Modellierung von Preiserwartungen durch neuronale Netze

Maik Heinemann*
Carsten Lange**

Diskussionspapier Nr. 203
April 1997

Zusammenfassung

Der Beitrag untersucht, wie Erwartungsbildung mit Hilfe neuronaler Netze modelliert werden kann. Die Grundlage bildet ein Cobweb-Modell, in dem Firmen Preiserwartungen auf Basis eines Feedforward-Netzes bilden.

Zunächst wird anhand von Simulationen gezeigt, daß Firmen durch neuronale Erwartungsbildung approximativ rationale Erwartungen bilden können. Im Gegensatz zur Hypothese rationaler Erwartungen ist dafür die Kenntnis des relevanten Modells nicht mehr erforderlich, lediglich Beobachtungen des Marktpreises und der ihn beeinflussenden Variablen vergangener Perioden werden benötigt.

Abschließend erfolgt eine allgemeine Analyse neuronalen Lernens. Es ist möglich, Bedingungen dafür abzuleiten, daß die neuronalen Erwartungen der Firmen asymptotisch mit rationalen Preiserwartungen übereinstimmen.

JEL–Klassifikation: C 45, D 83.

*Universität Hannover, Institut für Volkswirtschaftslehre / Wachstum & Verteilung, Königsworther Platz 1, D–30167 Hannover. E–Mail: heinemann@vwl.uni-hannover.de

**Universität Hannover, Institut für Volkswirtschaftslehre / Geld, Kredit, Währung, Königsworther Platz 1, D–30167 Hannover. E–Mail: lange@gkw.uni-hannover.de

1. Einführung

Neuronale Netze erfreuen sich im Rahmen der ökonomischen Forschung inzwischen einer recht großen Beliebtheit. Insbesondere werden sie für empirische Fragestellungen, wie zum Beispiel für die Prognose von Zeitreihen verwendet.[1] Der Grund hierfür ist wohl in erster Linie darin zu sehen, daß Verfahren wie den neuronalen Netzen, die im Rahmen der Forschungen zur künstlichen Intelligenz entwickelt wurden, zugetraut wird, Wirkungszusammenhänge zwischen Variablen zu „erlernen", ohne daß hierfür irgendwelche Vorinformationen bezüglich des tatsächlich zugrundeliegenden Zusammenhangs erforderlich sind. Zwar ist ein solcher Anspruch an die Leistungsfähigkeit neuronaler Netze überzogen, und es existieren auch eine ganze Reihe von Arbeiten (z. B. Arminger (1994)), die zeigen, daß neuronale Netze wie ökonometrische Modelle aufgefaßt und auch formal als solche behandelt werden können. Dennoch ist vermutlich gerade die neuronalen Netzen zugeschriebene Fähigkeit, lernen zu können, eine Erklärung für die Faszination, die von ihnen ausgeht.

Der vorliegende Beitrag knüpft an dieser Stelle an und geht der Frage nach, inwieweit neuronale Netze geeignet sind, die Erwartungsbildung von Wirtschaftssubjekten zu modellieren. Ausgangspunkt hierbei ist die Hypothese rationaler Erwartungsbildung. Das Bilden rationaler Erwartungen setzt seitens der betrachteten Wirtschaftssubjekte ein außerordentliches Vorwissen bezüglich der ökonomisch relevanten Umwelt voraus, schließlich handelt es sich hierbei um „. . . *predictions of the relevant economic theory*".[2] Gerade dieser Umstand jedoch begründet die vielerorts geäußerte Kritik an der Hypothese rationaler Erwartungsbildung. Wenn nun neuronale Netze tatsächlich in der Lage sind, Zusammenhänge zwischen ökonomischen Variablen auf der Grundlage von Beobachtungen dieser Variablen zu erlernen, dann müßten doch Wirtschaftssubjekte, die ein neuronales Netz zur Bildung ihrer Erwartungen verwenden, Erwartungen bilden, die rationalen Erwartungen zumindest sehr ähnlich sind. Im Gegensatz zur Hypothese rationaler Erwartungen müßte nun allerdings nicht mehr vorausgesetzt werden, den betrachteten Wirtschaftssubjekten sei das relevante Modell bekannt — sie müßten lediglich wissen, welche ökonomischen Variablen für den fraglichen Zusammenhang relevant sind.[3]

Grundlage der folgenden Ausführungen ist ein an den Cobweb-Fall angelehntes Modell, in dem Firmen zur Bestimmung ihrer Angebotsmenge den zukünftigen Marktpreis

[1] Eine Übersicht über einige Anwendungsgebiete neuronaler Netze findet sich in Murtagh (1990), Würtz und Murtagh (1991), Blien und Lindner (1993) sowie Bol, Nakhaeizadeh und Vollmer (1994).

[2] Muth (1961, S. 316).

[3] Die Idee, daß Wirtschaftssubjekte zur Erwartungsbildung Hilfsmodelle verwenden, deren Parameter sie dann auf der Grundlage ihrer Beobachtungen schätzen, ist nicht neu und wird insbesondere vom *„bounded rational learning approach"* verfolgt (vgl. Bray (1983), Marcet und Sargent (1989) und Sargent (1993)). Von diesen Ansätzen unterscheidet sich das vorliegende Papier dadurch, daß als Hilfsmodell ein neuronales Netz verwendet wird.

prognostizieren müssen. Da die Firmen das relevante Modell nicht kennen, verwenden sie ein neuronales Netz, das den Zusammenhang zwischen einer stochastischen Umweltvariablen und dem resultierenden Marktpreis abbilden soll.

Anhand von Simulationen wird der Lernalgorithmus des verwendeten neuronalen Netzes erläutert und gezeigt, daß Firmen durch neuronale Erwartungsbildung approximativ rationale Erwartungen bilden können. Hierzu benötigen sie lediglich die Beobachtungen des Marktpreises und der ihn beeinflussenden Umweltvariablen vergangener Perioden — nicht jedoch Informationen über die Struktur des Modells. Dabei wird berücksichtigt, daß der Marktpreis auch durch Erwartungsfehler der Firmen beeinflußt wird (Forecast-Feedback).[4]

Anschließend wird der Frage nachgegangen, ob sich auch über Simulationen hinausgehende Aussagen über die Eigenschaften neuronalen Lernens gewinnen lassen. Es wird gezeigt, daß die asymptotischen Eigenschaften eines auf einem neuronalen Netz beruhenden Lernvorgangs mittels eines speziellen Differentialgleichungssystems beschrieben werden können. Dies ermöglicht es, Bedingungen zu formulieren, unter denen den Firmen asymptotisch das Lernen rationaler Preiserwartungen möglich ist. Ein Beispiel illustriert abschließend die formalen Ausführungen.

2. Neuronale Erwartungsbildung am Beispiel eines Cobweb-Modells

2.1. Das Modell

Untersucht wird ein Markt mit vollkommener Konkurrenz, mit einem Kontinuum völlig identischer gewinnmaximierender Firmen im Intervall $[0, 1]$.[5] Die in Periode t entstehenden Produktionskosten (k_t) sind abhängig von der in Periode t produzierten Menge (x_t) und von einem Bündel Umweltfaktoren, die durch die Variable u_t wiedergegeben werden und von denen alle Firmen gleichermaßen betroffen sind. Mögliche Determinanten der Umweltvariablen u_t könnten z. B. die Höhe der Tarifabschlüsse, die Steuerbelastung oder die Höhe der Sozialabgaben sein. Im folgenden wird davon ausgegangen, daß u_t in allen Perioden gleichverteilt im Intervall $(0, 1]$ ist.

Der Einfluß der für den Markt produzierten Menge und der Umweltvariablen auf die

[4] Um die im folgenden erläuterten Simulationen detailliert nachvollziehbar zu machen, können die entsprechenden Simulationen via Internet von den Autoren bezogen werden (vgl. Heinemann und Lange (1997)). Die Simulationen stehen als Mathematica 2.2 Notebook sowie als ausführbare Datei zusammen mit dem zugrundeliegenden Delphi 2.0 Quellcode zur Verfügung.

[5] Aus der Annahme eines Kontinuums repräsentativer Firmen im Intervall [0,1] folgt, daß nicht zwischen aggregierten und einzelwirtschaftlichen Größen unterschieden werden muß. Zum Beispiel folgt mit x_t als aggregiertem Angebot und $\widetilde{x}_{i,t} = \widetilde{x}_t$ als einzelwirtschaftlichem Angebot der Firma i, daß $x_t = \int_0^1 \widetilde{x}_t \, di = \widetilde{x}_t$.

Produktionskosten wird durch eine quadratische Kostenfunktion modelliert:

$$k_t = \frac{1}{2} u_t x_t^2 \quad 0 < u_t \leq 1 \quad \text{mit: } \frac{\partial k_t}{\partial x_t} = u_t x_t \tag{1}$$

Die Produktionsmenge x_t wird von den Firmen gesetzt. Auf die exogene Variable u_t haben sie dagegen keinen Einfluß. u_t ist allerdings allen Firmen bekannt, bevor die Produktionsentscheidung getroffen werden muß. Die exogenen Einflüsse gehen folglich in die Produktionsentscheidung ein.

