Rekurrente Netze
-
Das Elman Netz

Bachelorarbeit

eingereicht von

Stefan Vikoler

Betreuerin:

Dr. Claudia-Lidia Badea

Universität Salzburg
Fachbereich Computerwissenschaften
Jakob-Haringer-Strasse 2
A-5020 Salzburg
Austria

Juni 2012

Vorwort

Diese Arbeit ist im Rahmen meines Bachelorabschlusses an der Universität Salzburg, Fachbereich Computerwissenschaften, unter der Aufsicht von Dr. Claudia-Lidia Badea entstanden. Es umfasst die Definition von rekurrenten Netzen und einen Lernalgorithmus, um rekurrente Netze zu trainieren. Hauptaugenmerk wird in dieser Arbeit auf das Elman Netz, die dynamische Backpropagation, um ein Elman Netz zu trainieren, und deren Zusammenhang mit der System Identification gelegt.

Alle Abbildungen dieser Arbeit wurden, sofern nicht anders angegeben, eigenhändig erstellt.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Architekturen von neuronalen Netzen

Der Lernalgorithmus, mit welchem man ein Netz versucht zu trainieren, hängt davon ab, wie die Neuronen eines neuronalen Netzes strukturiert sind. Man kann Netz-Architekturen in drei fundamentale Klassen unterscheiden:

1. Einschichtige Feedforward Netze
2. Mehrschichtige Feedforward Netze
3. Rekurrente Netze

1.1.1 Einschichtige Feedforward Netze

In geschichteten neuronalen Netzen werden die Neuronen in Form von Schichten organisiert. In der einfachsten Form von geschichteten Netzen gibt es eine Input- und eine Outputschicht, wobei die Inputschicht Signale zur Outputschicht, welche die Berechnungen durchführt, sendet und keine umgekehrte Verbindungen existieren. Solche Netze heißen feedforward- oder azyklische Netze. In Abbildung 1.1 sehen wir ein Netz mit drei Knoten in der Input- und drei Knoten in der Outputschicht. Ein solches Netz nennt man einschichtig.

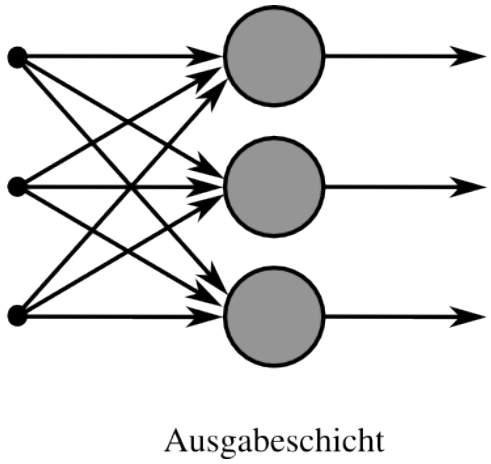

Abbildung 1.1: SingleLayer Network *(de.wikipedia.org)*

1.1.2 Mehrschichtige Feedforward Netze

Die zweite Klasse der feedforward Netze unterscheidet sich durch eine oder mehrere verdeckte Schichten, deren Knoten zur Berechnung zuständig sind und verdeckte Neuronen genannt werden. Ihre Aufgabe ist es, zwischen der Input- und der Outputschicht nützlich zu intervenieren. Abbildung 1.2 zeigt eine Abbildung eines mehrschichtigen feedforward Netzes mit einer einzelnen verdeckten Schicht mit 3 verdeckten Neuronen.

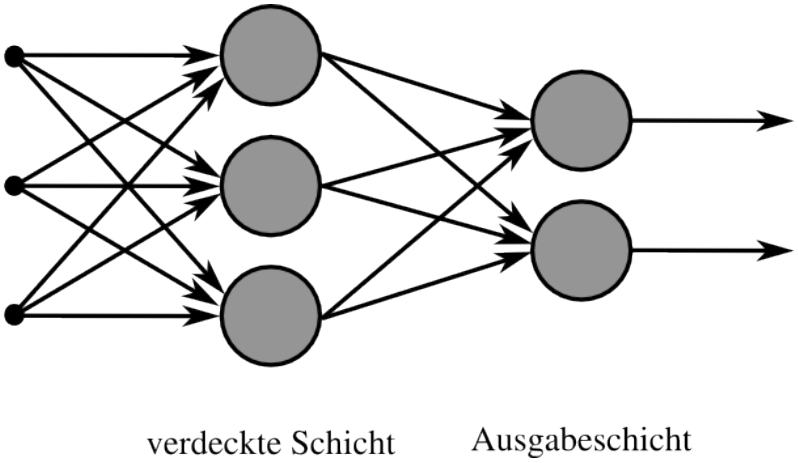

Abbildung 1.2: MultiLayer Network *(de.wikipedia.org)*

1.1.3 Rekurrente Netze

Rekurrente Netze sind Neuronale Netze welche sich von den feedforward Netzen unterscheiden, indem sie mindestens eine umgekehrte Verbindung (Rückkopplung) enthalten, eine sogenannte „feedback loop“. Zum Beispiel kann ein rekurrentes Netz (Abbildung 1.3) aus einer Schicht von Neuronen bestehen, welche ihr Outputsignal wieder als Inputsignal verwenden.

