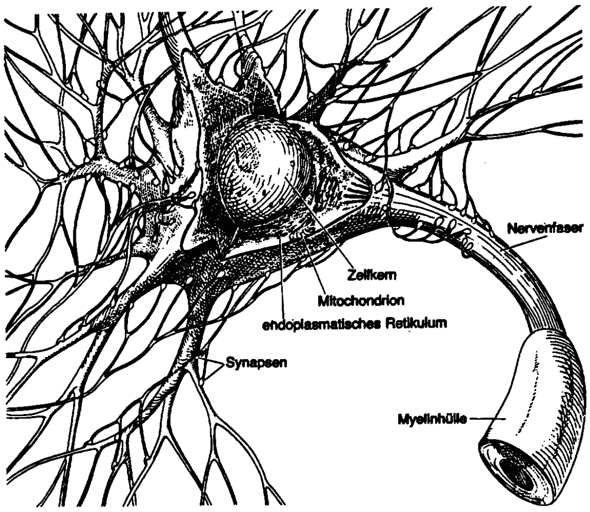
Einführung in die Neuronalen Netze und ihre Anwendung auf die Formel von Black-Scholes

Seminar Econo Physics WS 98/99

Jens Fischer



08.12.1998

Zusammenfassung

Bei der praktischen Arbeit mit der Formel von Black, Scholes (1973) und Merton (1973), für die es eine geschlosssene analytische Form für Optionspreise durch dynamische hedging-Argumente und das no arbitrage Prinzip gibt, erhält man durch Verallgemeinerung realistischere Bedingungen. Dies hat jedoch zur Folge, daß die geschlossene Form verloren geht. Hier kann man nun einen numerischen Ansatz gehen oder auf ein modernes Hilfsmittel in den Naturwissenschaften zurückgreifen - den Neuronalen Netzen. Um nun dieses Hilfsmittel ein wenig kennenzulernen, das in vielen Bereichen der Wissenschaft und der Industrie Einzug gehalten hat, wird im folgenden das Prinzip dargelegt. Als Anwendungsbeispiel wird im letzten Kapitel versucht zu zeigen, daß Neuronale Netze die Aufgabe der Formel von Black-Scholes prinzipiell übernehmen können und zum Berechnen eines fairen Optionspreises eingesetzt werden können.

Inhaltsverzeichnis

Kapitel 1

Grundlagen Neuronale Netze

Obwohl die Funktionsweise des Gehirns noch nicht komplett verstanden ist, existieren in der Medizin Modelle zur Erklärung. Bei der Entwicklung Neuronaler Netze dienten diese Modelle als Vorbild. Nicht nur die physikalische Struktur des Gehirns oder des Nervensystems standen Pate, sondern auch bei vielen Lernalgorithmen wird versucht, Lern- oder auch Erkennungsverhalten des Gerhirns nachzuahmen.

1.1 Analogie zur Biologie

Im Gehirn gibt es eine Vielzahl von Nervenzelltypen, die alle unterschiedliche Aufgaben erfüllen. Die Abbildung 1.1 zeigt einen Schnitt durch den Kortex, in dem nur wenige Neuronen mit der Golgi-Anfärbetechnik sichtbar gemacht wurden.

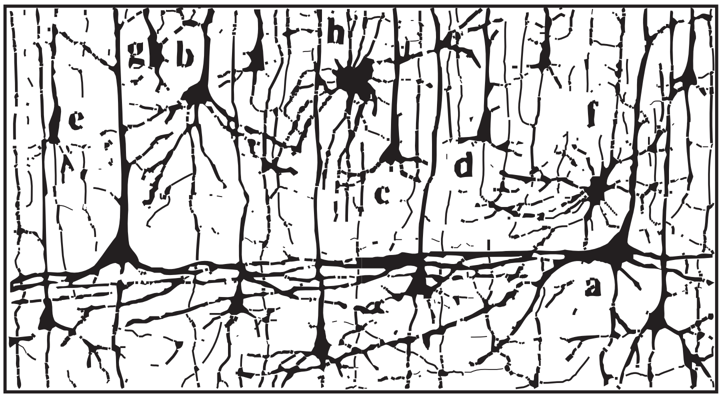


Abb.1.1: Biologische Neuronen

Das einzelne Neuron (Abb.1.2) besteht aus baumartig verzweigten Dendriten, dem Zellkörper, dem Zellkern und dem relativ langen Axon (Nervenfaser) mit den Synapsen.

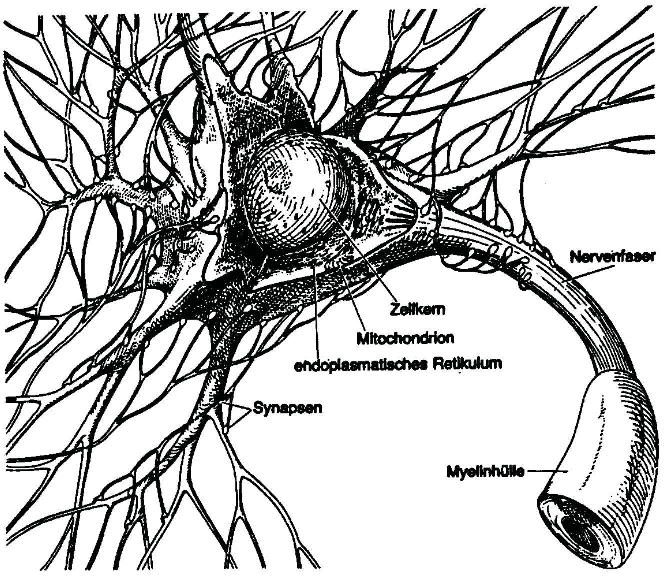


Abb.1.2: Eine Pyramidenzelle (nach [Cajal, 1955])

Die Zelle ist umgeben von der Zellmembran. Sie enthält neben dem Zellkern diverse andere funktionelle Einheiten (sog. Organellen), wie den Golgi-Apparat, Mitochondrien, endoplasmatisches Retikulum usw., die für die Arbeit des Neurons notwendig sind. So sind u.a. die Mitochondrien für die Energieversorgung der Zelle zuständig. Der Zellkern hat die Aufgabe die Eingangssignale zu verarbeiten und unter gewissen Randbedingungen die Weiterleitung von Ausgangssignalen anzustoßen. Das Axon (die Nervenfaser) übernimmt die Weiterleitung der Ausgangssignale des Neurons. Das Axon verdichtet sich an ihrem Ende und bildet die Synapse. Die Dentriten sind dünne, röhrenförmige und meist stark verästelte Fortsätze der Zelle, mit denen die Zelle Eingangssignale aufnimmt. Die Synapsen, die Endköpfchen des Axons, bilden über den synaptischen Spalt die Kontaktstelle zwischen den Enden des Axons und den Dendriten von weiteren Neuronen. Die synaptische Übertragung ist zum größten Teil chemischer Art. Man unterteilt sie in hemmende und erregende Übertragung. Bei jedem ankommenden elektrischen Reiz schütten die Synapsen aus einigen Vesikeln den Inhalt, den sogenannten Neurotransmitter, aus. Dieser gelangt über den synaptischen Spalt auf die postsynaptische Zellmembran eines Dendriten der nächsten Zelle. Hier sorgen Rezeptormoleküle für die Steuerung der (Na^+) Natrium- und (K^+) Kaliumkanäle (bei Erregung) bzw. (Cl^-) Chlor- und (K^+) Kaliumkanäle (bei Hemmung). Eine Erregung hat eine Depolarisation der Außenseite gegenüber der Innenseite der Membran zur Folge, eine Hemmung bewirkt eine Hyperpolarisation (Abb.1.3).