Weil von vollständiger Konkurrenz ausgegangen wird, werden die gewinnmaximierenden Firmen ihre Produktionsmenge so setzen, daß der erwartete Marktpreis[6] (p_t^e) ihren Grenzkosten entspricht. Damit entsprechen die aggregierten Grenzkosten gemäß (1) dem von allen Firmen gleichermaßen erwarteten Preis, und (2) gibt die aggregierte Angebotsfunktion wieder:

$$x_t = \frac{p_t^e}{u_t} \tag{2}$$

Die gesamtwirtschaftliche Nachfrage sei wie folgt vom Marktpreis (p_t) abhängig:[7]

$$x_t = 75 - 3p_t - \frac{1}{2} p_t^2 \tag{3}$$

Die Angebotsfunktion (2) und die Nachfragefunktion (3) implizieren, daß der markträumende Preis (p_t), durch die Umweltvariable (u_t) und den erwarteten Preis (p_t^e) determiniert wird:[8]

$$p_t = M(p_t^e, u_t) = -3 + \frac{\sqrt{159u_t^2 - 2p_t^e u_t}}{u_t} \quad \text{für: } p_t^e < 75u_t \tag{4}$$

Bildeten die Firmen rationale Erwartungen im Sinne perfekter Voraussicht, dann wären der erwartete und der markträumende Preis identisch. Mit p_t^* als Preis im rationalen Erwartungsgleichgewicht folgt (5) als rationale Preiserwartungsfunktion:

$$p_t^* = R(u_t) = -\left(\frac{3u_t + 1}{u_t}\right) + \sqrt{\left(\frac{3u_t + 1}{u_t}\right)^2 + 150} \quad \text{für: } p_t^e = p_t = p_t^* \tag{5}$$

[6] p_t^e ist hier als der für die Periode t erwartete Preis zu verstehen. Die Erwartungsbildung erfolgt zu Beginn der Periode t.

[7] Die Nachfragefunktion ist nichtlinear gewählt, um im folgenden zu zeigen, daß neuronale Netze auch für nichtlineare Problemstellungen einsetzbar sind. Die quadratische Form ist gewählt, um die Rechenbarkeit des Modells zu gewährleisten. Empirische Relevanz wird nicht postuliert.

[8] Es wird nur die erste Lösung des Gleichungssystems (2) und (3) dargestellt, weil die zweite Lösung im Bereich reeller Zahlen ausschließlich negative Preise generiert. Damit durch die dargestellte erste Lösung keine negativen oder konjugiert komplexe Preise generiert werden, ist es nötig, den Definitionsbereich von p_t^e einzuschränken.

Rationale Erwartungen können die Firmen jedoch nur dann bilden, wenn ihnen das Modell vollständig bekannt ist. Dies ist aber in vielen Fällen eine unrealistische Annahme. Deshalb wird im folgenden gezeigt, wie die Firmen mit Hilfe neuronaler Erwartungsbildung, die für eine korrekte Produktionsentscheidung benötigte rationale Preiserwartungsfunktion (5) nahezu perfekt approximieren können, obwohl sie weder den Funktionstyp noch die Parameter der Nachfragefunktion kennen. Die Firmen besitzen nur Modellkenntnisse darüber, daß der sich am Markt einstellende Preis p_t in irgendeiner Weise durch die Umweltvariable u_t beeinflußt wird.

2.2. Neuronale Erwartungsbildung

Um neuronale Erwartungsbildung im Modell zu berücksichtigen, wird ein Feedforward-Netz als Erwartungsbildungsmodul in das Modell implementiert. Ein Feedforward-Netz ist ein neuronales Netz, das einem oder mehreren Informations-Inputs einen oder mehrere Informations-Outputs zuordnet. Die Informationsverarbeitung erfolgt ausschließlich vorwärts. Rückkopplungen existieren nicht.[9]

Hier wird als Informations-Input der Umwelteinfluß u_i verwendet. Der *Output* des Netzes entspricht dann dem von den Firmen erwarteten Preis p_i^e. Das verwendete Netz ist in Abbildung 1 dargestellt. Es verfügt über eine Inputschicht mit einem Neuron, eine verdeckte Schicht mit drei Neuronen und eine Outputschicht mit einem Neuron.[10]

Der *Input* (u_i) wird in das Netz eingelesen und an das Input-Neuron weitergeleitet. Die *Aktivität* des Input-Neurons (a_i^I) wird mit Hilfe der sogenannten Aktivierungsfunktion aus dem *Input* u_i berechnet. Hier wird für das Input-Neuron die Einheitsfunktion als Aktivierungsfunktion verwendet, so daß sich *Aktivität* und *Input* entsprechen ($a_i^I = u_i$).

Die *Aktivität* des Input-Neurons wird dann als *Ausgangsgröße*[11] des Input-Neurons an jedes Neuron der verdeckten Schicht weitergeleitet und dort mit Hilfe der Gewichte $w^{I,j}$ (j=1, 2, 3) jeweils zum *effektiven Eingang* e_i^j (j=1, 2, 3) der 3 Neuronen der verdeckten Schicht transformiert.

$$e_i^j = w^{Ij} a_i^I \quad \text{für: } j = 1,\ 2,\ 3 \tag{6}$$

Die *effektiven Eingänge* der verdeckten Neuronen werden wiederum mit Hilfe einer

[9] Rückgekoppelte Netze eignen sich besonders für die Mustererkennung (vgl. für einen Überblick Blien und Lindner (1993, S. 507 ff.))

[10] Bei komplexeren Problemen sind Netze mit einer größeren Anzahl Neuronen und unter Umständen mehreren verdeckten Schichten gebräuchlich.

[11] In einigen Netzwerktopologien weichen *Aktivität* und *Ausgangsgröße* der Neuronen voneinander ab, weil zwischen *Aktivität* und *Ausgangsgröße* eine sogenannte Ausgangsfunktion zwischengeschaltet ist, die z.B nur dann eine *Ausgangsgröße* ungleich null generiert, wenn die *Aktivität* einen bestimmten Schwellenwert überschreitet (vgl. Hoffmann (1993, S. 20 ff.)). Hier wird auf die Implementierung von Ausgangsfunktionen verzichtet, bzw. als Ausgangsfunktion die Einheitsfunktion verwendet. *Aktivität* und *Ausgangsgröße* eines Neurons sind immer identisch.

Aktivierungsfunktion in die *Aktivität* der entsprechenden Neuronen umgewandelt. Als Aktivierungsfunktion wird hier die sigmoide Funktion (7) verwendet, wobei q^j einen Lageparameter der Aktivierungsfunktion des Neurons j darstellt:

$$a_i^j = A^j(e_i^j,\ q^j) = \frac{1}{1+\exp(e_i^j + q^j)} \quad \text{für: } j = 1,\ 2,\ 3 \tag{7}$$

Die Aktivierungsfunktion (7) bildet jeden *effektiven Eingang* des Neurons j, unabhängig davon, welcher Wert q^j zugewiesen wird, auf ein Intervall [0, 1] ab. Die so erhaltenen *Aktivitäten* der verdeckten Neuronen werden dann wiederum als *Ausgangsgrößen* an das Output-Neuron weitergeleitet. Zuvor werden sie jeweils mit den dem Output-Neuron vorgeschalteten Gewichten ($w^{j,O}$; $j = 1,\ 2,\ 3$) multipliziert und zum *effektiven Eingang* des Output-Neurons aufsummiert:

$$e_i^O = \sum_{j=1}^{3} w^{j,O} a_i^j \tag{8}$$

Obwohl die zu prognostizierenden *Outputs* im allgemeinen nicht im Intervall $[0,1]$ liegen, ist es unproblematisch, daß die Aktivierungsfunktionen der verdeckten Schicht nur Werte im Intervall $[0,1]$ generieren, da die den Neuronen der verborgenen Schicht nachgelagerten Gewichte ($w^{j,O}$) jeden reellen Zahlenwert annehmen können. Dies ermöglicht

eine Skalierung der *Ausgänge* auf das korrekte Niveau.[12] Außerdem wird hier für das Output-Neuron die Aktivierungsfunktion

$$a_i^O = e_i^O + q^O \tag{9}$$

verwendet. Das heißt, die *Aktivität* des Output-Neurons (a_i^O) unterscheidet sich durch einen konstanten Wert (q^O) vom *effektiven Eingang* (e_i^O) und entspricht dem *Ausgang* des Netzes — der zu prognostizierende Preiserwartung (p_i^e).