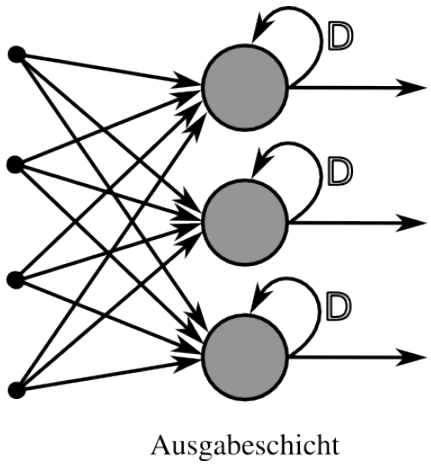

Abbildung 1.3: Recurrent Network 1 *(de.wikipedia.org)*

Ein anderes Beispiel (Abbildung 1.4) zeigt ein rekurrentes Netz, welches das berechnete Signal der Outputschicht über ein zusätzliches Neuron in der Inputschicht wieder in das Netz einspeist.

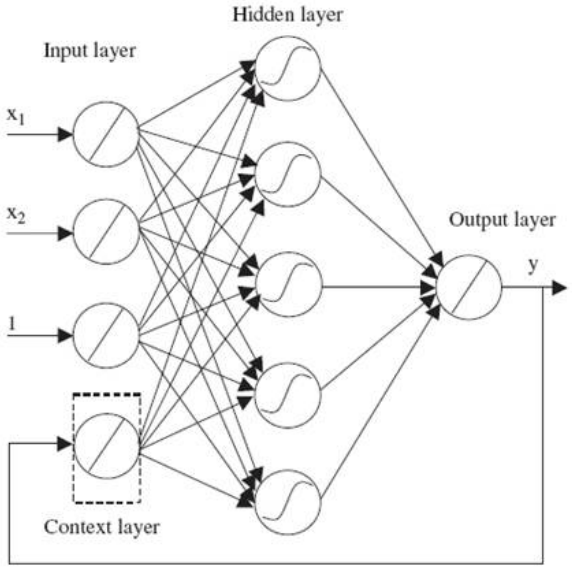

Abbildung 1.4: Recurrent Network 2 *(web.mst.edu)*

Die Verwendung von „feedback loops“ hat eine nachhaltige Auswirkung auf die Lernkapazität und die Performanz eines Netzes. Meist werden solche umgekehrten Verbindungen zeitlich verzögert, sodass ein Gedächtnis und ein dynamisches Verhalten für das Netz entsteht.

Kapitel 2

Rekurrente Netze

2.1 Modelle von Rekurrenten Netzen

Wie in der Einleitung bereits erwähnt, können die Architekturen eines rekurrenten Netzes viele verschiedene Formen annehmen. In diesem Kapitel werden zwei Modelle beschrieben, das Input-Output Model und das State-Space Model.[5]