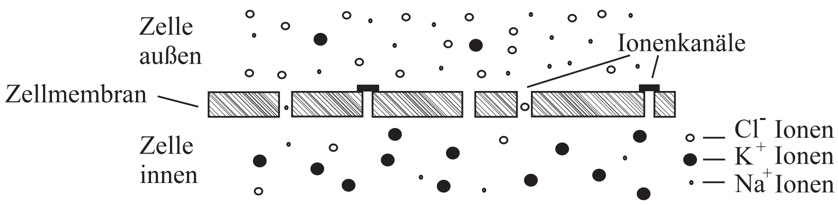


Abb.1.3: Membranpotential und Ionenkanäle

Da die Leitfähigkeit im Bereich der Dendriten überwiegend passiv ist, da nicht besonders viele Ionenkanäle für eine aktive Weiterleitung dieser Ladungsverschiebungen existieren, werden die Potentiale, die in der Umgebung der Synapsen entstehen, sowohl räumlich als auch zeitlich aufintegriert.
Eine Nervenzelle hat ein Ruhepotential von -70 Millivolt (Innen gegenüber Außen). Bewirken die aufintegrierten Potentiale kurzzeitig eine Überschreitung der Schwelle von etwa -60 Millivolt, so strömt Natrium in die Zelle und Kalium heraus. Das Potential steigt so bis auf etwa +30 Millivolt und breitet sich am Axon, welches mit Ionenkanälen bespickt ist, in Form eines Aktionspotentials wellenartig aus. Nach kurzer Zeit stellt sich durch den Rücktransport der Ionen wieder das Ruhepotential ein. Nach einer Refraktärzeit kann ein neuer Impuls entstehen. Je dichter die Impulse aufeinander folgen, d.h. je größer die Frequenz der Impulsfolge ist, desto mehr Transmitter werden pro Zeit an den Endköpfchen der Synapsen ausgeschüttet und desto größer ist der hemmende oder erregende Einfluß auf die nachfolgende Zelle. Außer der Frequenz sind noch die Verbindungsstärke und die Erregbarkeit der Membran in der Umgebung der Synapsen maßgebend für die Wirkungsweise, die die Synapse auf die Gesamtzelle ausüben kann.
(Mit freundlicher Genehmigung mit Modifikationen aus [Fischer].)

1.2 Grundlegender Aufbau eines künstlichen Neuronalen Netzes

„Ein Neuronales Netz ist ein massiv parallel verteilter Prozessor, der eine natürliche Neigung zum Speichern und Verfügbarmachen von auf Erfahrung basierendem Wissen hat. Es ähnelt dem Gehirn in zwei Bereichen:
1. Wissen wird durch einen Lernprozeß angeeignet.
2. Die Stärke von Verbindungen zwischen den Neuronen, auch bezeichnet als Synapsen wird genutzt, um Wissen zu speichern .“

Aleksander und Morton 1990

Die vorangegangene Definition betrachtet Neuronale Netze als adaptive Maschine. Implementiert werden solche Netze durch elektronische Komponenten oder durch Softwaresimulation auf konventionellen Computern.
Neuronale Netze bestehen im allgemeinen aus folgenden Komponenten (nach [Zell]):

1. **Neuronen** (Abb. 1.4) bestehen aus :

   - *Aktivierungszustand* a_i(t): Er gibt den Grad der Aktivierung des Neurons an.
   - *Aktivierungsfunktion* f_{act}: Diese Funktion errechnet den neuen Aktivierungszustand a_j(t+1)des Neurons j aus der alten Aktivierung a_j(t), der Netzeingabe net_j(t) und dem Schwellwert θ_j(t) über

   $$a_j(t+1) = f_{act}(a_j(t), net_j(t), \theta_j(t)). \tag{1.1}$$

   - *Ausgabefunktion* f_{out}: Das Ausgabesignal des Neurons j wird mit dieser Funktion aus der Aktivierung der Zelle bestimmt

   $$o_j = f_{out}(a_j). \tag{1.2}$$

   (Aktivierungs- und Ausgabefunktion eines Neurons werden häufig verknüpft als *Transferfunktion* bezeichnet)

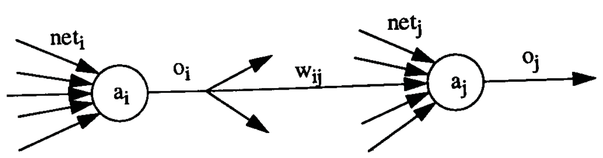


Abb.1.4: Darstellung von Zellen eines Neuronalen Netzes

2. **Verbindungsnetzwerk** der Neuronen. Ein neuronales Netz kann als gerichteter, gewichteter Graph angesehen werden, wobei die Kanten die gewichteten Verbindungen zwischen den Neuronen darstellen. Das Gewicht der Verbindung von Neuron i nach Neuron j wird durch w_{ij} bezeichnet. Die Matrix der Verbindungen aller Zellen (Gewichtsmatrix) wird dann mit W bezeichnet.

3. **Propagierungsfunktion**. Diese Funktion errechnet, wie sich die Netzeingabe eines Neurons aus den Ausgaben der anderen Neuronen und den Verbindungsgewichten. Die Netzeingabe net_j(t) von Neuron j berechnet sich nach

$$net_j(t) = \sum_i o_i(t) w_{ij}. \tag{1.3}$$

4. **Lernregel**. Die Lernregel ist ein Algorithmus, gemäß dem das neuronale Netz lernt, für die vorgegebene Eingabe eine gewünschte Ausgabe zu produzieren. Lernen erfolgt in neuronalen Netzen meist durch Modifikation der Stärke der Verbindungen als Ergebnis der wiederholten Präsentation von Trainingsmustern.

Beliebige disjunkte Mengen von Neuronen können zu einer *Schicht* (Abb.1.5) zusammengefaßt werden. Alle Neuronen einer Schicht sollen die gleiche Transferfunktion besitzen und werden gleichzeitig aktiviert. Die Neuronen, die Eingabeverbindungen aus der „Außenwelt“ empfangen, werden oft in der Eingabeschicht zusammengefaßt. Entsprechend bilden die Neuronen, die ihre Ausgaben an die „Außenwelt“ schicken, die Ausgabeschicht. Die übrigen Schichten nennt man verborgen oder verdeckt, da sie keine Verbindungen nach „Außen“ besitzen. Nach dieser Einteilung ergeben sich drei Typen von Neuronen - Eingabeneuronen, Ausgabeneuronen und verdeckte Neuronen. Wenn man von Netzen spricht mit 5 Schichten gibt die Zahl nur die Zahl der informationsverarbeitenden Schichten an. Da die Eingangsschicht jedoch nur Informationen von außen weiterleitet, wird diese nicht mitgezählt.