Werden die Parameter der Aktivierungsfunktionen $\left(q^1,\ q^2,\ q^3 \text{ und } q^O\right)$ zusammen mit den sechs Gewichten ($w^{I,j},\ w^{j,O}$ für $j = 1,\ 2,\ 3$) im Vektor θ zusammengefaßt,

$$\theta = (q^O, w^{1,O}, q^1, w^{I,1}, w^{2,O}, q^2, w^{I,2}, w^{3,O}, q^3, w^{I,3})$$

und sind alle Elemente des Vektors θ gegeben, so kann durch das Netz jedem Umweltzustand u_i ein erwarteter Preis p_i^e zugeordnet werden. Die Erwartungsbildung der Firmen kann über das neuronale Netz erfolgen. Dies wird auch deutlich, wenn das in Abbildung 1 grafisch dargestellte neuronale Netz algebraisch formuliert wird:[13]

$$p_i^e = N(u_i, \theta) = \sum_{j=1}^{3} w^{j,O} \frac{1}{1 + \exp(q^j + w^{I,j} u_i)} + q^O \tag{10}$$

Mit Hilfe des neuronalen Netzes (10) können nun neuronale Preiserwartungen abgebildet werden. Wie gut dadurch die rationale Preiserwartungsfunktion (5) approximiert werden kann, hängt entscheidend von den Elementen des Gewichtungsvektors θ ab. In diesem Zusammenhang sind die weitreichenden Approximationseigenschaften neuronaler Netze von besonderem Interesse:

Satz 1 (Hornik, Stinchcombe und White (1989)) *Ein neuronales Netz mit einer verdeckten Schicht bestehend aus j Neuronen, für deren Aktivierungsfunktionen $A^j(z)$ gilt, daß* $\lim_{z\to-\infty} A^j(z) = 1$ *und* $\lim_{z\to+\infty} A^j(z) = 0$*, kann jede über einer kompakten Menge definierte stetige Funktion beliebig genau approximieren, sofern das neuronale Netz eine hinreichend große Anzahl verdeckter Neuronen enthält.*

Satz 1 gewährleistet, daß ein neuronales Netz existiert, durch das die rationale Erwartungsfunktion (5) im relevanten Bereich perfekt abgebildet werden kann. Im folgenden Abschnitt 3 soll nun zum einen überprüft werden, ob das hier verwendete Netz genügend verdeckte Neuronen enthält, um die rationale Erwartungsfunktion zumindest approximativ wiederzugegeben; zum anderen soll ein Lernalgorithmus vorgestellt werden, der die entsprechenden Elemente des Vektors θ generiert.

[12] Bei neuronale Netzen, die für komplexe Prognoseaufgaben eingesetzt werden, ist es sinnvoll die *Inputs* und *Soll-Outputs* des Netzes durch geeignete Preprocessing-Verfahren auf ein einheitliches Niveau zu normieren, um die Lernfähigkeit der Netze zu optimieren. Auf diese Problematik wird hier aber nicht abgestellt (vgl. dazu Baun (1994, S. 148 ff. und 179 ff.))

[13] Ein formal ähnlich strukturiertes Netz findet sich bei Blien und Lindner (1993, S. 502 ff.).

3. Lernen mit neuronalen Netzen

Als Lernverfahren wird hier der Backpropagation-Algorithmus verwendet. Formal handelt es sich dabei um ein Verfahren zur Minimierung des Prognosefehlers mittels des Gradientenverfahrens. Seinen Namen erhielt das Verfahren, weil der Prognosefehler ausgehend vom *Output* des Netzes rückwärts über die verdeckte Schicht zum Input-Neuron zurückgeführt wird, wobei die entsprechenden Gewichte und Parameter sukzessive korrigiert werden (vgl. Hoffmann (1993, S. 75 ff.)). Ziel ist es, die Gewichte des Vektors θ so zu bestimmen, daß die rationale Preiserwartungsfunktion (5) durch das neuronale Netz (10) möglichst gut approximiert wird.

Der Backpropagation-Algorithmus soll in zwei Schritten vorgestellt werden. Zunächst wird in Abschnitt 3.1 in einem vereinfachten Szenario davon ausgegangen, daß den Firmen ein Datenset bestehend aus Realisationen der Umweltvariablen u_i und den zugehörigen Preisen im rationalen Erwartungsgleichgewicht p_i^* bekannt ist. Dieses Datenset wird zum Training des neuronalen Netzes verwendet, um die Grundzüge des Backpropagation-Algorithmus zu verdeutlichen.

Danach wird in einem zweiten Szenario im Abschnitt 3.2 berücksichtigt, daß den Firmen solange keine Informationen über Preise im rationalen Erwartungsgleichgewicht vorliegen können, solange durch die neuronale Erwartungsbildung noch Prognosefehler begangen werden. Denn nur der sich tatsächlich einstellende Marktpreis p_i ist von den Firmen beobachtbar. Dieser weicht aber vom Preis p_i^* im rationalen Erwartungsgleichgewicht ab, wenn Prognosefehler erfolgen. Dieser Sachverhalt wird als Forecast-Feedback bezeichnet: Jede Preiserwartung, die — bei gegebenem Umweltzustand u_i — größer (kleiner) als die rationale Preiserwartung gemäß (5) ist, führt dazu, daß die Angebotsmenge größer (kleiner) ist als die zum rationalen Erwartungsgleichgewicht gehörige Angebotsmenge. Dies hat zur Folge, daß am Markt nicht der Preis p_i^*, sondern ein vom Erwartungsfehler beeinflußter, geringerer (höherer) Preis (p_i) zustande kommt.[14] Deshalb wird im Abschnitt 3.2 das neuronale Netz mit solchen, durch Forecast-Feedback beeinflußten Marktpreisen trainiert. Es wird gezeigt, daß auch unter diesen erschwerten, aber realistischen Bedingungen die Preisentwicklung von den Firmen nahezu perfekt antizipiert werden kann.

3.1. Lernen ohne Forecast-Feedback

Wie bereits oben erläutert, wird in diesem Abschnitt davon ausgegangen, den Firmen sei ein Datenset bestehend aus Realisationen der Umweltvariablen u_i und den zugehörigen im rationalen Erwartungsgleichgewicht resultierenden Preisen bekannt. Dieses Datenset besteht aus 20 Datenpaaren (*Trainingsset*) und wurde durch die rationale Preiserwartungsfunktion (5) generiert (vgl. Tabelle 1).

[14] Die Dynamik entspricht in ihren Grundzügen der wohlbekannten Cobweb-Dynamik.

Tabelle 1: Trainingsset gemäß Gleichung (5)

i	1	2	3	4	5	$\cdots$	16	17	18	19	20
u_i	0.05	0.10	0.15	0.20	0.25	$\cdots$	0.80	0.85	0.9	0.95	1.00
p_i^*	3.06	4.86	5.94	6.63	7.11	$\cdots$	8.71	8.76	8.81	8.85	8.88

Der Backpropagation-Algorithmus hat nun die Aufgabe, das neuronale Netz so zu trainieren (die Gewichte des Vektors θ so zu setzen), daß immer dann, wenn ein *Input* (u_i) am Netz anliegt, der gemäß Tabelle 1 zugehörige *Soll-Output* (p_i^*) erzeugt wird.[15]

Zuvor muß das Netz aber initialisiert werden, indem die Anfangswerte des Vektors θ bestimmt werden. Dies geschieht in vielen Fällen durch das Generieren entsprechender Zufallszahlen für alle Elemente des Gewichtungsvektors. Hier werden dagegen die Elemente $q^O = 3$ sowie $w^{1,O} = w^{2,O} = w^{3,O} = 0$ gesetzt, und nur die verbleibenden Elemente des Initialisierungs-Vektors θ^0 werden mit Zufallszahlen aus dem Intervall [-10, 10] belegt:[16]

$$\theta^0 = (3,\ 0,\ 8.842,\ -5.282,\ 0,\ -1.309,\ 7.042,\ 0,\ -1.148,\ 0.528) \tag{11}$$

Auf diese Weise wird der Initialzustand des Netzes ökonomisch interpretierbar, was durch Abbildung 1 verdeutlicht werden kann: Durch die gesetzten Gewichte beträgt der *effektive Eingang* des Output-Neurons ($e_i^O = \sum_{j=1}^{3} 0 \cdot a_i^j$) immer null — unabhängig von der *Aktivität* der verdeckten Neuronen. Durch die Aktivierungsfunktion des Output-Neurons (vgl. Gleichung (9)) wird dann dieser *effektive Eingang* um $q^O = 3$ erhöht. Dadurch nimmt der *Output* des Netzes (die *Aktivität* des Output-Neurons) unabhängig von der *Aktivität* der verdeckten Neuronen und folglich unabhängig vom *Input* des Netzes einen Wert von 3 an. Das Prognoseverhalten des Netzes im Initialisierungszustand ähnelt dem statischer Preiserwartungen.

$$p_i^e = N(u_i, \theta^0) = 3 \quad \text{für alle } u_i \tag{12}$$

Werden nun die Umweltvariablen (u_i) der 20 Datenpaare des Trainingssets nacheinander als *Inputs* am Netz angelegt, so errechnet das Netz, statt der durch die rationale Preiserwartungsfunktion vorgegebenen Preise (vgl. p_i^* in Tabelle 1), in allen Fällen $p_i^e = 3$ als erwarteten Preis. Es werden Erwartungsfehler generiert. Diese Erwartungsfehler bilden

[15] Vgl. für den Lernalgorithmus erläuternde Computer-Simulationen Heinemann und Lange (1997).