2.1.1 Input-Output Model

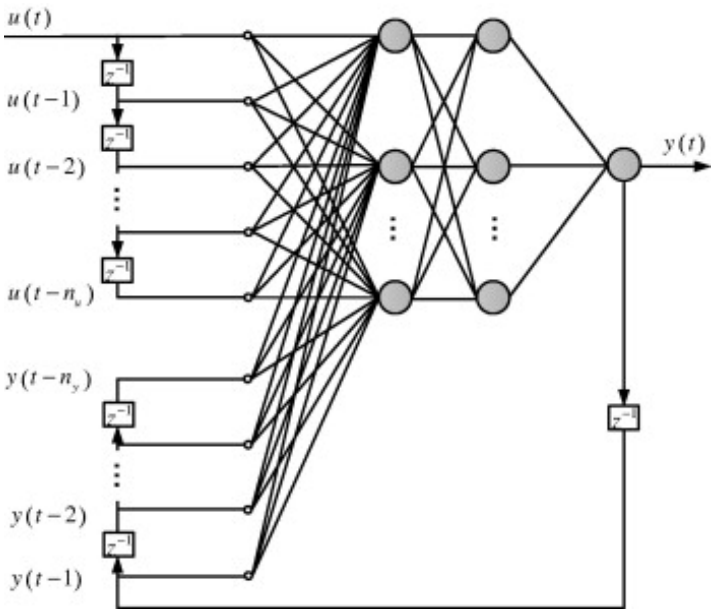

Abbildung 2.1: Input-Output Model *(sciencedirect.com)*

Abbildung 2.1 zeigt eine Architektur eines rekurrenten Netzes mit einem einzelnen Inputsignal $u(t)$ mit einer Zeitverschiebung von n_u Einheiten. Außerdem hat es ein Outputsignal $y(t)$, welches über n_y Zeitverschiebungen wieder als Input verwendet wird. Ein solches Modell nennt man Input-Output Model.
Dadurch ergeben sich folgende Inputdaten für das Netz:

- Die akutellen und die vergangenen Inputwerte $u(t), u(t-1), \ldots, u(t-n_u)$
- und die zeitverzögerten Werte des Outputneurons $y(t-1), \ldots, y(t-n_y)$.

Ein solches Modell besitzt ein dynamsiches Verhalten welches durch

$$y(t) = F\left(y(t-1), \ldots, y(t-n_y), u(t), \ldots, u(t-n_u)\right)$$

beschrieben wird, wobei F eine lineare Funktion ist.

2.1.2 State-Space Model

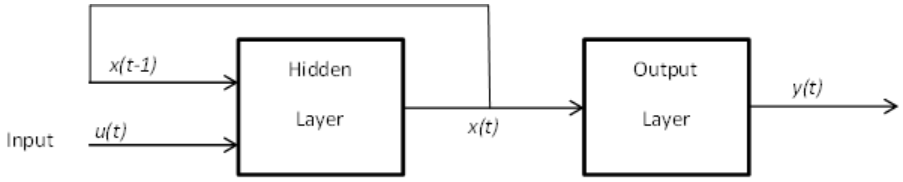

Abbildung 2.2: State-Space Model

Abbildung 2.2 zeigt eine weitere Architektur eines rekurrenten Netzes, welches State-Space Model genannt wird. Dabei wird der Output der verdeckten Schicht $x(t)$ zeitversetzt wieder zum Input dazugenommen. Der Input besteht also aus einer Verkettung der Inputwerte $u(t)$ und den zeitversetzten Outputwerten der verdeckten Schicht. Auch hier können mehrere zeitversetzte Werte der Input- beziehungsweise der verdeckten Schicht verwendet werden. Somit erhalten wir ein dynamisches Verhalten des State-Space Model mit der Gleichung

$$\begin{aligned} x(t) &= f\left(x(t-1), u(t)\right) \text{ und} \\ y(t) &= Cx(t), \end{aligned}$$

wobei f eine lineare Funktion und C die Gewichte zur Outputschicht darstellen.

Beide Modelle lassen sich durch ein Elman Netz darstellen.

Kapitel 3

Das Elman Netz

Elman Netze[3] besitzen die Eigenschaft zeitveränderliche Muster zu erkennen und zu klassifizieren, da bei einem Elman Netz der Output der verdeckten Schicht gespeichert und zeitversetzt als Input verwendet wird, indem man in der Inputschicht eine Kontextschicht anlegt.

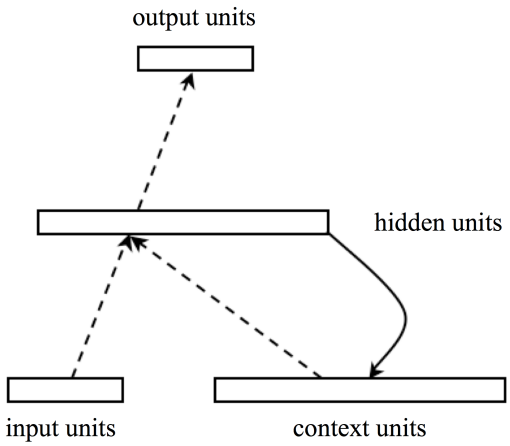

Abbildung 3.1: Elman Network*(plato.stanford.edu)*

Die Architektur (Abbildung 3.1) eines Elman Netzes besteht also aus einer Inputschicht, einer verdeckten Schicht, einer Outputschicht und der Kontextschicht, in der die Werte der verdeckten Schicht zeitversetzt gespeichert werden. Somit wird der innere Zustand des Netzes in der Kontextschicht gespeichert.