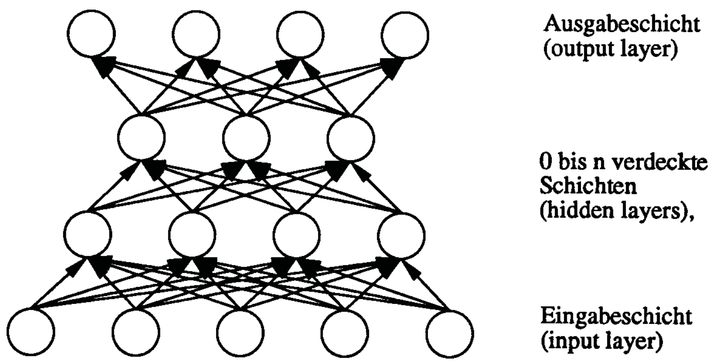


Abb.1.5: Typen von Neuronen

Kapitel 2

Architekturen Neuronaler Netze

Neuronale Netze lassen sich abhängig von der Struktur der Neuronenverbindungen und ihrer Funktionsweise in unterschiedliche Klassen einteilen. Grob teilt man Neuronale Netze in *feedforward* und *rekursive* Netze ein. Hierbei heißt rekursiv, daß ein Neuron Output an Neuronen vorheriger Schichten zurückgibt (Rückkopplung). Die Art der verwandten Architektur hängt stark mit der benutzten Lernmethode zusammen, mit der das Netz trainiert wird. Man kann Neuronale Netze auf Grund ihrer Struktur im wesentlichen in vier Klassen von Architekturen einteilen.

2.1 Einschicht feedforward Netze

Die einfachste Form von in Schichten eingeteilten Neuronalen Netzen ist das Einschicht Netz. Diese Neuronalen Netze haben nur eine Schicht von Eingabeneuronen, die direkt auf eine Schicht von Ausgabeneuronen projiziert. Da es sich hier um ein *feedforward* Netz handelt, werden keine Daten von den Ausgabeneuronen an die vorherige Schicht zurückgereicht. Ein Beispiel für diese Einschicht *feedforward* Netze ist ein linearer assoziativer Speicher. Bei einer derartigen Aufgabe assoziiert das Netz Eingaben mit bestimmten Ausgaben. (z.B. Ziffernerkennung)

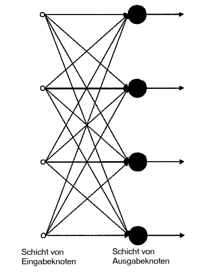


Abb.2.1: Feedforward Netz mit einer Schicht von Neuronen

2.2 Mehrschicht feedforward Netze

Eine weitere Klasse unterscheidet sich von der ersten Klasse nicht-rückgekoppelter Netze durch das Vorhandensein von verdeckten Schichten. Die Aufgabe der Neuronen in diesen verdeckten Schichten liegt darin, zwischen der Eingabeschicht und der Ausgabeschicht zu vermitteln. Durch das Hinzufügen dieser Schichten nimmt die Fähigkeit des Netzes in Bezug auf die Analysemöglichkeiten der Eingabedaten und der Speicherung zu. Weiter spricht man von vollständig verbundenen Netzen, wenn jedes Neuron zu jedem Neuron seiner Nachfolgeschicht eine Ausgangsverbindung hat, bzw. zu jedem Neuron seiner Vorgängerschicht eine Eingangsverbindung hat. Fehlen einige dieser Verbindungen nennt man sie partiell verbundene Netze. Eine weitere Besonderheit sind sogenannte Shortcut-Verbindungen (Abb. 2.2), von diesen spricht man, wenn Verbindungen zwischen Neuronen bestehen zwischen denen mindestens eine Schicht liegt.

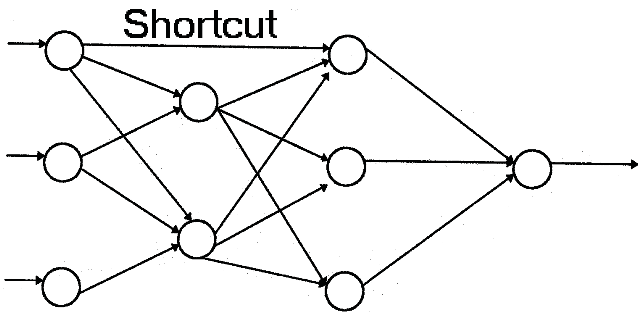


Abb.2.2: Netz mit Shortcuts

2.3 Rekursive Neuronale Netze

Diese Netze unterscheiden sich von feedforward Netzen dadurch, daß sie mindestens eine Rückkopplung haben. Von Eigenrückkopplung (*direct feedback* im Gegensatz zu *indirekt feedback*) spricht man, wenn ein Neuron seine Ausgabe an seine Eingabe zurückkoppelt. Eine dritte Möglichkeit der Rückkopplung ist diejenige, die innerhalb einer Schicht (*lateral feedback*) auftritt. Rekursionen haben wesentlichen Einfluß auf die Lernfähigkeit und Leistungsfähigkeit von Neuronalen Netzen. Ein Beispiel für rekursive Neuronale Netze mit indirekter Rückkopplung sind die Hopfield-Netze (s.Abb 2.3). Diese Netze haben noch zwei Einschränkungen, ihre Gewichtsmatrix ist symmetrisch und die Diagonalelemente sind Null. Eine Anwendung für das Hopfield-Netz ist die Lösung des „schweren“ Optimierungsproblems “Traveling Salesman Problem“(TSP). Hier besteht die Aufgabe darin, den kürzesten Weg zu finden, der eine gegebene Anzahl von Städten über Verkehrslinien verbindet, wobei jede Stadt genau einmal besucht wird und der Weg zum Ausgangspunkt zurückführt.

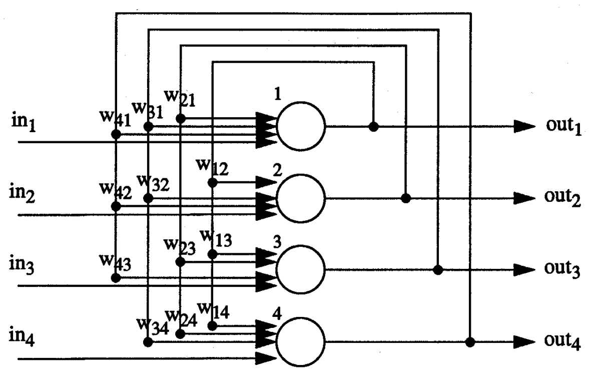


Abb.2.3: Hopfield-Netz

2.4 Gitternetz Strukturen

Unter einem Gitter versteht man eine ein- oder mehrdimensionale Anordnung von Neuronen mit einer dazugehörigen Menge von Knoten, die die Eingangssignale an diese Anordnung weiterreichen.