[16] Zur Erzeugung der Zufallszahlen wurde die Funktion *Random[Real,{-10,10}]* des Programmpaketes Mathematica 2.2 verwendet. Sie wurde zuvor mit *SeedRandom[456]* initialisiert, um die Reproduzierbarkeit des Zufallsereignisses sicherzustellen.

die Grundlage für die Fehlerfunktion $F(\theta,\ P_t,\ U_t)$:

$$\begin{aligned} F(\theta,\ P_t,\ U_t) &= \frac{1}{t}\sum_{i=1}^{t}\left(N(u_i,\theta)-p_i^*\right)^2 \\ &= \frac{1}{t}\sum_{i=1}^{t}\underbrace{\left(\sum_{j=1}^{3} w^{j,O}\frac{1}{1+\exp(q^j+w^{I,j}u_i)}+q^O-p_i^*\right)^2}_{\left(p_i^e-p_i^*\right)^2} \end{aligned} \tag{13}$$

Dabei steht t für die Länge der Zeitreihe des Trainingssets (hier: $t=20$), U_t für den Vektor der Umwelteinflüsse $(U_t=(u_1,u_2,\ \ldots,\ u_t))$ und P_t für den Vektor, der aus den jeweiligen Umwelteinflüssen gemäß (5) resultierenden Preise $(P_t=(p_1^*,p_2^*,\ldots,p_t^*))$.

Mit Hilfe der obigen Fehlerfunktion kann das Lernziel als Optimierungsproblem formuliert werden:

$$\min_{\theta} F(\theta,\ P_t,\ U_t) \tag{14}$$

Stellt man sich (13) als 11-dimensionales Fehlergebirge (10 Elemente des Gewichtungsvektors θ als exogene Variablen und den Prognosefehler $F(\theta,\ P_t,\ U_t)$ als endogene Variable) vor, so gilt es, θ so zu bestimmen, daß ein Minimum erreicht wird. Der Weg des steilsten Fehlerabstiegs wird in diesem Gebirge durch den negativen Gradienten von (13) bestimmt:

$$\begin{aligned} -\nabla F\left(\theta,\ P_t,\ U_t\right) &= -\frac{2}{20}\sum_{i=1}^{20}\nabla N\left(u_i,\ \theta\right)\left(p_i^e-p_i^*\right) \\ &= \frac{2}{20}\sum_{i=1}^{20}\nabla N\left(u_i,\ \theta\right)\left(p_i^*-N\left(u_i,\ \theta\right)\right) \end{aligned} \tag{15}$$

Da sämtliche Elemente des Vektors θ sowie die des Trainingssets $(U_t,\ P_t)$ bekannt sind, enthält der negative Gradient (15) nur numerische Werte. Wird er mit einem hinreichend kleinen Skalar, der Lernrate λ, multipliziert und anschließend zum Initialisierungs-Vektor (θ^0) des Netzes addiert, so wird der neue Gewichtungsvektor θ^1 bzw. das dadurch modifizierte neuronale Netz einen i.d.R. geringeren Erwartungsfehler generieren. Der hier verwendete Lernalgorithmus kann demzufolge in zwei Schritten dargestellt werden:[17]

1. Ein Initialisierungs-Vektor θ^0 wird vorgegeben.

2. Der Parametervektor θ wird ausgehend vom Initialisierungs-Vektor θ^0 $\overline{n}$-mal iterativ aktualisiert:

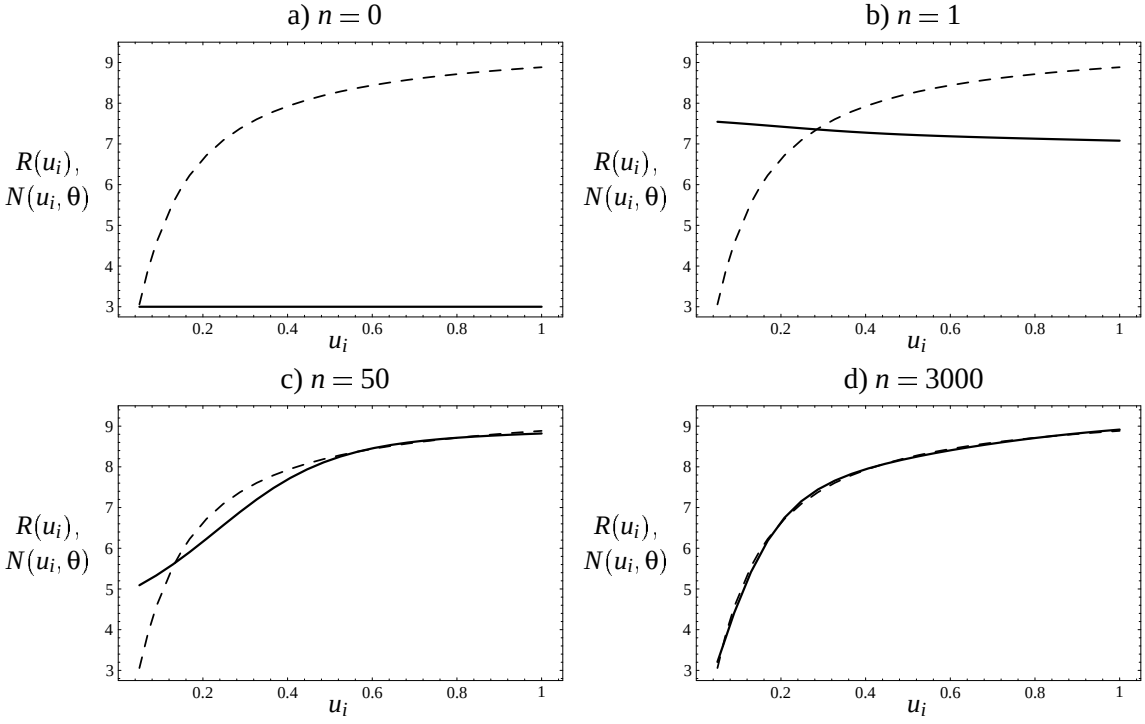
$$\theta^n = \theta^{n-1} - \lambda\nabla F(\theta^{n-1},\ P_t,\ U_t) \quad \text{für: } n=1,\ldots,\overline{n} \tag{16}$$

[17] Der hochgestellte Index gibt an, wie oft der Parametervektor θ aktualisiert wurde.

Sofern $\overline{n}$ groß genug gewählt wurde, sinkt auf diese Weise der Fehler solange, bis approximativ ein lokales Erwartungsfehler-Minimum erreicht wird. Ein weiteres Training des Netzes bringt keine nennenswerten Verbesserungen.

Das Erreichen eines globalen Minimums ist auf diese Weise allerdings nicht garantiert. Sollte der verbleibende Fehler unakzeptabel hoch sein und durch weitere Iterationen nicht nennenswert zu senken sein, so ist zu vermuten, daß ein lokales Minimum auf hohem Fehlerniveau erreicht wurde. Dann muß mit anderen Anfangswerten für θ^0 versucht werden, ein anderes, niedrigeres lokales Minimum oder sogar das globale Minimum zu erreichen.

Abbildung 2: *Verlauf der Trainingsphase* ($\lambda = 0.3$).[a]



[a] Die gestrichelten Kurven zeigen die rationale Preiserwartungsfunktion gemäß (5), die durchgezogenen Kurven zeigen jeweils die vom neuronalen Netz (10) gelernte Preiserwartungsfunktion.

Die Prognosequalität des hier untersuchten Netzes erscheint jedoch hinreichend gut. Wie die Abbildung 2 a – d zeigen, konvergiert die vom Netz gelernte Preiserwartungsfunktion gegen die rationale Erwartungsfunktion (5).

3.2. Lernen mit Forecast-Feedback

Im Abschnitt 3.1 wurde die rationale Erwartungsfunktion (5) durch das neuronale Netz gelernt, nachdem das Netz mit Daten trainiert wurde, die durch diese rationale Preiserwartungsfunktion generiert wurden. Damit wird aber impliziert, daß die bei alternativen Werten der Umweltvariablen im rationalen Erwartungsgleichgewicht resultierenden Preise zu

jeder Zeit von den Firmen beobachtbar sind. Dies ist aber nicht der Fall. Beobachtbar sind lediglich die gewöhnlich durch Prognosefehler beeinflußten Marktpreise. Deshalb soll in diesem Abschnitt überprüft werden, ob die Firmen durch neuronale Erwartungsbildung die rationale Preiserwartungsfunktion (5) auch dann approximativ lernen können, wenn ihnen als Trainingsset nur solche, durch Forecast-Feedback beeinflußten Marktpreise zur Verfügung stehen.[18]

Dazu wird zunächst das neuronale Netz mit den gleichen Gewichten initialisiert, wie im Abschnitt 3.1. Das heißt, es wird für die Ausgangssituation in Periode 1 ein θ_1 unterstellt, das eine konstante Preiserwartung von 3 generiert (vgl. auch Abbildung 2 a). Daran anschließend wird die zum Zufallswert[19] $u_1 = 0.945$ gehörige Preiserwartung mit Hilfe des neuronalen Netzes ermittelt. Der erwartete Preis beträgt, bedingt durch die entsprechende Initialisierung des Netzes, $p_1^e = 3$. Dadurch wird p_1^* unterschätzt (vgl. $p_1^* = 8.844$ gemäß (5) für $u_1 = 0.945$). Die Firmen bieten aufgrund ihrer zu pessimistischen Preiserwartung zu wenig an (vgl. $x_1 = 3.175$ statt 9.359 gemäß (2) für $p_1^e = 3$ bzw. $p_1^* = 8.844$) und der Marktpreis (p_1)überschießt p_1^*. Er ergibt sich gemäß (4) für $u_1 = 0.945$ und $p_1^e = 3$ zu $p_1 = 9.355$. Nur dieser Marktpreis ist in Verbindung mit dem Zustand der Umweltvariable für die Firmen beobachtbar.