Kapitel 4

Der Algorithmus

Es existieren verschiedenste Lernalgorithmen um Elman Netze und im Allgemeinen rekurrente Netze zu trainieren. Vieles hängt von der Architektur und der Aufgabenstellung des Netzes ab. Beispiele sind etwa

- der dynamische Backpropagation Algorithmus,
- der Real-Time Recurrent Learning Algorithmus
- oder die Kalman Filter.

In dieser Arbeit wird speziell auf den dynamischen Backpropagation Algorithmus[6] eingegangen.

4.1 Dynamische Backpropagation

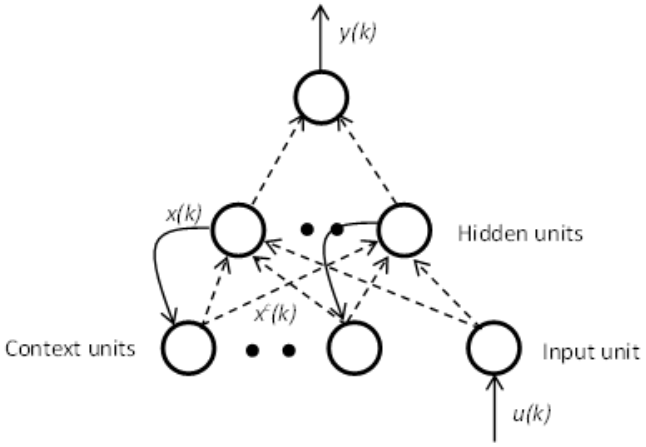

Abbildung 4.1: Basic Elman Network

Abbildung 4.1 zeigt die Architektur eines klassischen Elman Netzes nach Elman 1990[3]. Die feedforward-Verbindungen (gestrichelte Linien) sind veränderbar und die rekurrenten Verbindungen (geschlossene Linien) sind fix.

4.1.1 Berechnung der Neuronen

Sei der Input des Netzes $u(k)$, der Output $y(k)$, der gesamte Input zu den einem i-ten verdeckten Neuron $v_i(k)$, der Output eines i-ten verdeckten Neuron $x_i(k)$ und der Output eines j-ten Kontextneurons $x_j^c(k)$, dann gilt:

$$v_i(k) = \sum_{j=1}^{n} w_{i,j}^x(k-1)x_j^c(k) + w_i^u(k-1)u(k), \tag{1a}$$

$$x_i(k) = f(v_i), \tag{1b}$$

$$x_j^c(k) = x_j(k-1), \tag{1c}$$

$$y(k) = \sum_{i=1}^{n} w_i^y(k-1)x_i(k), \tag{1d}$$

wobei $w_i^u, w_{i,j}^x$ und w_i^y, $i, j = 1, \ldots, n$, die Gewichte der Verbindungen zwischen Input- und verdeckter Schicht, Kontext- und verdeckter Schicht und zwischen verdeckter und Outputschicht sind und f eine sigmoidale Aktivierungsfunktion ist.
Wird nun der Input $u(k)$ um einen Zeitschritt verzögert, bevor er gesendet wird, wird $x_j^c(k)$ mit $x_j(k-1)$ ersetzt und angenommen die verdeckten Neuronen sind linear, dann gelten folgende Gleichungen:

$$v_i(k) = \sum_{j=1}^{n} w_{i,j}^x(k-1)x_j(k-1) + w_i^u(k-1)u(k-1), \tag{2a}$$

$$x_i(k) = v_i(k), \tag{2b}$$

$$y(k) = \sum_{i=1}^{n} w_i^y(k-1)x_i(k). \tag{2c}$$

Diese Gleichungen beschreiben ein State-Space Model n-ter Ordnung eines linearen dynamischen Systems und können als folgende Gleichung dargestellt werden:

$$\begin{aligned} y(k) = {} & A_1 y(k-1) + A_2 y(k-2) + \cdots + A_n y(k-n) \\ & + B_1 u(k-1) + B_2 u(k-2) + \cdots + B_n u(k-n). \end{aligned} \tag{3}$$

4.1.2 Training durch dynamische Backpropagation

Seien die Trainingswerte das Paar $(u(k), y_d(k))$, $k = 1, 2, \ldots, N$, mit $u(k)$ als Input und $y_d(k)$ als gewünschter Output.
Die Fehlerfunktion des Netzes zum Zeitpunkt k ist

$$E_k = \frac{1}{2}\left(y_d(k) - y(k)\right)^2. \tag{4}$$

Die Gewichte des Netzes werden zu jedem Zeitpunkt k modifiziert. Der Fehlergradient für $w_i^y(k-1)$ ist

$$\begin{aligned}\frac{\partial E_k}{\partial w_i^y(k-1)} &= -(y_d(k) - y(k))\frac{\partial y(k)}{\partial w_i^y(k-1)} \\ &= -(y_d(k) - y(k))x_i(k).\end{aligned} \tag{5}$$

