GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Fakultät für
Physik

Bericht

Spezialisierungspraktikum

angefertigt von

Roland Simon Zimmermann

aus Recklinghausen

am Max-Planck-Institut für Dynamik und Selbstorganisation

Bearbeitungszeit: 1. April 2009 bis 15. Juli 2009

Erstgutachter/in: Prof. Dr. Ulrich Parlitz

Zusammenfassung

Hier werden auf einer halben Seite die Kernaussagen der Arbeit zusammengefasst.

Stichwörter: Physik, Bachelorarbeit

Abstract

Here the key results of the thesis can be presented in about half a page.

Keywords: Physics, Bachelor thesis

Inhaltsverzeichnis

Nomenklatur

Lateinische Buchstaben

Variable	Bedeutung	Einheit
A	Querschnittsfläche	m^2
c	Geschwindigkeit	m/s

Griechische Buchstaben

Variable	Bedeutung	Einheit
α	Winkel	°; –
ϱ	Dichte	kg/m^3

Indizes

Index	Bedeutung
m	Meridian
r	Radial

Abkürzungen

Abkürzung	Bedeutung
2D	zweidimensional
3D	dreidimensional
max	maximal

1 Einführung

Dieser Bericht gibt eine Übersicht über die im Spezialisierungspraktikum gewonnen Erkenntnisse. In diesem wurde aus dem Bereich des *Machine Learnings* die Technik des *Reservoir Computings* an Hand der Echo State Networks untersucht. Hierbei wurden zuerst die theoretischen Grundlagen betrachtet und die gewonnenen Informationen anschließend auf zwei Anwendungsbeispiele bezogen. Die Wahl der Echo State Networks wurde dadurch bedingt, dass diese eine Vielseite Umsetzung rekurrenter Neuronaler Netzwerke darstellen, wie im Folgenden dargestellt wird.

2 Überblick über Recurrent Neural Networks

In den letzten Jahren hat die Technik der Neuronalen Netzwerke erneut stark an Popularität gewonnen. Dies liegt zum einen an der gestiegenen verfügbaren Rechenleistung und zum anderen an der Entwicklung hierfür notwendiger Algorithmen.
Allgemein lassen sich diese Netze in zwei große Gruppen aufteilen: die der Feed Forward Neural Networks und die der Recurrent Neural Networks, welche im Folgenden als FFNN respektive RNN bezeichnet werden. Ein FFNN besteht aus mehreren Ebenen, welche jeweils aus verschiedenen nicht-linearen Einheiten zusammengesetzt sind. Hierbei wird die erste Ebene für die Eingabe eines Signals und die letzte Ebene als Ausgabe genutzt. Die Einheiten zweier benachbarter Ebenen sind mit individuellen Gewichten vollständig in Richtung der Ausgabe verbunden. Dies bedeutet, dass jedes Einheit x_i^n sein Signal an alle Einheiten der folgenden Ebene x_j^{n+1} mit einem individuellen Gewicht $w_{i\to j}^n$ weitergibt. Zwischen den Einheiten innerhalb einer Ebene bestehen keinerlei Verbindungen. Damit ein solches Netzwerk Vorhersagen treffen kann, muss es trainiert werden. Dies wurde durch die Entwicklung des Backpropagation-Algorithmus stark vereinfacht.

RNN haben einen ähnlichen Aufbau, doch sind bei Ihnen nicht nur Informationsweitergaben in Richtung der Ausgabeebene möglich, sondern auch in die entgegengesetzte Richtung - es können somit alle Einheiten an alle anderen Einheiten Signale weitergeben und von diesen erhalten. Dies kann die Vorhersage in bestimmten Anwendungsbeispielen wie der Text und Sprachanalyse verbessern. Mit diesem Vorteil kommt allerdings auch der Nachteil, dass zum Trainieren nicht mehr der einfachere Backpropagation-Algorithmus Algorithmus genutzt werden kann, sondern eine für RNNs abgewandelte Form genutzt werden muss. Hierfür werden die verschiedenen Zustände die das RNN im Laufe der Signal-Propagation annimmt nacheinander betrachtet und auf diese zeitliche Entwicklung anschließend der Backpropagati-

on-Algorithmus angewendet. Diese Methode ist unter dem Namen Backpropagation through Time (BTT) bekannt. Sie ist rechenaufwendiger und dadurch, dass das Verschwindenden des Gradienten wahrscheinlicher ist, instabiler.

Um dieses Problem zu lösen wurden unter anderem die Echo State Networks (ESN) entwickelt.

3 Echo State Networks

3.1 Überblick

TODO: Etwas über die Entwicklung und das Entstehen der ESNs schreiben.