Am Ende von Periode 1 wird das neuronale Netz mit Hilfe des in Abschnitt 3.1 vorgestellten Lernalgorithmus und den zur Verfügung stehenden Daten trainiert, um den Parametervektor θ_2 zu generieren. Das Trainingsset besteht nur aus einem Datenpaar ($U_1 = (0.945)$, $P_1 = (9.355)$). In Abbildung 3 a ist das Ergebnis des Lernprozesses grafisch dargestellt($\lambda = 0.3$, $\bar{n} = 500$).[20] Die vom Netz generierte Erwartungsfunktion hat sich zwar so verschoben, daß ein Umweltzustand $u = 0.945$ approximativ eine Preiserwartung von $p^e = 9.355$ erzeugen würde, mit der rationalen Preiserwartungsfunktion hat sie aber noch nicht viel gemein.

Dies hat zur Folge, daß zu Beginn der Periode $t = 2$, für die ein Umweltzustand $u_2 = 0.274$ zufällig generiert wurde, vom Netz eine Preiserwartung von $p_2^e = 9.454$ erzeugt wird.[21] Gleichung (5) und Abbildung 3 a zeigen gleichermaßen, daß damit der im rationalen Erwartungsgleichgewicht resultierende Preis $p_2^* = 7.287$ überschätzt wird, was ein Unterschießen des markträumenden Preises $p_2 = 6.488$ im Vergleich zu p_2^* zur Folge hat.

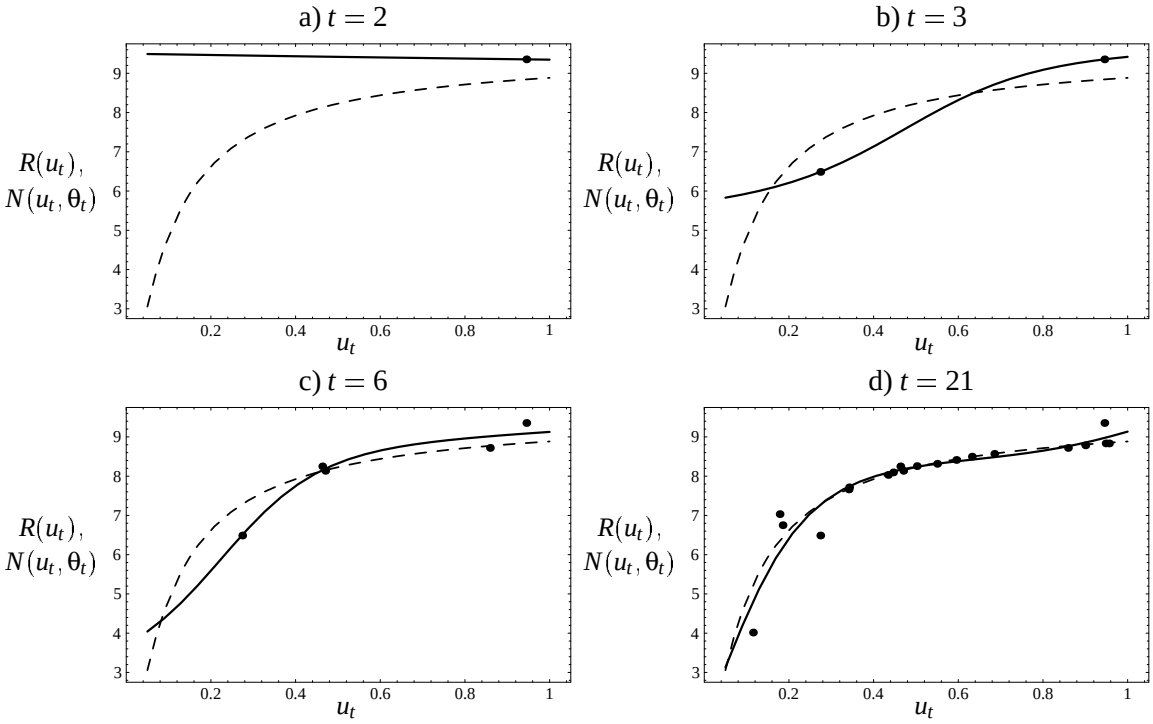
[18] Vgl. für Computer-Simulationen für den Fall des Lernens mit Forecast-Feedback Heinemann und Lange (1997).

[19] Die Zufallswerte für die Umweltvariable wurden mit der Funktion *Random[Real,{0.05,1}]* des Programmpaketes *Mathematica 2.2* erzeugt, die wiederum mit *SeedRandom[456]* initialisiert wurde.

[20] Eine deutlich höhere Lernrate führt zu *Overlearning*, was bedeutet, daß das neuronale Netz das Trainingsset zwar ausgezeichnet approximiert, nicht aber die zu lernende Funktion. Die Problematik des *Overlearning* soll hier aber nicht weiter vertieft werden (vgl. dazu Zimmermann (1994, S. 58 ff.)).

[21] Vgl. $N(u_2, \theta_2)$ aus (10) für $\theta_2 = (7.251, 0.089, 8.841, -5.282, 0.020, -1.309, 7.042, 2.781, -1.402, 0.288)$ und $u_2 = 0.274$

Abbildung 3: *Lernen mit Forecast–Feedback ($\bar{n} = 500 \,/\, \lambda = 0.3$).*[a]



[a] Die gestrichelten Kurven zeigen die rationale Preiserwartungsfunktion gemäß (5), die durchgezogenen Kurven zeigen jeweils die vom neuronalen Netz gelernte Preiserwartungsfunktion zu Beginn der Periode t, also nach $t-1$ Beobachtungen.

Am Ende von Periode 2 können die Firmen auf die Erfahrung von 2 Perioden zurückgreifen. Das Netz wird mit diesen Werten ($U_2 = (0.945, 0.274)$ und $P_2 = (9.355, 6.488)$) trainiert. Ergebnis ist die in Abbildung 3 b dargestellte Preiserwartungsfunktion. Sie approximiert die rationale Preiserwartungsfunktion nur unwesentlich besser. Abbildung 3 c gibt aber schon eine erste Annäherung an die rationale Preiserwartungsfunktion (5) wieder, obwohl das Trainingset lediglich aus fünf Datenpaaren bestand.

Der den Forecast-Feedback berücksichtigende Lernalgorithmus kann für eine beliebige Periode t ($t > 0$) auch formal dargestellt werden. Dabei ist zu berücksichtigen, daß der Parametervektor θ_t sowie die Vektoren der Variablen U_t und P_t am Ende der Periode t bekannt sind. θ_t ist am Ende von Periode $t-1$ ermittelt bzw. für $t = 1$ gesetzt worden und die Elemente der Vektoren P_t bzw. U_t sind am Ende von Periode t beobachtbar. Der Lernalgorithmus arbeitet in 4 Schritten:

1. Der Parametervektor der Periode $t+1$ wird initialisiert, indem ihm der Parameter-

vektor der Periode t zugewiesen wird:[22]

$$\theta_{t+1}^{0} = \theta_t$$

2. Die in Gleichung (16) beschriebene Iteration wird $\overline{n}$-mal ausgeführt:

$$\theta_{t+1}^{n} = \theta_{t+1}^{n-1} + \lambda \nabla F(\theta_{t+1}^{n-1}, P_t, U_t) \quad \text{für: } n = 1, 2, \ldots, \overline{n}$$

3. Dem für die Erwartungsbildung relevanten Parametervektor θ_{t+1} wird der Wert des Parametervektors der letzten Iteration $\theta_{t+1}^{\overline{n}}$ zugewiesen:

$$\theta_{t+1} = \theta_{t+1}^{\overline{n}}$$

4. Wenn der Parametervektor θ_{t+1} ermittelt ist, kann bei gegebenenem Umwelteinfluß u_{t+1} mit Hilfe der Netzfunktion (10) der erwartete Preis p_{t+1}^{e} für Periode $t+1$ bestimmt werden. Dieser führt gemäß Gleichung (4) zum markträumenden Preis p_{t+1} und ergänzt zusammen mit u_{t+1} die Zeitreihen zu P_{t+1} und U_{t+1}. Damit sind die benötigten Werte für Periode $t+1$ bekannt und der Lernalgorithmus kann erneut bei Schritt 1 beginnen.

Dieser Vorgang wurde hier für insgesamt 20 Perioden wiederholt. Abbildung 3 d zeigt das resultierende Netz. Nach 20 Beobachtungen ist zu erkennen, daß sich rationale und neuronale Erwartungen nur unwesentlich unterscheiden.

Es zeigt sich zumindest für den hier untersuchten Fall, daß Wirtschaftssubjekte ohne Kenntnis des funktionalen Zusammenhanges zwischen der zu schätzenden Variablen und der sie bestimmenden Variablen durch ein neuronales Netz Erwartungen bilden können, die approximativ mit rationalen Erwartungen übereinstimmen. Die Kenntnis des gesamten Modells bzw. der *relevanten Theorie* ist nicht notwendig. Es reicht aus, daß die Firmen die aus den exogenen Variablen (u_t) resultierenden Marktpreise (p_t) beobachten können und daß sie einen Zusammenhang zwischen diesen Variablen vermuten.