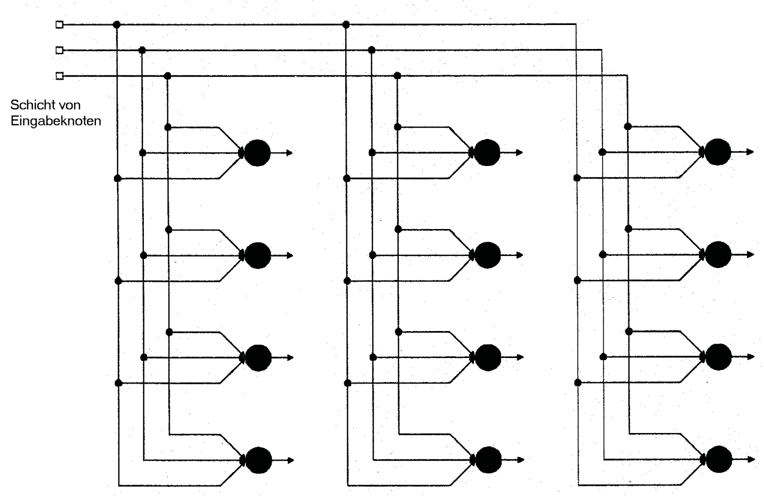


Abb.2.4: Beispiel einer Gitterstruktur

Kapitel 3

Lernen - Wissensabbildung in Neuronalen Netzen

Das Lernen eines Neuronalen Netzes geschieht durch Selbstmodifikation gemäß einer fest vorgeschriebenen Lernregel, indem die variablen Parameter des Netzes an die Umwelt angepaßt werden. Es gibt nun verschiedene Möglichkeiten wie ein Netz lernen kann, nämlich:

- Entwicklung und Löschen von Verbindungen
- Modifikation der Stärken der Verbindungen (Veränderung der Gewichte)
- Modifikation der Schwellenwerte
- Modifikation von Aktivierungs-, Propagierungs- und Ausgabefunktion
- Entwicklung und Löschen von Zellen

Unter diesen Methoden ist die zweite, das Lernen durch Veränderung der Gewichte, die am häufigsten verwendete. Mit dieser werden wir uns im folgenden beschäftigen. Neuerdings haben Verfahren, die auch die Topologie des Netzes verändern, zugenommen.
Betrachtet man nun ein Neuron i, so ist es unstrittig, daß die Ausgabe dieses Neurons von den Gewichten der Synapsen abhängt. Durch Veränderung dieser Gewichte kann man nun den Lerneffekt erzielen. Sei w_{ij}(n) das Gewicht w_{ij} zum Zeitpunkt n, dann bewirkt Lernen im Zeitpunkt n eine Veränderung von w_{ij}(n) zu w_{ij}(n+1) wie folgt:

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n). \tag{3.1}$$

Die Berechnung der jeweiligen Δw_{ij}(n) ist nun Aufgabe der verschiedenen Lernverfahren, von denen drei grundlegende behandelt werden. Als Lernparadigmen werden die verschiedenen Situationen bezeichnet, in denen sich ein lernendes Netz befinden kann, wie das Netz also mit der Umwelt interagiert.

3.1 Lernverfahren

3.1.1 Hebbsches Lernen

Die erste Lernregel wurde von dem Psychologen Donald Hebb im Jahre 1949 veröffentlicht:

1. *Wenn zwei Neuronen an den Enden einer Synapse simultan (synchron) reagieren, so wird die Synapse gestärkt.*

2. *Wenn andererseits zwei Neuronen an den Enden einer Synapse nicht simultan (asynchron) reagieren, so wird die Stärke der Synapse verringert.*

Diese Lernregel ist die Grundlage für die meisten komplizierten Lernregeln. In der mathematischen Form lautet die Hebbsche Regel wie folgt:

$$\Delta w_{ij} = \eta o_i a_j . \tag{3.2}$$

Hier ist η eine Konstante, die sogenannte Lernrate, o_i die Ausgabe der Vorgängerzelle i und a_j die Aktivierung der Nachfolgerzelle j.
Dieser Mechanismus besitzt die vier Eigenschaften:

1. Zeitabhängigkeit: Die Modifikationen innerhalb des Lernprozesses hängen vom exakten Zeitpunkt ab, an dem sogenannte prä- und postsynaptische Aktivitäten auftreten.

2. Lokalität: Modifikationen werden rein aus lokal verfügbarer Information hergeleitet, es werden nur die prä- und postsynaptischen Aktivitäten und keine Information von außen benötigt.

3. Interaktivität: Die Modifikationen kommen in einem interaktiven Prozeß zwischen prä- und postsynaptischen Aktivitäten zustande und hängen vom jeweiligen Grad der Aktivitäten in den Neuronen vor und hinter der Synapse ab.

4. Konjunktion, Korrelation: Diese Eigenschaften ergeben sich aus dem Umstand, daß die Konjunktion von prä- und postsynaptischen Aktivitäten essentielle Bedingungen für den Lernprozeß sind. Aus dem Grad beider Aktivitäten ergeben sich die entscheidenen Veränderungen im Lernprozeß.

Die Stärkung einer Synapse resultiert aus der positiven Korrelation prä- und postsynaptischer Aktivitäten, während negativ korrelierte oder nicht korrelierte Aktivitäten zu einer Schwächung der Synapse zwischen den beiden betroffenen Neuronen führen. Synapsen mit diesen Eigenschaften werden hebbsche Synapsen genannt.

3.1.2 Lernen durch Fehlerkorrektur

Sei d_k(n) eine gewünschte Ausgabe des Neuron k zum Zeitpunkt n. Der zu diesem Zeitpunkt tatsächliche Output y_k(n) wird durch den Inputvektor x_k(n) erzeugt. Das Lernverfahren basiert nun darauf, daß dem Netz zu jedem Zeitpunkt n ein Paar (x_k(n),d_k(n)) präsentiert wird und das Netz sein Verhalten anhand der Abweichung der tatsächlichen Ausgabe vom Zielwert verbessert. Das Netz versucht sich also der gewünschten Ausgabe durch verändern der Gewichte anzunähern. Diese Abweichung definiert das Fehlersignal e_k(n)=d_k(n)-y_k(n). Das letztliche Ziel dieses Verfahrens besteht in der Minimierung einer Kostenfunktion basierend auf diesen Fehler. Mit geeigneter Kostenfunktion ergibt sich so ein Optimierungsproblem. Eine häufig verwandte Kostenfunktion ist der mittlere quadratische Fehler

$$E(n) = \frac{1}{2}\sum_k e_k^2(n). \tag{3.3}$$

Das Netz wird nun optimiert, indem E(n) minimiert wird, was z.B. unter Berücksichtigung der sogenannten Delta-Regel zu folgendem führt:

$$\Delta w_{ij}(n) = \eta e_i(n) x_j(n). \tag{3.4}$$

Hier ist η wieder die Lernrate. Δw_{ij}(n) ist also proportional zum Fehlersignal und Inputsignal des betrachteten Neurons k (wobei hier x_j(n)=y_i(n) ist). Dem Parameter η kommt nun bei diesem Lernverfahren eine wichtige Rolle zu. Ist nämlich η groß, geht der Lernprozeß zwar schnell, der Prozeß kann jedoch instabil werden und divergieren. Wenn η aber klein ist, dauert der Lernvorgang unter Umständen sehr lange. Trägt man nun die Kostenfunktion E gegen die Synapsengewichte auf, erhält man die Fehleroberfläche. Man