3.1.1 Aufbau

Ein ESN ist eine besondere Spezialform eines RNNs. Hierbei wird allerdings eine auf dem ersten Blick eigenartige Entscheidung getroffen: Während des gesamten Trainingsvorganges werden die Verbindungen der einzelnen Einheiten größtenteils nicht verändert. Diese Entscheidung beruht auf der Beobachtung, dass sich während des Trainings durch die bekannten Algorithmen die meisten Gewichte nicht verändern.

Im Folgenden wird der Aufbau und anschließend die Funktionsweise eines solchen Netzwerkes beschrieben. Allgemein bildet das Netzwerk S ein zeitliches Signal $\vec{u}(n) \in \mathbb{R}^{N_u}$ auf eine zeitlich variable Ausgabe $\vec{y}(n) \in \mathbb{R}^{N_y}$ für die Zeiten $n = 1, ..., T$ ab. Zudem besitzt das System ein sogenanntes RESERVOIR aus N nicht-linearen Einheiten. Der innere Zustand des Netzwerkes wird durch diese Einheiten beschrieben und als $s(n) \in \mathbb{R}^N$ bezeichnet.
Die Verbindungen der inneren Einheiten untereinander wird durch die Gewichtsmatrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ beschrieben. Das Eingangssignal wird zusammen mit einem *Bias* $b_{in} \in \mathbb{R}$ durch die Matrix $\mathbf{W}_{in} \in \mathbb{R}^{N \times (N_u+1)}$ auf die inneren Einheiten weitergeleitet. Die zeitliche Entwicklung der inneren Zustände berechnet sich nach der Vorschrift

$$\vec{s}(n) = (1 - \alpha) \cdot \vec{x}(n-1)\alpha \cdot f_{in}\left(\mathbf{W}_{in}[b_{in}; \vec{u}(n)] + \mathbf{W}\vec{x}(n-1)\right), \tag{3.1}$$

wobei f_{in} eine beliebige (meistens *sigmoid*-förmige) Transferfunktion ist, und $[\cdot\,;\,\cdot]$ das vertikale Aneinanderfügen von Vektoren beziehungsweise Matrizen bezeichnet. Für diese Zustandsgleichung wurde das Modell eines *Leaky Integrator Neurons* ge-

nutzt, wobei $\alpha \in (0, 1]$ die Verlustrate beschreibt. Für $\alpha = 1$ ergibt als Spezialfall ein gewöhnliches Neuron.

Da für manche Anwendungsfälle auch eine direkte Rückkopplung wünschenswert sein kann, kann man das System noch um eine Ausgabe-Rückkopplung erweitert werden. Diese verbindet die Ausgabe erneut mit den inneren Einheiten durch die Matrix $\mathbf{W}_{fb} \in \mathbb{R}^{N \times N_y}$. Somit ergibt sich

$$\vec{s}(n) = (1 - \alpha) \cdot \vec{x}(n-1)\alpha \cdot f_{in}\left(\mathbf{W}_{in}[b_{in}; \vec{u}(n)] + \mathbf{W}\vec{x}(n-1) + \mathbf{W}_{fb}\vec{y}(n)\right) \tag{3.2}$$

als Zustandsgleichung.

An Hand der inneren Zustände lassen sich nun noch die sogenannten erweiterten inneren Zustände $x(n) = [b_{out}; \vec{s}(n); \vec{u}(n)] \in \mathbb{R}^{1+N_u+N}$ definieren, wobei b_{out} ein Bias für die Ausgabe darstellt.

Aus diesen erweiterten inneren Zuständen kann nun die Ausgabe $\vec{y}(n)$ konstruiert werden. Dies kann entweder im Sinne einer Linearkombination durch die Ausgangsmatrix $\mathbf{W}_{out} \in \mathbb{R}^{(1+N_u+N)\times N_y}$ oder durch andere nicht lineare Klassifizierer wie beispielsweise einer Support Vector Machine (SVM) durchgeführt werden. Hier wird nur der Fall einer Linearkombination betrachtet, da sich für die anderen Methoden ein analoges Verfahren ergibt. Hierdurch erhält man die Ausgabe

$$\tag{3.3}$$

$$vecy(n) = f_{out}\left(\mathbf{W}_{out}\vec{x}(n) = \mathbf{W}_{out}[b_{out}; \vec{s}(n); \vec{u}(n)]\right), \tag{3.4}$$

wobei f_{out} nun die Transferfunktion der Ausgabe steht.

Während die Matrix $\mathbf{W}_{out}$ durch den Trainingsvorgang bestimmt wird, werden die Matrizen $\mathbf{W}_{in}$ und $\mathbf{W}$ a priori generiert und festgelegt. Hierbei hat sich für das Generieren der Eingangsmatrix eine Zufallswahl zwischen -1 und 1 als geschickt herausgestellt. Falls ein Feedback gewünscht ist, also Gleichung ??? genutzt wird, wird $\mathbf{W}_{fb}$ gleichartig konstruiert. Auf das Generieren der inneren Matrix $\mathbf{W}$ wird später genauer eingegangen.