Weitere Simulationen haben gezeigt, daß die Performance des Netzes noch erheblich verbessert werden kann, z. B. indem weiter zurückliegende Daten entweder ganz gestrichen oder mit geringerem Gewicht in der Fehlerfunktion berücksichtigt werden. Da hier aber die Bedeutung eines neuronalen Netzes für die Erwartungsbildung analysiert werden soll und nicht die Optimierung des Lernalgorithmus, ist aus Gründen der Übersichtlichkeit auf die Implementierung eines komplexeren Lernalgorithmus verzichtet worden.

[22] Der tiefgestellte Index gibt wie auch bei den anderen Variablen die Periodennummer wieder, während der hochgestellte Index anzeigt, wieviele Iterationsschritte des Lernalgorithmus absolviert sind.

4. Asymptotische Eigenschaften neuronalen Lernens

4.1. Formale Analyse

Im vorangegangenen Abschnitt wurde anhand von Simulationen dargestellt, wie Firmen mit Hilfe neuronaler Netze Erwartungen bilden können, die gegen rationale Preiserwartungen konvergieren. Hier sollen über Simulationen hinaus notwendige Bedingungen dafür abgeleitet werden, daß solche Lernprozesse langfristig zu rationaler Erwartungsbildung führen.

Aussagen über die Konvergenz des Vektors θ sind möglich, wenn die im vorangegangenen Abschnitt unterstellte Lernregel geringfügig abgewandelt wird: Es wird nicht mehr unterstellt, daß die Firmen am Ende einer Periode t sämtliche bis dahin erfolgten Beobachtungen der Umweltvariablen und der markträumenden Preise zur Aktualisierung der Gewichte heranziehen, sondern, daß sie lediglich die Beobachtungen u_t und p_t zur Modifikation von θ_t und damit zur Erwartungsbildung in $t+1$ verwenden. Außerdem erfolgt in jeder Periode lediglich ein Iterationschritt zur Anpassung der Gewichte, das heißt im weiteren gilt $\bar{n} = 1$. Darüber hinaus wird die Annahme einer konstanten Lernrate aufgegeben und stattdessen eine im Zeitablauf sinkende Lernrate unterstellt. Diese Modifikationen ermöglichen es, Resultate von Ljung (1977) sowie Kuan und White (1994) heranzuziehen, um die asymptotischen Eigenschaften neuronalen Lernens zu analysieren.

Unverändert wird angenommen, daß der tatsächliche Preis p_t, wie im eingangs formulierten Modell, eine Funktion der Preiserwartungen p_t^e und einer exogenen, beobachtbaren Umweltvariablen u_t ist. Es gilt weiterhin $p_t = M(p_t^e, u_t)$, allerdings wird keine spezielle funktionale Form für die Funktion M unterstellt und die Umweltvariable u_t kann beliebige Werte in einem abgeschlossenen Intervall $[u_u, u_o]$ annehmen.[23]

Die Firmen bilden ihre Preiserwartung p_t^e in Abhängigkeit von der Realisation der Umweltvariablen u_t auf der Grundlage des neuronalen Netzes, so daß $p_t^e = N(u_t, \theta_t)$ gilt. Wird unterstellt, daß das zugrundeliegende neuronale Netz m verborgene Neuronen enthält, besteht der Parametervektor θ_t aus insgesamt $r = 3m+1$ Gewichten. Wie oben erwähnt, wird der Parametervektor θ_t bei der hier betrachteten Lernregel am Ende von Periode t allein in Abhängigkeit vom Erwartungsfehler $p_t - p_t^e = p_t - N(u_t, \theta_t)$ modifiziert. Es gilt:

$$\begin{aligned}\theta_{t+1} - \theta_t &= \lambda_{t+1} \left[\nabla N(u_t, \theta_t) \left(p_t - N(u_t, \theta_t)\right)\right] \\ &= \lambda_{t+1} \left[\nabla N(u_t, \theta_t) \left(M(N(u_t, \theta_t), u_t) - N(u_t, \theta_t)\right)\right] \\ &= \lambda_{t+1} L(u_t, \theta_t)\,, \end{aligned} \tag{17}$$

[23] Alle im folgenden dargestellten Aussagen lassen sich ohne weiteres auf den Fall eines Vektors von Zufallsvariablen übertragen. Darüber hinaus ist es ohne weiteres möglich, eine unbeobachtbare Störgröße ε_t hinzuzufügen, so daß $p_t = M(p_t^e, u_t) + \varepsilon_t$ gilt. Im weiteren wird derartiges aus Gründen der Vereinfachung unterlassen. Vgl. hierzu Kuan und White (1994).

wobei für die Lernrate gilt, daß $\lambda_t = t^{-\kappa}$, $0 < \kappa \leq 1$. Die Funktion $L(u_t, \theta_t)$ in (17) ist nichts anderes als der Gradient des quadrierten, in t begangenen Erwartungsfehlers bezüglich θ_t, das heißt, sie ist analog zu (15) zu interpretieren. Zu beachten ist, daß dem im weiteren betrachteten Lernalgorithmus (17) nunmehr eine sinkende Lernrate unterstellt wird. Dies bedeutet, daß die Modifikationen des Parametervektors θ auch unabhängig von den Erwartungsfehlern immer geringer werden, was eine notwendige Bedingung für die asymptotische Analyse des Lernalgorithmus ist.

Die Frage ist, ob das Lernverfahren (17) asymptotisch zu rationalen Preiserwartungen der Marktteilnehmer führt.[24] Dies ist allerdings nur dann möglich, wenn überhaupt ein θ existiert, so daß $N(u, \theta) = M(N(u, \theta), u)$ für alle $u_u \leq u \leq u_o$ gilt.[25] Es wird daher die folgende Annahme getroffen:

Annahme 1 *Es existiert eine Menge Θ^* r-dimensionaler Parametervektoren, die rationale Preiserwartungen implizieren, das heißt, für alle $\theta \in \Theta^*$ gilt $N(u, \theta) = M(N(u, \theta), u)$ für alle $u_u \leq u \leq u_o$.*

Da aufgrund von Satz 1 bei einer hinreichend großen Anzahl verborgener Neuronen immer ein Netz existiert, das Annahme 1 genügt, bedeutet diese Annahme letztlich, daß das hier zugrundeliegende neuronale Netz insofern korrekt spezifiziert ist, als es eine hinreichende Anzahl verdeckter Neuronen enthält.

Die Frage, ob die Marktteilnehmer asymptotisch rationale Erwartungen bilden, ist gleichbedeutend mit der Frage, ob der Parametervektor θ_t gemäß der Lernregel (17) asymptotisch gegen ein $\theta \in \Theta^*$ konvergiert. Nur wenn dies der Fall ist, werden asymptotisch rationale Erwartungen gebildet.

Nun ist der Lernalgorithmus (17), der die zeitliche Entwicklung des Parametervektors θ beschreibt, ein nichtlineares, nicht-autonomes, stochastisches Differenzengleichungssystem, dessen formale Handhabung nicht unproblematisch ist. Wie jedoch Ljung (1977) zeigt, kann das asymptotische Verhalten von θ_t gemäß (17) unter hier erfüllten Bedingungen durch ein deterministisches Differentialgleichungssystem approximiert werden. Dieses Differentialgleichungssystem ist gegeben durch:

$$\dot{\theta} = \widetilde{L}(\theta) = \mathrm{E}_u \left[L(u, \theta) \right] \tag{18}$$

Die Funktion $\widetilde{L}(\theta)$ entspricht dem Erwartungswert von $L(u, \theta)$ bezüglich u bei gegebe-

[24] Es ist nicht ausgeschlossen, daß multiple rationale Erwartungsgleichgewichte existieren.

[25] Zu beachten ist, daß der entsprechende Parametervektor nicht eindeutig bestimmt ist. Im hier verwendeten Netz (10) weisen alle verdeckten Neuronen identische Aktivierungsfunktionen auf, so daß durch einfache Transformation des Parametervektors eine beobachtungsäquivalente Funktion generiert werden kann. Liegt beispielsweise für ein Netz mit einem verdeckten Neuron ein Parametervektor $\theta_1^* = (q^O, w^{1,O}, q^1, w^{I,1})$ vor, so gilt mit $\theta_2^* = (q^O + w^{1,O}, -w^{1,O}, -q^1, -w^{I,1})$, daß $N(u, \theta_2^*) = N(u, \theta_1^*)$ für alle $u_u \leq u \leq u_o$.

nem θ.[26] Zu beachten ist, daß (17) ein stochastisches Differenzengleichungssystem ist, während das Differentialgleichungssystem (18) deterministisch ist. Die angesprochene Approximation sowie deren Eigenschaften gelten deshalb nur im wahrscheinlichkeitstheoretischen Sinn. Weil im hier betrachteten Modell nur eine exogene Variable vorliegt, genügt es zu unterstellen, daß u_t für alle t die Realisation einer unabhängig und identisch verteilten Zufallsvariablen ist, um die beschriebene Approximation verwenden zu können.[27]

Für die Approximation des Algorithmus (17) durch das Differentialgleichungssystem (18) gilt folgendes (vgl. Ljung (1977), Kuan und White (1994)):

(a) Asymptotisch stimmen die Zeitpfade für die Parametervektoren gemäß (17) mit den Trajektorien des Differentialgleichungssystems (18) überein.

(b) Sofern θ_t aus (17) für $t \to \infty$ konvergiert, dann gegen einen stationären Punkt von (18).