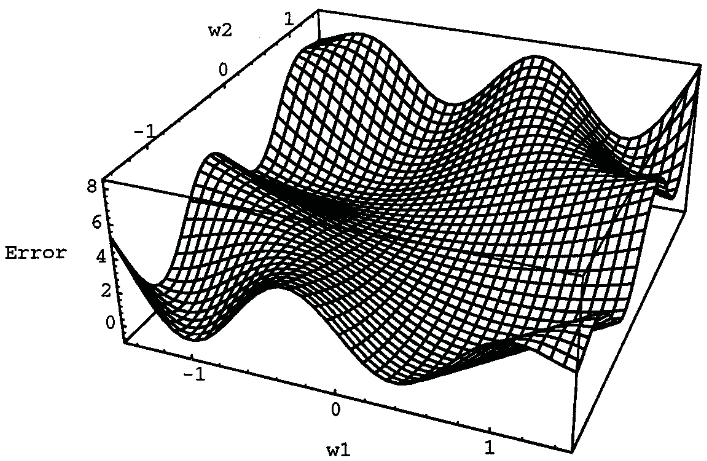


Abb.3.1: Fehleroberfläche eines Neuronalen Netzes

sieht, daß die Gefahr besteht während eines Lernprozeßes in einem der lokalen Minimum hängenzubleiben und so nicht die optimale Gewichtsparameter zu finden. Man muß den Lernprozeß abbrechen, wenn das gewünschte Lernziel erreicht ist, die Kostenfunktion also einen bestimmten kleinen Wert erreicht hat.

3.1.3 Backpropagation

Die Backpropagation-Regel ist eine Verallgemeinerung der Delta-Regel für Netze mit mehr als einer Schicht trainierbarer Gewichte, dessen Neuronen Aktivierungsfunktionen besitzen die semilinear (monoton und differenzierbar) sind. Dieses Lernverfahren ist nun ein Gradientenabstiegsverfahren, bei dem der Gradient der Fehlerfunktion (Abb. 3.1) berechnet wird und die Gewichte um einen Bruchteil des negativen Gradienten der Fehlerfunktion geändert werden.

$$\Delta W = -\eta \nabla E(W) \Rightarrow \Delta w_{ij} = -\eta \frac{\partial}{\partial w_{ij}} E(W) \tag{3.5}$$

Hier ist η wieder die Lernrate oder die Schrittweite. Als Fehlerfunktion wird auch hier häufig der quadratische Fehler angenommen. So erhält man für die Summe der Fehler über alle Muster p den folgenden Ausdruck:

$$E = \sum_p E_p; E_p = \frac{1}{2} \sum_j (t_{pj} - o_{pj})^2. \tag{3.6}$$

Dabei sind E_p der Fehler für ein Muster p, t_{pj} die Lerneingabe, o_{pj} die Ausgabe von Neuron j bei Muster p. Nach einer etwas längeren Herleitung erhält man dann die folgende Backpropagation-Regel (online backpropagation):

$$\Delta_p w_{ij}(n) = \eta o_{pi} \delta_{pj}. \tag{3.7}$$

mit

$$\delta_{pj} = \begin{cases} o_{pj}(1 - o_{pj})(t_{pj} - o_{pj}) & \text{falls j Ausgabeneuron ist} \\ o_{pj}(1 - o_{pj}) \sum_k \delta_{pk} w_{jk} & \text{falls j verdecktes Neuron ist} \end{cases} \tag{3.8}$$

Hier erkennt man, woher das Verfahren seinen Namen hat, das Netz wird von hinten nach vorne verbessert.

3.1.4 Konkurrierendes Lernen

Wie der Name schon sagt, konkurrieren die Neuronen der Ausgabeschicht darum, aktiv zu sein. Im Gegensatz zu anderen Verfahren, wo mehrere Ausgabeneuronen gleichzeitig aktiv sein können, tätigt hier zu einem Zeitpunkt nur ein Neuron eine Ausgabe. Diese Art von Netzen werden zur Mustererkennung oder -klassifikation eingesetzt. Die wesentlichen Merkmale dieses Lernverfahrens sind:

- Eine Menge von gleichen Neuronen, die sich nur in ihren synaptischen Gewichten unterscheiden.

- Es existiert ein Grenzwert für die Stärke jedes Neurons.
- Ein Mechanismus, der es den Neuronen erlaubt, um das Recht zu konkurrieren, auf einen bestimmten Input zu antworten, so daß nur ein Ausgabeneuron (oder eines einer Gruppe) zu einem Zeitpunkt aktiv ist. Dies Neuron wird dann Gewinner genannt (the-winner-takes-it-all-neuron).

Eine mögliche Realisierung ist folgende: wenn ein Inputvektor an das Netz angelegt wird, gewinnt das Neuron k mit der größten Summe über die gewichteten Inputs $v_k(n) = \sum_i w_{ki}(n)x_i$ den Konkurrenzkampf und wird aktiv sein. Seine Ausgabe y_k wird auf 1 gesetzt. Alle anderen Neuronen schalten ab und geben 0 aus. Das gewinnende Neuron verändert seine interne Struktur so, daß es beim Mal noch besser auf diesen Input reagiert. Um zu verhindern, daß dasselbe Neuron in jedem Fall (bei jeder Eingabe) gewinnt, muß den Gewichten ein Grenzwert auferlegt werden. Mit $\sum_i w_{ki} = 1$ wird eine solche feste Größe für die Gewichte des Neurons festgelegt. Der Gewinner stärkt somit seine internen Gewichte w_{ki} für alle aktiven Inputs i um den Faktor $\Delta w_{ki} = \eta(x_i - w_{ki})$ auf Kosten der nicht aktiven Inputs. Er verteilt seine Gewichte so um, daß er beim nächsten Mal auf diesen Input noch besser reagiert, auf andere Inputs aber nicht mehr anspricht. In dén verlierenden Neuronen finden im übrigen keine Veränderungen statt.
Mit diesem Verfahren kann man also eine Klassifizierung von Mustern vornehmen, deren Klasseneinteilung unbekannt ist. Die Neuronen lernen, sich auf spezielle Inputmuster zu spezialisieren. Sie können also Eingabemuster wiedererkennen und ähnliche Eingabemuster klassifizieren.