Die Fehlergradienten für $w_i^u(k-1)$ und $w_{i,j}^x(k-1)$ lauten

$$\begin{aligned}\frac{\partial E_k}{\partial w_i^u(k-1)} &= -\frac{\partial E_k}{\partial y(k)}\frac{\partial y(k)}{\partial x_i(k)}\frac{\partial x_i(k)}{\partial v_i(k)}\frac{\partial v_i(k)}{\partial w_i^u(k-1)} \\ &= -(y_d(k) - y(k))w_i^y(k-1)f'(u(k))\end{aligned} \tag{6}$$

und

$$\begin{aligned}\frac{\partial E_k}{\partial w_{i,j}^x(k-1)} &= -\frac{\partial E_k}{\partial y(k)}\frac{\partial y(k)}{\partial x_i(k)}\frac{\partial x_i(k)}{\partial w_{i,j}^x(k-1)} \\ &= -(y_d(k) - y(k))w_i^y(k-1)\frac{\partial x_i(k)}{\partial w_{i,j}^x(k-1)},\end{aligned} \tag{7}$$

wobei

$$\begin{aligned}\frac{\partial x_i(k)}{\partial w_{i,j}^x(k-1)} &= \frac{\partial x_i(k)}{\partial v_i(k)}\frac{\partial v_i(k)}{\partial w_{i,j}^x(k-1)} \\ &= f'\left(\frac{\partial v_i(k)}{\partial w_{i,j}^x(k-1)}\right) \\ &= f'\left(x_j(k-1) + \sum_{l=1}^{n} w_{i,l}^x(k-1)\frac{\partial x_l(k-1)}{\partial w_{i,j}^x(k-2)}\right).\end{aligned} \tag{8}$$

4.1.3 Zusammenfassung

Der dynamische Backpropagation-Algorithmus kann nun wie folgt zusammengefasst werden.

Berechnung der Neuronen:

$$v_i(k) = \sum_{j=1}^{n} w_{i,j}^x(k-1)x_j(k-1) + w_i^u(k-1)u(k), \tag{9a}$$

$$x_i(k) = f(v_i), \tag{9b}$$

$$y(k) = \sum_{i=1}^{n} w_i^y(k-1)x_i(k). \tag{9c}$$

Update der Gewichte:

$$w_i^y(k) = w_i^y(k-1) - \eta(y_d(k) - y(k))x_i(k), \tag{10a}$$

$$w_i^u(k) = w_i^u(k-1) - \eta(y_d(k) - y(k))w_i^y(k-1)f'(u(k)), \tag{10b}$$

$$w_{i,j}^x(k) = w_{i,j}^x(k-1) - \eta(y_d(k) - y(k))w_i^y(k-1)\frac{\partial x_i(k)}{\partial w_{i,j}^x(k-1)}, \tag{10c}$$

$$\frac{\partial x_i(k)}{\partial w_{i,j}^x(k-1)} = f'\left(x_j(k-1) + \sum_{l=1}^{n} w_{i,l}^x(k-1)\frac{\partial x_l(k-1)}{\partial w_{i,j}^x(k-2)}\right). \tag{10d}$$

Kapitel 5

Anwendungen

In dieser Arbeit werden zwei Anwendungen eines Elman Netzes vorgestellt. Zum Einen eine klassische Aufgabenstellung für solche Netze, die System Identification eines Input-Output Model und zum Anderen eine ungewöhnliche Problematik für Elman Netze, die Approximation von Funktionen.

5.1 System Identification

System Identification[5] ist die experimentelle Annäherung, um einen Prozess oder eine Anlage mit unbekannten Parametern zu modellieren. Dazu werden folgende Schritte benötigt:

- Experimentelles Planen
- Auswahl einer Modellstruktur
- Schätzung der Parameter
- Modellauswertung

Der Prozess der System Identification ist iterativ, in dem man Schritte zurück und wieder vorwärts gehen muss, bis ein zufriedenstellendes Modell erzeugt wurde. Bei der System Identification haben wir die Wahl zwischen einem State-Space Model oder einem Input-Output Model.

