Problem.

Quelle: [3]

Legende

ω_j:	Gewichte-Vektor der Eingänge von Neuron j
ω_{ij}:	Gewicht für die Verbindung von Neuron i nach Neuron j
V_i:	Ausgansgwert von Neuron i
x_i:	Eingangsneuronen
o_i:	Ausgangsneuronen
h_i:	Neuronen der Hidden-Layers

Tabelle 1: Legende

Literatur

[1] D. E. Rumelhart, G. E. Hinton and R. J. Williams. Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the mictrostructure of cognition, volume I*, pages 318–362. Bradford Books, 1986.

[2] Raúl Rojas. *Neural Networks - A Systematic Introduction.* Springer Verlag, 1996.

[3] Scott E. Fahlman and Christian Lebiere. *The Cascade-Correlation Learning Architecture (CMU-CS-90-100).* 1991.

[4] Sue Becker and Yann le Cun. Improving the convergence of back-propagation learning with second order methods. In *Proc. of the 1988 Connectionist Models Summer School*, pages 29–37. D. Touretzky, G. Hinton and T Sejnowski, editors, 1989.

[5] W. Schiffmann, M. Joost, R. Werner. *Comparison of Optimized Backpropagation Algorithms.* ESANN 93, Brüssel, 1993.