(c) Es besteht nur dann eine positive Wahrscheinlichkeit dafür, daß θ_t gemäß (17) gegen einen stationären Punkt von (18) konvergiert, wenn dieser stationäre Punkt lokal stabil ist.[28]

Wie (b) verdeutlicht, müssen lediglich die stationären Punkte des Differentialgleichungssystems (18) betrachtet werden, um Aussagen über die mögliche Konvergenz des Lernalgorithmus (17) treffen zu können. Aufgrund von Annahme 1 sind einige solcher stationären Punkte bereits bekannt: Für alle Parametervektoren $\theta \in \Theta^*$, also für alle Parametervektoren, die rationale Erwartungen implizieren, gilt $M(N(u,\theta),u) = N(u,\theta)$ für alle $u_u \leq u \leq u_o$, so daß $\widetilde{L}(\theta) = 0$. Nur diese stationären Punkte des Differentialgleichungssystems (18) sind von Interesse, wenn die Konvergenz gegen rationale Erwartungen untersucht werden soll. Zusammen mit (c) folgt, daß der Parametervektor θ_t gemäß (17) nur dann gegen ein $\theta \in \Theta^*$ konvergieren kann, demnach nur dann asymptotisch rationale Preiserwartungen resultieren können, wenn die stationären Punkte $\theta \in \Theta^*$ des Differentialgleichungssystems (18) lokal stabil sind. Es ergibt sich somit:

[26] Eine Begründung für die Gültigkeit der Approximation ist hier nicht möglich, da diese formal recht aufwendig ist. Ausführliche Beschreibungen und ökonomische Anwendungen des hier verwendeten Ansatzes finden sich z. B. bei Marcet und Sargent (1989) sowie Sargent (1993).

[27] Das Modell ist dann äquivalent zu dem von White (1989) betrachteten Modell. Allerdings liegt hier im Gegensatz zu White (1989) Forecast-Feedback vor. White (1989) zeigt ausführlich, daß der von Ljung (1977) entwickelte Formalismus auf dieses Modell anwendbar ist.

[28] Unter der Stabilität eines stationären Punktes ist hier und im weiteren immer Stabilität im Sinne von Lyapunov zu verstehen.

Satz 2 *Die Wahrscheinlichkeit, daß θ_t gemäß (17) asymptotisch gegen ein $\theta \in \Theta^*$ konvergiert, ist dann positiv, wenn die Matrix der partiellen Ableitungen:*

$$J(\theta) = \frac{\partial \widetilde{L}(\theta)}{\partial \theta'} = \nabla \widetilde{L}(\theta) \tag{19}$$

nur Eigenwerte mit negativen Realteilen besitzt.

Aus (18) folgt:

$$J(\theta) = \mathrm{E}_u \left\{ \nabla^2 N \left[M(N(u,\theta),u) - N(u,\theta) \right] + \left(\nabla N \nabla N' \right) \left[M_p(N(u,\theta),u) - 1 \right] \right\}$$

Hierbei gilt $\nabla N = \nabla N(u,\theta)$ sowie $\nabla^2 N = \nabla^2 N(u,\theta)$. M_p bezeichnet die partielle Ableitung der Funktion $M(p^e,u)$ nach der Preiserwartung p^e. Wegen $M(N(u,\theta),u) = N(u,\theta)$ für $\theta \in \Theta^*$ folgt daraus:

$$J(\theta) = \mathrm{E}_u \left\{ \left(\nabla N \nabla N' \right) \left[M_p(N(u,\theta),u) - 1 \right] \right\} \tag{20}$$

Da $J(\theta)$ eine symmetrische Matrix ist, ist die Bedingung, daß θ ein stabiler stationärer Punkt von (18) ist, gleichbedeutend damit, daß $J(\theta)$ negativ definit ist. Leider läßt sich diese Bedingung nicht einfach überprüfen: Zwar ist $\mathrm{E}_u \{ \nabla N \nabla N' \}$ positiv definit, wenn das neuronale Netz keine redundanten verdeckten Neuronen enthält.[29] Damit kann jedoch nicht notwendigerweise auf die Eigenschaften von $J(\theta)$ geschlossen werden. Es kann aber zumindest eine notwendige Bedingung für die Stabilität von θ abgeleitet werden: $J(\theta)$ ist nur dann negativ definit, wenn alle Hauptdiagonalelemente negativ sind. Nun ist das erste Element des Vektors θ ein absolutes Glied (vgl. Gleichung (10)), so daß eines der Elemente des Vektors ∇N gleich eins ist. Eines der Hauptdiagonalelemente von $J(\theta)$ aus (20) ist daher gegeben durch $\mathrm{E}_u [M_p(N(u,\theta),u) - 1]$.

Somit ist $\mathrm{E}_u [M_p(N(u,\theta),u) - 1] < 0$ eine notwendige Bedingung dafür, daß θ ein stabiler stationärer Punkt des Differentialgleichungssystems (18) ist und es resultiert als notwendige Bedingung für die Konvergenz von θ_t:

Satz 3 *Es sei θ ein Parametervektor, bei dem rationale Preiserwartungen resultieren, so daß $N(u,\theta) = M(N(u,\theta),u)$ für alle $u_u \leq u \leq u_o$. Es besteht nur dann eine positive Wahrscheinlichkeit dafür, daß θ_t gemäß (17) asymptotisch gegen $\theta \in \Theta^*$ konvergiert, wenn $\mathrm{E}_u [M_p(N(u,\theta),u)] < 1$ gilt.*

[29] Redundante verdeckte Neuronen liegen vor, wenn die unbekannte Funktion mit Hilfe von *m* verdeckten Neuronen perfekt approximiert werden kann, tatsächlich jedoch $n > m$ verdeckte Neuronen verwendet werden. In diesem Fall liegt ein Problem vor, das dem der Multikollinearität im klassischen linearen Regressionsmodell nicht unähnlich ist. Liegen redundante verdeckte Neuronen vor, sind einzelne Elemente des Vektors ∇N identisch, so daß $J(\theta)$ Eigenwerte besitzt, deren Realteile gleich null sind (vgl. White (1989, S. 143)).

Satz 3 erfordert, daß der Erwartungswert der Ableitung des Marktpreises nach der Preiserwartung $E_u[M_p(p_t^e, u_t)]$ im rationalen Erwartungsgleichgewicht — mit $p_t^e = p^* = N(u_t, \theta)$ — kleiner als eins ist. Dies ist eine notwendige Bedingung dafür, daß ein $\theta \in \Theta^*$ ein lokal stabiler stationärer Punkt des Differentialgleichungssystems (18) ist, also dafür, daß die Wahrscheinlichkeit, daß θ_t gemäß (17) für $t \to \infty$ gegen θ konvergiert, positiv ist.

Die in Satz 3 genannte Bedingung kann inhaltlich wie folgt interpretiert werden: Der Lernalgorithmus (17) sorgt dafür, daß immer dann, wenn das neuronale Netz bei gegebenem u_t und θ_t eine Preiserwartung $N(u_t, \theta_t)$ generiert, die den tatsächlich resultierenden Preis (unter-) überschätzt, der Parametervektor so modifiziert wird, daß bei diesem u_t eine nunmehr geringere (größere) Preiserwartung resultiert. Bei unverändertem u_t kann eine ständige Wiederholung dieses Vorgangs nur dann die für dieses u_t korrekte Preiserwartung liefern, wenn $M_p(p^*, u_t) < 1$ gilt. Nur dann wird die Differenz zwischen dem erwarteten und dem tatsächlichen Preis im Zeitablauf geringer. Nun ist jedoch u_t nicht konstant, sondern eine Zufallsvariable, so daß es sich auch bei der Ableitung $M_p(p^*, u_t)$ um eine Zufallsvariable handelt. Von daher erfordert die Konvergenz des Lernalgorithmus (17), daß diese Stabilitätsbedingung im Erwartungswert erfüllt ist.

Diese Bedingung kann recht einfach inhaltlich erläutert werden. Schauen wir uns dazu die folgende Abbildung an.

Die Abbildung zeigt den tatsächlichen Preis in Abhängigkeit von der Preiserwartung bei gegebenem Schock $\bar{u}$. Die Funktion M hat im vorliegenden Fall bei der rationalen Preiserwartung eine Steigung, die größer als Eins ist, das heißt, die eben dargestellte Bedingung ist nicht erfüllt.

Wenn nun beispielsweise die Preiserwartung p_1^e vorliegt, so resultiert der tatsächliche Preis p_1, das heißt die Erwartung p_1^e stellt sich als zu hoch heraus. Der hier unterstellte Lernalgorithmus wird in einem solchen Fall die Gewichte so anpassen, daß hinterher bei diesem $\bar{u}$ eine geringere Preiserwartung generiert wird. Die Folge ist jedoch wie man sieht, daß sich die Preiserwartungen von der rationalen Preiserwartung fortbewegen.

Es ist relativ leicht zu zeigen, daß diese notwendige Konvergenzbedingung in unserem vorhin betrachteten Cobweb-Modell erfüllt ist.