5.1.1 System Identification eines Input-Output Model

Angenommen der unbekannte Prozess ist nur über seinen Output erreichbar. Seien $u(t)$ der externe Input und $y(t)$ der Netzoutput zum Zeitschritt t und das Input-Output Model wie folgt:

$$y(t) = a_1 y(t-2) + a_2 y(t-2) + a_3 y(t-3) + b_1 u(t-1),$$

wobei $a_1 = 2.627771, a_2 = -2.333261, a_3 = 0.697676, b_1 = 0.017203$.
Danach wähle ich meine Architektur wie folgt:

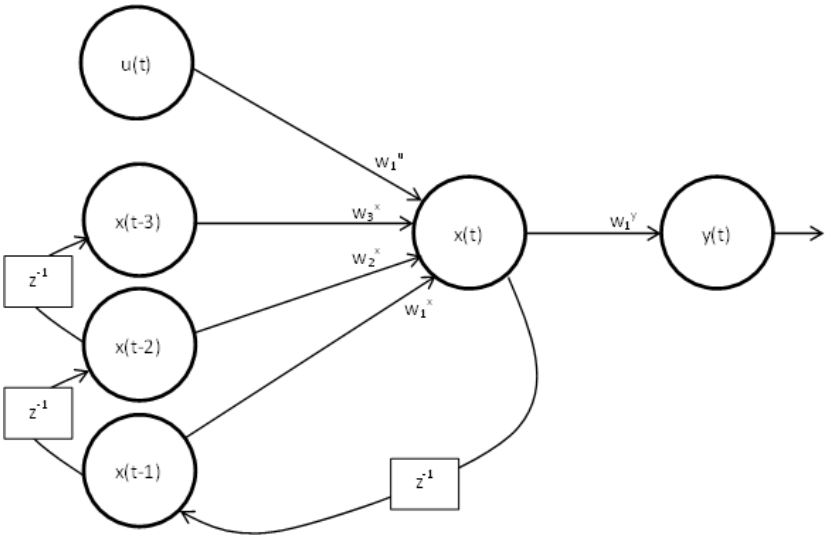

Abbildung 5.1: Architektur

Abbildung 5.1 zeigt eine Architektur mit

- einem Inputneuron $u(t)$,
- einem verdeckten Neuron $x(t)$,
- einem Outputneuron $y(t)$
- und drei Kontextneuronen $x(t-1), x(t-2), x(t-3)$

mit den dazugehörigen Gewichten. Wobei w_1^u das Gewicht der Verbindung vom Inputneuron zum verdeckten Neuron, w_i^x, $1 \leq i \leq 3$, die Gewichte von der Kontextschicht zum verdeckten Neuron und w_1^y das Gewicht vom verdeckten Neuron zum Outputneuron darstellen. z^{-1} stellt den Shiftoperator dar, welcher den Wert der Verbindung um einen Zeitschitt verzögert. Weiters wird als Lernparameter $\eta = 0,7$ und als Aktivierungsfunktion die Sigmoide (Abbildung 5.2) verwendet:

$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

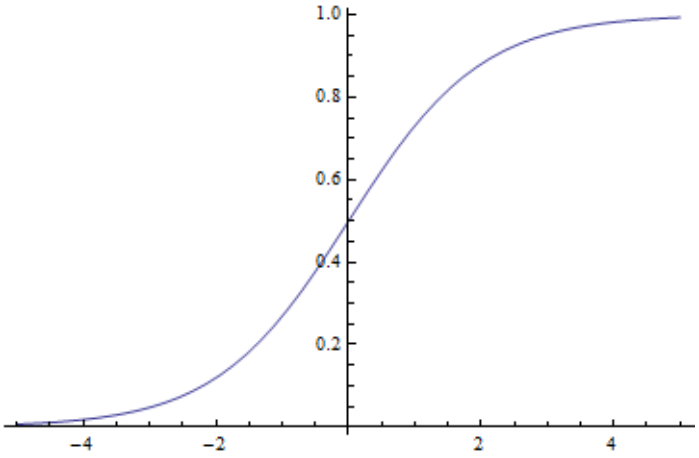

Abbildung 5.2: Sigmoide

Dieses Netz kann man nun mit dem dynamischen Backpropagation Algorithmus sehr erfolgreich trainieren. Dazu werden 100 Input- und Outputwerte verwendet und es sind folgende Berechnungen pro Zeitschitt nötig:

$$
\begin{aligned}
v(t) &= w_1^x x(t-1) + w_2^x x(t-2) + w_3^x x(t-3) + w_1^u u(t) \\
x(t) &= \varphi(v) \\
y(t) &= w_1^y x(t)
\end{aligned}
$$

$$
\begin{aligned}
w_1^u(t) &= w_1^u(t-1) - \eta(y_d(t) - y(t)) w_1^y(t-1) \varphi'(u(t)) \\
w_1^x(t) &= w_1^x(t-1) - \eta(y_d(t) - y(t)) w_1^y(t-1) \frac{\partial x(t)}{\partial w_1^x(t-1)} \\
w_2^x(t) &= w_2^x(t-1) - \eta(y_d(t) - y(t)) w_1^y(t-1) \frac{\partial x(t)}{\partial w_2^x(t-1)} \\
w_3^x(t) &= w_3^x(t-1) - \eta(y_d(t) - y(t)) w_1^y(t-1) \frac{\partial x(t)}{\partial w_3^x(t-1)} \\
w_1^y(t) &= w_1^y(t-1) - \eta(y_d(t) - y(t)) x(t)
\end{aligned}
$$