3.1.2 Training

Nachdem der Aufbau des Netzwerkes definiert ist, ergibt sich nun die Frage, wie man den Trainingsvorgang durchführt.
Hierfür wird für die Zeiten $n = 0, ..., T$ das ESN mit dem Signal $\vec{u}(n)$ betrieben und die erweiterten Zustände $\vec{x}(n)$ als Spalten in der *Zustandsmatrix* $\mathbf{X} \in \mathbb{R}^{(1+N_u+N)\times T}$ gesammelt. Analog dazu werden die gewünschten Ausgaben $\vec{y}(n)$ nach dem Anwenden der Inversen f_{out}^{-1} der Ausgabe-Transferfunktion f_{out} auch als Spalten in der *Ausgabematrix* $Y \in \mathbb{R}^{N_y \times T}$ gesammelt.
Nun wird eine Lösung der Gleichung

$$\mathbf{Y} = \mathbf{W}_{out}\mathbf{X} \tag{3.5}$$

für $\mathbf{W}_{out}$ gesucht. Hierfür stehen mehrere Verfahren zur Verfügung, von denen zwei prominente erwähnt sein sollen. Zum einen kann die Lösung durch eine *Tikhonov Regularisierung* mit der Regularisierung $\beta \cdot ||W_i||^2$ mit der Konstante β erhalten werden. Das Verfahren

$$\mathbf{W}_{out} = \mathbf{Y}\mathbf{X}^T \left(\mathbf{X}\mathbf{X}T + \beta I\right)^{-1} \tag{3.6}$$

ist sehr leistungsstark, aber auch teilweise numerisch instabil. Das Verfahren kann allerdings bei geeigneter Wahl von β die besten Ergebnisse erzielen.

Zum anderen kann zur Lösung die *Moore-Penrose-Pseudoinverse* $\mathbf{X}'$ genutzt werden, sodass

$$\mathbf{W}_{out} = \mathbf{Y}\mathbf{X}' \tag{3.7}$$

folgt. Dieses Verfahren ist zwar sehr rechenaufwendig aber dafür numerisch stabil. Nichts desto trotz, kann allerdings auf Grund des Fehlens einer Regularisierung leicht der Effekt des Overfittings auftreten. Dieser kann umgangen werden, wenn in der Zustandsgleichung ??? eine leichte normalverteilte Störung $\vec{\nu}(n)$ der Größenordnung 0.01 bis 0.0001 addiert wird. Dieser Ansatz beruht auf Empirie, da eine mathematische Begründung hierfür noch nicht vollständig gelungen ist.

Nachdem das Training beschrieben wurde, ergibt sich somit der Folgende Funktionsablauf für das Betreiben eines ESN:

1. Zufälliges Generieren der Matrizen $\mathbf{W}_{in}, \mathbf{W}_{fb}$ und Konstruktion der Matrix $\mathbf{W}$

2. Einspeisen des Signals $u(n)$ und Konstruktion der Zustandsmatrix $\mathbf{X}$ und der Ausgabematrix $\mathbf{Y}$

3. Berechnung der Ausgabematrix $\mathbf{W}_{out}$

4. Einspeisen des Signals $u(n)$ für Vorhersagen des Signales $y(n)$ für $n > T$

3.2 Theorie

3.3 Anwendungen

3.3.1 Vorhersage eines Rössler-Systems

3.3.2 Vorhersage eines Mackey-Glass-Systems

4 Fazit

5 Ausblick

Für das Schreiben des Berichtes für dieses Spezialisierungspraktikums hat sich die Wahl der LaTeX-Umgebung als gut geeignet herausgestellt. Die optisch Darstellung von mathematischen Ausdrücken sowie des erklärenden Textes ist hierbei intuitiver umzusetzen als bei den bekannten Alternativen.

Das Programmieren der Anwendungsbeispiele wurde mit der Sprache Python und den bekannten Frameworks Numpy und Scipy umgesetzt. Diese Entscheidung wurde hauptsächlich durch die große Flexibilität dieser Sprache und der ausreichenden Leistung der Bibliotheken beeinflusst. Die hierbei entstandenen Grafiken wurden mit der Pyplot Bibliothek umgesetzt.

Während des Praktikums wurde für die Literaturrecherche der Dienst Google Scholar benutzt. Dies liegt daran, dass er nicht nur eine umfassende Datenbank besitzt, sondern auch, dass das Erhalten der indizierten Werke sehr einfach ist.
Die benutzte Literatur wurde über BibTeX direkt in den LaTeX-Quellcode des Berichts eingebunden und zitiert.

Dieses gesamte Vorgehen hat sich im Laufe des Praktikums als gut erwiesen und wird somit auch in der anschließenden Bachelorarbeit weiterverfolgt werden.