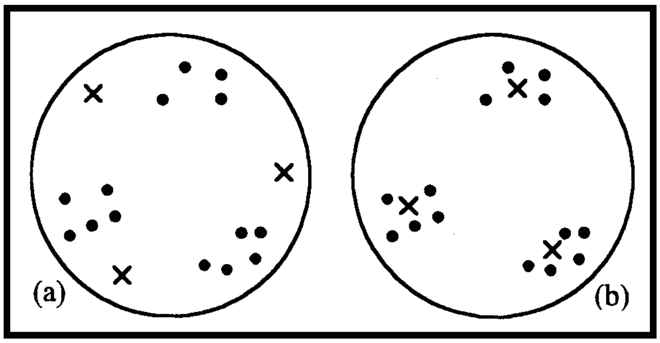


Abb.3.2: Klassifizierung durch konkurrierendes Lernen

Das bedeutendste Beispiel für dieses Lernverfahren ist das der selbstorganisierenden Karten SOM nach Kohonen. Hierauf wird noch kurz im folgenden Abschnitt eingegangen (s. auch Folien zum Vortrag).

3.2 Lernparadigmen

3.2.1 Überwachtes Lernen (supervised learning)

Überwachtes Lernen, das Lernen mit Lehrer, ist dadurch definiert, daß es einen externen Lehrer gibt, der Wissen über die Umwelt des Neuronalen Netzes mitbringt. Dieses Wissen des Lehrers zeigt sich in den Input-Output-Beispielen. Hierbei hat der Lehrer nun die Fähigkeit dem Netz für einen bestimmten Input die optimale Ausgabe zu präsentieren. Ein Beispiel für diese Art des Lernens haben wir in 3.1.2 gesehen. Hier lernt das Netz durch die Minimierung einer Fehlerfunktion, die durch den Unterschied zwischen Referenzausgabe und reale Ausgabe definiert ist. Mit diesem Verfahren wird also versucht das Wissen des Lehrers auf dem Netz zu emulieren. Nun gibt es zwei Möglichkeiten "supervised learning"zu implementieren: als Online- bzw. Offline-Lernen. Bei der Offline-Implementierung wird eine zusätzliche Logikeinheit benötigt, die das Netz trainiert. Nach dem erfolgreichen Lernen werden die Gewichte des Netzes festgehalten, und das Netz arbeitet dann ohne jegliche Gewichtsänderung. Beim Online-Lernen wird die Logikeinheit mit ins Netz eingebaut, das Lernen funktioniert also in Echtzeit. Das Netz bleibt also während des gesamten Betriebes dynamisch. Es kann sich, im Gegensatz zum Offline-Fall, an eine sich ändernde Umwelt anpassen.
Beiden Verfahren liegt aber die Schwäche zugrunde, daß ein Lehrer vorhanden sein muß. Falls das Wissen um die Umwelt nicht vorhanden ist, muß ein anderes Verfahren angewandt werden.

3.2.2 Lernen durch Verstärkung (reinforcement learning)

> „*Wenn auf eine Aktion eines Lernsystems irgendein positiver Zustand folgt, wird die Tendenz des Systems, diese bestimmte Aktion erneut hervorzurufen, verstärkt. Andernfalls wird die Tendenz abgeschwächt.* "

Diese Aussage charakterisiert das "reinforcement learning". Bei diesem Verfahren wird nicht eine gewünschte Ausgabe angegeben, an dem das Netz lernen kann, sondern die Ausgabe wird mit richtig oder falsch, eventuell noch graduell bewertet. Dieses Lernen durch Verstärkung ist ein Online-Verfahren, welches ohne Lehrer auskommt und bei dem das Netz in einem "*trial and error*"Verfahren, sich seiner Umwelt anzupassen versucht. Grundprinzip ist, daß das Netz versucht, in Abhängigkeit eines globalen Verstärkungssignals seine Gewichte so zu ändern, daß die erwartete Verstärkung durch das Lernen erhöht wird. Dies Verfahren ist neurobiologisch bzw. evolutionär plausibler, weil man einfache Rückkopplungsmechanismen der Umwelt (Bestrafung bei falschen Entscheidungen, Belohnung bei richtigen) bei niederen und höheren Lebewesen beobachten kann. Das Problem hierbei besteht nun darin, daß das Lernen auf Grund der Unwissenheit der richtigen Ausgaben länger dauern kann, da weniger Informationen über die korrekte Modifikation der Gewichte zur Verfügung steht. Ein sehr ausführliches und ansprechendes Buch mit Beispielen zu diesem Thema findet man bei Sutton ([Sutton]).

3.2.3 Nichtüberwachtes Lernen

Bei diesem Verfahren wird dem Netz während des Trainings nur die Menge der Eingabemuster angeboten und keine Information bezüglich erwünschter Ausgabe oder über die Qualität der Ausgabe. Der Algorithmus muß nun selbständig versuchen Gruppen (cluster) ähnlicher Eingabevektoren zu identifizieren und diese auf Gruppen ähnlicher oder benachbarter Neuronen abzubilden. Bei den selbstorganisierenden Karten von Kohonen, einem Vertreter dieses unüberwachten Lernverfahrens, werden durch das Lernverfahren die Gewichtsvektoren des Neurons, das zum Eingabevektor am ähnlichsten ist, und seiner Nachbarneuronen dem jeweils anliegenden Eingabevektor noch ähnlicher gemacht. Das Ziel ist es, daß im trainierten Zustand der selbstorganisierenden Karte ähnliche Eingabevektoren auf topologisch benachbarte Neuronen abgebildet werden. Dadurch, daß dieses Verfahren ähnliche Vektoren in ähnliche Klassen gruppiert, kann man die mit ihnen trainierten neuronalen Netze auch wieder zur Klassifikation einsetzen, für die man eigentlich überwachte Lernverfahren verwendet. Man trainiert hierfür nur die Karte durch eine Menge von bekannten Referenzvektoren, deren Abbildung auf die Neuronen bekannt ist, und dann auch unbekannte Muster nach ihrer Nähe zum nächsten Referenzvektor klassifizieren kann.

Kapitel 4

Beispiele für Neuronale Netze

In diesem Kapitel werden zwei gebräuchliche Netztypen dargestellt, die im nächsten Kapitel benutzt werden, um die Formel von Black-Scholes mit Hilfe eines Neuronalen Netzes zu emulieren. Bei beiden Typen handelt es sich um mehrschichtige feedforward Netze.

4.1 RBF-Netze - Radiale-Basisfunktionen-Netze

Diese Art von Netzen mit nur einer Schicht verdeckter Neuronen hat ihren Namen dem Umstand zu verdanken, daß eben diese Neuronen der verdeckten Schicht spezielle, radialsymmetrische Aktivierungsfunktionen besitzen (Abb.4.1). Mathematisch gesehen sind diese Funktionen Basisfunktionen eines Funktionensystems zur Approximation von mehrdimensionalen Funktionen anhand von Werten der Stützstellen. Ein Vorteil dieser Netze ist, daß die Aktivierungsfunktion nur große Werte liefert, wenn sich die Eingabemuster in der Nähe einer Stützstelle befinden, ansonsten liefert das Netz geringe Aktivität. Der andere Vorteil besteht darin, daß die einfache Struktur mit nur einer Schicht verdeckter Neuronen es erlaubt, die Gewichte des Netzes direkt zu errechnen. Neben dieser Initialisierung läßt sich das RBF-Netz noch z.B. mit Backpropagation nachtrainieren.