Ein so trainiertes Elman Netz liefert zum obigen Input-Output Model und nach 100 Epochen folgendes Ergebnis (Abbildung 5.3), wobei die blaue Linie das anzunähernde Input-Output Model und die violetten Punkte den Netzoutput darstellen.

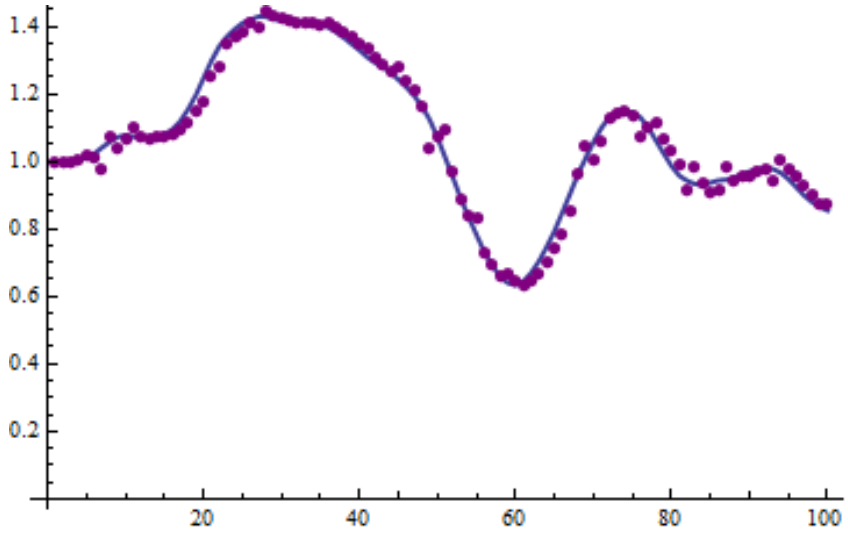

Abbildung 5.3: System Identification

Zu beachten ist, dass durch die Wahl einer anderen Architektur oder bei einem zu niedrig gewählten Lernparameter die Ergebnisse trotz dynamischer Backpropagation abweichen können. Dies zeigt Abbildung 5.4.

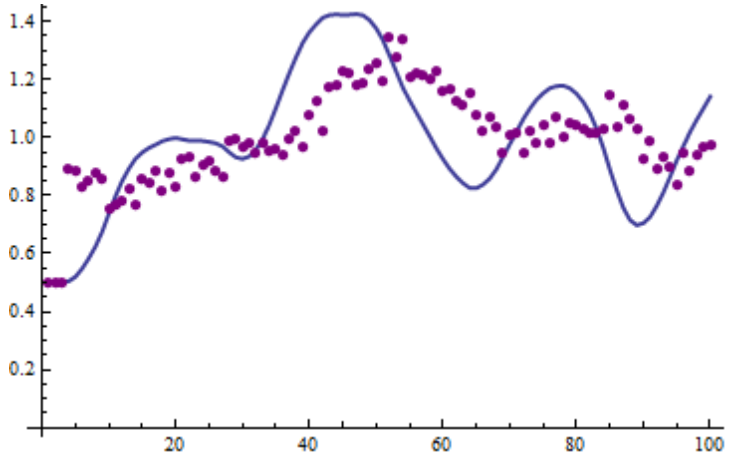

Abbildung 5.4: Fehlerdarstellung

5.2 Approximation von Funktionen

Da die Approximation von Funktionen nicht zu den gewöhnlichen Aufgaben eines Elman Netzes gehört, habe ich versucht eine Funktion mit dem selben Netz, welches bei der System Identification verwendet wurde, zu approximieren. Das heißt, es wurde die selbe Architektur (Abbildung 5.1), der selbe Lernparameter $\eta = 0,7$, die selbe Aktivierungsfunktion $\varphi(x) = \frac{1}{1+e^{-x}}$ und die gleich Menge an 100 Input- und Outputwerten verwendet. Als die zu approximierende Funktion wählte ich

$$e^{-\frac{x}{4}} \cos x.$$

Diese Funktion wird in Abbildung 5.5 dargestellt.