Es sollte beachtet werden, daß die in Satz 3 genannte Bedingung unabhängig davon ist, ob ein neuronales Netz oder ein anderes Verfahren zum Lernen der Preiserwartungsbildung eingesetzt wird: Da die Funktion $N(\theta, u_t)$ für alle $\theta \in \Theta^*$ annahmegemäß rationale Preiserwartungsbildung impliziert, bezieht sich diese Bedingung ganz allgemein auf die Eigenschaften der Funktion $M(p_t^e, u_t)$ in der Umgebung des betrachteten rationalen Erwartungsgleichgewichts. Somit ist es letztlich von den Eigenschaften des unterstellten Modells abhängig, ob asymptotisch rationale Erwartungsbildung erlernt werden kann. Beispielsweise ergibt sich für das in Abschnitt 2.1 formulierte Cobweb-Modell, daß:

$$M_p(p^*, u) = -\frac{1}{\sqrt{2 + 6u + 159u^2} - 2\sqrt{1 + 6u + 159u^2}}$$

Für alle $u \in (0,1]$ gilt somit $M_p(p^*,u) < 0$ und die notwendige Bedingung für die Konvergenz des Lernalgorithmus (17) wird durch das betrachtete Modell erfüllt.

Sollte die Bedingung aus Satz 3 nicht erfüllt sein, dann ist es im Rahmen des hier betrachteten Modells nicht möglich, mittels eines neuronalen Netzes asymptotisch rationale Erwartungsbildung zu lernen. Selbst wenn Satz 3 erfüllt ist, folgt daraus allerdings weder die lokale Stabilität von θ, noch, daß θ_t mit Sicherheit gegen θ konvergiert. Letzteres erfordert, daß θ ein lokal stabiler stationärer Punkt des Differentialgleichungssystems (18) ist und zudem, daß sich θ_t mit Wahrscheinlichkeit eins unendlich häufig in einer Umgebung dieses stationären Punktes befindet, aus der alle Trajektorien asymptotisch in den stationären Punkt münden. Nur unter diesen Bedingungen konvergiert θ_t mit Sicherheit gegen θ. Entsprechende hinreichende Bedingungen für die sichere Konvergenz von θ_t gegen ein $\theta \in \Theta^*$ können formuliert werden, wenn der Lernalgorithmus (17) modifiziert, das heißt Ljung (1977) folgend um eine sogenannte Projektionseigenschaft (*projection facility*) erweitert wird.[30] Die entsprechende Projektion stellt sicher, daß der Parametervektor θ_t den Attraktionsbereich eines stationären Punktes θ nicht verläßt, so daß θ_t mit Sicherheit gegen θ konvergiert. Inhaltlich bereitet diese Projektionseigenschaft allerdings Probleme: Wird wie hier davon ausgegangen, daß die Firmen den Lernalgorithmus verwenden, um das Bilden von Preiserwartungen zu lernen, dann sind es die Firmen, die die angesprochene Projektion des Parametervektors vornehmen. Wie jedoch beispielsweise Zenner (1996) bemerkt, dürfte es den Firmen ohne Kenntnis des wahren Modells schwerfallen, diese Aufgabe zu lösen. Es sei denn, sie kennen das wahre Modell — dann existiert allerdings kein Lernproblem mehr.

5. Schlußbemerkungen

Ausgangspunkt des vorliegenden Papiers war die Frage, inwieweit neuronale Netze dazu genutzt werden können, die Erwartungsbildung von Firmen im Rahmen eines einfachen, an den Cobweb-Fall angelehnten Modells zu beschreiben. Zu diesem Zweck wurde eine spezielle Form neuronaler Netze — sog. feedforward Netze mit einer Schicht verdeckter Neuronen — herangezogen. Diese Klasse neuronaler Netze zeichnet sich durch weitreichende Approximationseigenschaften aus, verfügt also über eine große Flexibilität, die es ermöglicht, verschiedenste Zusammenhänge zwischen relevanten Variablen abzubilden.

Es konnte zunächst anhand eines Simulationsbeispiels gezeigt werden, daß die Firmen im betrachteten Modell mittels eines solchen neuronalen Netzes zumindest langfristig Erwartungen bilden, die sich kaum von den entsprechenden rationalen Erwartungen unterscheiden. Letzteres ist insofern bemerkenswert, als das Bilden rationaler Erwartungen in der Regel ein außerordentliches Vorwissen der Wirtschaftssubjekte bezüglich ihrer ökonomisch relevanten Umwelt erfordert, welches im Rahmen des hier betrachteten Modells

[30] Für eine ausführliche Diskussion dieser Projektionseigenschaft siehe Marcet und Sargent (1989).

gerade nicht vorausgesetzt wurde.

Da ein Simulationsbeispiel keine weitreichenden Schlußfolgerungen zuläßt, wurde versucht, allgemeine Aussagen über die asymptotischen Eigenschaften solchen neuronalen Lernens abzuleiten. Eine besondere Rolle spielt hierbei das mit dem betrachteten Lernalgorithmus assozierte Differentialgleichungssystem, das herangezogen werden kann, um die fraglichen asymptotischen Eigenschaften zu beschreiben. Die Analyse dieses Differentialgleichungssystems ermöglicht es, allgemeine Bedingungen zu formulieren, die erfüllt sein müssen, damit die Firmen asymptotisch das Bilden rationaler Preiserwartungen lernen.

Anzumerken ist, daß bei der formalen Analyse neuronaler Erwartungsbildung unterstellt wurde, das neuronale Netz könne die fragliche rationale Erwartungsfunktion perfekt approximieren. Diese Annahme ist gleichbedeutend mit der Annahme einer — im Hinblick auf die Anzahl der verdeckten Neuronen — korrekten Spezifikation des zugrundeliegenden neuronalen Netzes. Sofern diese Annahme nicht erfüllt ist, sind die Firmen auch mittels eines neuronalen Netzes nicht in der Lage, asymptotisch rationale Erwartungen zu bilden. Allerdings können sie wie im betrachteten Simulationsbeispiel neuronale Erwartungen bilden, die rationale Erwartungen hinreichend gut approximieren.

Eine hinreichend gute Approximation rationaler Erwartungen durch neuronale Netze bedeutet aber, daß neuronale Erwartungen zumindest annähernd die gleichen Ergebnisse hervorbringen wie rationale Erwartungen — ohne daß die unrealistische Annahme vollständiger Information erforderlich ist.

Literaturverzeichnis

Arminger, G. (1994), Ökonometrische Schätzmethoden für neuronale Netze, in: G. Bol, G. Nakhaeizadeh und K.-H. Vollmer (Hrsg.), Finanzmarktanwendungen neuronaler Netze und ökonometrischer Verfahren, Heidelberg, 25-40.

Baun, S. (1994), Neuronale Netze in der Aktienkursprognose, in: H. Rehkugler und H.G. Zimmermann (Hrsg.), Neuronale Netze in der Ökonomie, München, 131-208.

Blien, U. / *Lindner,* H.-G. (1993), Neuronale Netze — Werkzeuge für empirische und ökonomische Fragestellungen, Jahrbücher für Nationalökonomie und Statistik 212, 497-521.

Bol, G. / *Nakhaeizadeh,* G. / *Vollmer,* K.-H. (1994), Finanzmarktanwendungen neuronaler Netze und ökonometrischer Verfahren, Heidelberg.

Bray, M. (1983), Convergence to Rational Expectations Equilibrium, in: R. Frydman / E. Phelps (Hrsg.), Individual Forecasting and Aggregate Outcomes, Cambridge/Mass., 123-132.

Hoffmann, N. (1993), Kleines Handbuch neuronaler Netze, Braunschweig/Wiesbaden.

Heinemann, M. / *Lange,* C. (1997) Simulationen zur neuronalen Erwartungsbildung, Internet, URL: http://www.wiwi.uni-hannover.de/gkw/lange/veroeffe/neuro/.

Hornik, K. / *Stinchcombe,* M. / *White,* H. (1989), Multilayer Feedforward Networks are Universal Approximators, Neural Networks 2, 359-366.

Kuan, C.-M. / *White,* H. (1994), Adaptive Learning with Nonlinear Dynamics driven by Dependend Processes, Econometrica 62, 1087-1114.

Ljung, L. (1977), Analysis of Recursive Stochastic Algorithms, IIIE Transactions on Automatic Control AC–22, 551-575.

Marcet, A. / *Sargent,* T. (1989), Convergence of Least Squares Learning Mechanisms in Self Referential Linear Stochastic Models, Journal of Economic Theory 48, 337-368.

Murtagh, F. (1990), Neural Networks for Statistical and Economic Data, Dublin.

Muth, J. (1961), Rational Expectations and the Theory of Price Movements, Econometrica 29, 315-335.

Sargent, T. (1993), Bounded Rationality in Macroeconomics, Oxford.

White, H. (1989), Some Asymptotic Results for Learning in Single Hidden Layer Feedforward Networks, Journal of the American Statistical Association 84, 1008-1013.

Würtz, D. / *Murtagh,* F. (1991), PASE 1991: International Workshop on Parallel Problem Solving from Nature — Applications in Statistics and Economics, Zürich.

Zenner, M. (1996), Learning to Become Rational: The Case of Self-Referential and Non-Stationary Models, Berlin.

Zimmermann, H.G. (1994), Neuronale Netze als Entscheidungskalkül, in: H. Rehkugler und H.G. Zimmermann (Hrsg.), Neuronale Netze in der Ökonomie, München, 1-87.