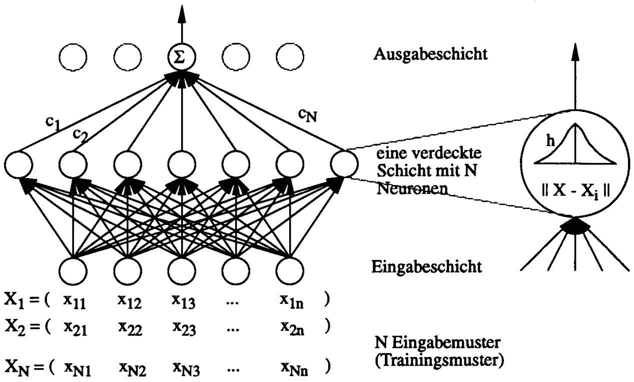


Abb.4.1: Schema eines RBF-Netzes

Ursprünglich sind RBFs zur Interpolation genutzt worden, also zum exakten *Fitten* einer Funktion durch einen Satz von Punkten. Verallgemeinert werden RBFs nun zur Approximation von (auch) nichtlinearen Funktionen f : $\Re^d \rightarrow \Re$ eingesetzt. Die mathematische Beschreibung lautet:

$$y_l = f(\mathbf{x}_l) + \epsilon_l \wedge E\left[\epsilon_l | \mathbf{x}_l\right] = 0. \tag{4.1}$$

(Interpolationsansatz ohne ϵ)

Über einen Variationsansatz kommt man zu dem folgenden Funktional, was durch die gesuchte Funktion f minimiert wird:

$$H[\hat{f}] = \sum_{i=1}^{T} \left(\|\hat{y}_l - \hat{f}(\mathbf{x}_l)\|^2 + \lambda \|P\hat{f}(\mathbf{x}_l)\|^2 \right) \tag{4.2}$$

Hierbei ist P ein Differentialoperator, der als Stabilisator des Variationproblems wirkt, λ der Variationsparameter und $\|...\|$ eine Norm des Funktionsraumes in dem f liegt. Nach Lösung der Euler-Lagrangschen Gleichungen für das Funktional H[f] bekommt man einen allgemeinen Lösungsansatz für die Funktion f von folgender Form:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{k} c_i h_i(\|\mathbf{x} - \mathbf{z}_i\|) + p(\mathbf{x}). \tag{4.3}$$

Hier sind $\mathbf{z}_i$ d-dimensionale Zentren (die Stützstellen im Interpolationsmodell, die im Netzbild die Gewichte des Netzes sind), c_i skalare Koeffizienten, h_i skalare Funktionen, p() Polynome.

Betrachtet man nur den konstanten und linearen Teil des Polynoms mit dessen Koeffizienten und W als die d*d Gewichtsmatrix erhält man:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{k} c_i h_i\left((\mathbf{x} - \mathbf{z}_i)WW(\mathbf{x} - \mathbf{z}_i)\right) + \alpha_0 + \alpha_\mathbf{1}\mathbf{x}. \tag{4.4}$$

Passende Wahl für eine Basisfunktion ist nun die Gauß-Funktion $e^{-\frac{x}{\sigma^2}}$. Bei dieser Wahl bekommt man optimale Ergebnisse der Approximation auch ohne den Polynomanteil. Bei der Wahl der Zentren entscheidet man sich einfach für eine Teilmenge der Trainigsmuster, in der Hoffnung, daß diese Teilmenge den ganzen Bereich einigermaßen abdeckt. Da die Lage der Zentren aber entscheidend für die Approximation ist, kann man bei ungünstig gewählten Zentren diese durch kleine Schritte verschieben und so noch zu einem guten Ergebnis kommen. Diese Zentren, also im Bild des Netzes die Gewichte der verdeckten Schicht, können neben diesen berechenbaren Einstellungen auch durch iterative Verfahren wie beim Backpropagation mit Gradientenabstieg nachtrainiert werden.

4.2 MLP - Multi-Layer-Perceptron

Diese Klasse von Netzen ist wohl die bekannteste. Die drei Haupteigenschaften sind:

- Die Neuronen besitzen nichtlineare Ausgabefunktionen wie die sigmoide Funktion
- Es besitzt neben der Input- und der Outputschicht eine oder mehrere verdeckte Schichten
- Die MPLs sind im allgemeinen stark vernetzt

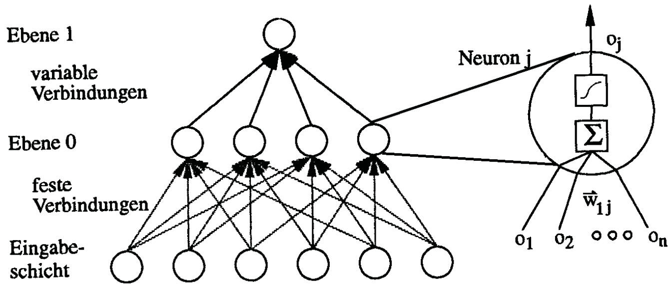


Abb.4.2: Schema eines Multilayerperceptrons

In (Abb. 4.2) ist die Architektur eines MLPs mit einer verdeckten Schicht dargestellt. Um in der Sprache des Abschnitts vorher zu bleiben, kann man eine Formel für solch ein MLP aufstellen:

$$\hat{f}(\mathbf{x}) = h\left(\sum_{i=1}^{k} \delta_i h(w_{0i} + \mathbf{w}_{1i}\mathbf{x}) + \delta_0\right). \tag{4.5}$$

Hier sind h()die sigmoide Funktion $\frac{1}{1+e^{-x}}$, δ_0 (w_{0i}) Schwellenwerte des Ausgabeneurons (des i. verdeckten Neurons), $\delta_{1..k}$ ($\mathbf{w}_{1i}$) die Gewichte des Ausgabeneurons (der Gewichtsvektor des i. verdeckten Neurons). Somit hat man die Möglichkeit, nach dieser Formel wieder eine Funktion zu approximieren. Trainiert werden diese Netze mit Hilfe des Backpropagationverfahrens (s. Formel 3.8).

Kapitel 5

Eine Anwendung: Black-Scholes Formel

In diesem Kapitel soll als Anwendung von Neuronalen Netzen gezeigt werden, daß man mit relativ einfachen Netzstrukturen bei dem Versuch faire Optionspreise durch Neuronale Netze berechnen zu lassen, schon bemerkenswert gute Ergebnisse erzielt. Der faire Preis einer Call-Option ist nach der Formel von Black-Scholes nun:

$$G(S(t), t) = S(t)N(d_1) - Ee^{-r(T-t)}N(d_2) \tag{5.1}$$

(aktueller Aktienpreis S(t), Erfüllungspreis E, Zeitraum der Option T-t, N(x) ist ℵ(0,1)-Gauss-Verteilung)

$$d_1 = \frac{\log(\frac{S(t)}{E}) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \tag{5.2}$$

$$d_1 = \frac{\log(\frac{S(t)}{E}) + (r - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \tag{5.3}$$

Diese Formel ist der erste analytische Ansatz gewesen einen fairen Preis für eine Option zu berechnen. Diese Formel wird im weiteren als Vergleichsobjekt für die Güte der Neuronalen Netze gebraucht.