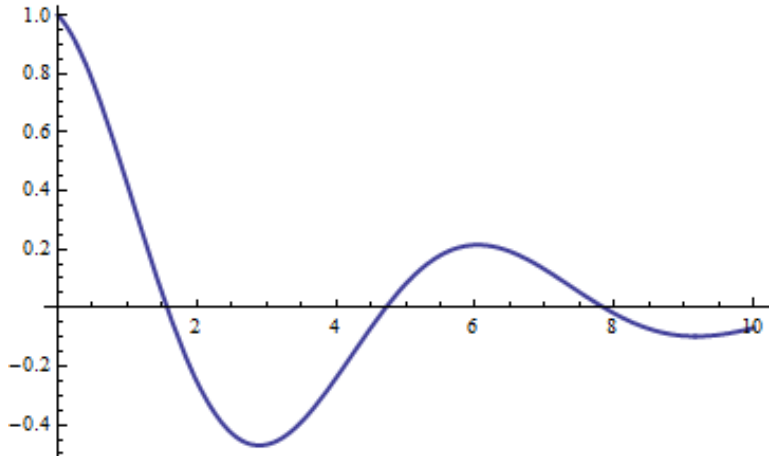

Abbildung 5.5: Funktion

Nach 100 Epochen meines Netzes erhalte ich folgende Gegenüberstellung (Abbildung 5.6), wobei die blaue Linie die zu approximierende Funktion und die violetten Punkte den Netzoutput darstellen.

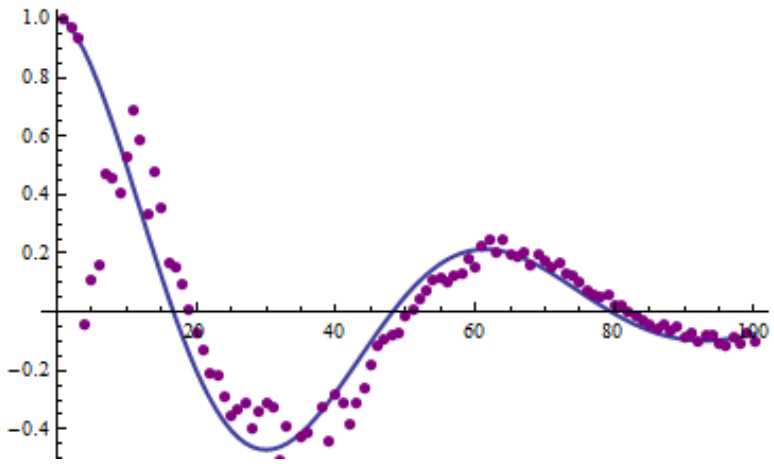

Abbildung 5.6: Approximation

5.3 Fazit

Wie die Gegenüberstellung zeigt, ist die Approximation noch nicht ausgereift und man kann mit einer anderen Architektur eventuell bessere Werte erzielen. Außerdem ist diese Gegenüberstellung ein Ausnahmebeispiel meines Netzes und nicht alle 100 Epochen wurden so gute Ergebnisse getroffen. Der Gesamtfehler liegt hier bei $E = 1,3207026786535492$, wobei der Gesamtfehler der System Identification unter $0,2$ liegt. Abschließend kann man sagen, dass das Problem der System Identification mittels der durch die dynamische Backpropagation trainierten Elman Netze gute Ergebnisse erzielt und für die Funktionenapproximation größere Fehler aufweist.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Claudia-Lidia Badea. *Grundlagen Künstliche Neuronale Netze.* Vorlesungsskript an der Universität Salzburg, Fachbereich Computerwissenschaften, 2012.

[2] Holk Cruse. *Neural Networks as Cybernetic Systems.* Department of Biological Cybernetics and Theoretical Biology, Bielefeld University, second edition, 2006.

[3] JEFFREY L. ELMAN. *Finding Structure in Time.* COGNITIVE SCIENCE, University of California, San Diego, 1990.

[4] Torsten Felzer. *Analyse menschlicher Bewegungssequenzen im Hinblick auf eine Übertragbarkeit auf KÜNSTLICHE LERNENDE SYSTEME.* Technische Hochschule Darmstadt, Darmstadt, 2002.

[5] Simon Hayking. *Neuronal Networks: a conprehensive foundation.* Prentice-Hall, Inc., New Jersey, second edition, 1999.

[6] D. T. Pham and X. Liu. *Training of Elman networks and dynamic system modelling.* volume 27. International Journal of Systems Science, 1996.