5.1 Monte-Carlo-Simulation

Es wird nun versucht, Neuronalen Netzen die Formel von Black-Scholes lernen zu lassen, um zu sehen, ob es prinzipiell möglich ist, den fairen Preis einer Option lernen zu können. Die Aufgabe, die Formel von Black-Scholes zu approximieren, welche eine nichtlineare

Funktion ist, ist nach den Betrachtungen des letzten Kapitels prädestiniert für Neuronale Netze. Die Frage ist nun wie komplex die Netze sein müssen (wieviele Neuronen in der verdeckten Schicht sein müssen), um ein genügend zufriedenstellendes Ergebnis zu erzielen. Um hierzu eine Abschätzung machen zu können, schafft man sich mit Hilfe einer Monte-Carlo-Simulation Trainingsdaten. Diese Trainingsdaten sind Tagesaktienpreise für zwei Jahre und hierauf aufbauend Optionspreise. Die Aktienpreise sind mit den Bedingungen über das stochastische Verhalten von Aktien (s. Vortrag zu Black-Scholes von Tobias Galla) und den typischen Eigenschaften einer New York Stock Exchange (NYSE) Aktie erzeugt worden. Die Opionsscheinpreisverläufe wurden dann hierauf mit Hilfe der Formel von Black-Scholes und den Eigenschaften von Optionen nach dem Chicago Board Options Exchange (CBOE) erzeugt (über Details informiere man sich in [Poggio]).
Als Netzstrukturen nutzt man nun MLP und RBF-Netze. Hierbei werden die MLPs mit Hilfe eines online Backpropagationverfahrens und die RBF-Netze mit einem Levenberg-Marquard Verfahren im batch mode trainiert. Als Input bekommen die Netze nun $\frac{S(t)}{E}$ und T-t. Diese beiden Eingaben reichen aus, da man die Annahme macht, daß der Aktienkursverlauf unabhängig von der absoluten Höhe des Aktienpreises ist und der faire Preis der Aktie homogen vom Grade eins in S(t) und E ist. Ohne hier auf Details einzugehen und mehrere Ergebnistabellen und Diagramme zu zeigen (s. [Poggio]), soll hier nur das grundsätzliche Ergebnis dargestellt werden. Bei einer sehr einfachen Netzstruktur, bestehend aus vier Neuronen in der verdeckten Schicht bei den beiden betrachteten Netzarten, bekommt man schon eine sehr gute Approximation der Formel von Black-Scholes. Auf diese Aussage soll nicht weiter quantitativ eingegangen werden, nur insoweit, wenn man auf betrachteten Testpfaden Delta-Hedging vornimmt, die Ergebnisse der Neuronalen Netze im Vergleich zu hedgen, mit der Formel von Black-Scholes bis zu 36 Prozent der betrachteten Netze zu besseren Ergebnissen zu kommen. Die Aussage ist also, daß Neuronale Netze auf künstlich erzeugten Aktiendaten die Formel von Black-Scholes gleichwertig ersetzen können.

5.2 S&P500 Daten - Test

Im letzten Abschnitt ist qualitativ gezeigt worden, daß Neuronale Netze schon mit sehr einfacher Struktur, die Formel von Black-Scholes hinreichend genau approximieren können. Diese Aussage ist auf Grundlage von Tests auf künstlichen Daten getroffen worden. Hier soll nun die Aussage auf echte historische Aktien- und Optionspreisverläufe ausgedehnt werden (Details s. [Poggio]). Dabei verwendet man Daten aus den Jahren 1987 bis 1991. Man geht nun folgendermaßen vor: diese 5 Jahresperiode wird in 10 mal 6 Monateperioden unterteilt. Man trainiert die Netze wie oben nun auf eine Halbjahresperiode und testet dann auf die Daten der folgenden Halbjahresperiode mit Delta-Hedging. Dies geschieht zum einen unter Zuhilfenahme von Black-Scholes und zum anderen mit Hilfe der Neuro-

nalen Netze. Die Formel von Black-Scholes benötigt als Parameter noch die Volatilität, welche aus historischen Daten extrahiert wird. Auch an dieser Stelle soll nur das Ergebnis plakativ dargestellt werden. Es zeigt sich, daß das Hedging mittels Neuronaler Netze dem Hedging durch Black-Scholes bei vielen Testpfaden sogar überlegen ist. Hierzu kann man wieder Details in [Poggio] finden. Ein Vorteil der Neuronalen Netze liegt noch darin, daß keine Paramter, wie die Volatilität, bestimmt werden müssen. Diese extrahiert sich das Netz durch das Training selbständig aus den Daten. Man muß hier jedoch anführen, daß die Neuronalen Netze von Zeit zu Zeit nachtrainiert werden müssen, um die den Daten unterlegende Verteilung zu erfassen. Dies ist auch der Grund, daß immer Trainings- und Testpfad aufeinanderfolgende Perioden sind.
Die Ergebnisse dieses Kapitels sollen in ihrer doch sehr plakativen Art und Weise zeigen, daß es sinnvoll sein kann, Neuronale Netze zur Bestimmung z.B. eines fairen Optionspreises einzusetzen. In dem Paper von [Poggio] erfährt man hierzu genauere Ergebnisse, die diese Aussage auch quantitativ ein wenig mehr belegen. Dennoch wird ganz deutlich, daß in diesem Bereich der Forschungsbedarf noch sehr notwendig ist, jedoch in Anbetracht der Ergebnisse Grund zur Hoffnung besteht, diese Methoden sinnvoll einsetzen zu können.

Literaturverzeichnis

[Fischer, 1998] Jörn Fischer: Strategiebildung mit Neuronalen Netzen; Diplomarbeit im Fach Physik WWU Münster

[Haykin] S. S. Haykin: Neuronal Network - A comprehensive foundation; Prentice Hall ISBN: 0132733501

[Hertz, 1991] J. Hertz, A. Krogh, R. G. Palmer: Introduction to the theory of neural computation; Addison-Weslay Publishing Company

[Kohonen] T. Kohonen: Self Organizing Maps; Springer Verlag

[Poggio, 1994] J. M. Hutchinson, A. Lo, T. Poggio: A nonparametric approach to pricing and hedging derivative securities via learning networks; MIT Paper C.B.C.L. No. 92

[Rojas, 1996] Rojas: Theorie der neuronalen Netze; Springer 1996, Berlin

[Seminar] Ausarbeitungen zum Seminar (SS 97): Einführung in Künstliche Neuronale Netze und Fuzzy-Systeme; Prof. Dr. Lippe, Institut für Informatik WWU Münster

[Sutton] Richard S. Sutton, Andrew G. Barto: Reinforcement Learning; MIT Press

[Zell, 1997] A. Zell: Simulation neuronaler Netze; R. Oldenbourg Verlag