GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Fakultät für
Physik

Bachelorarbeit

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

Zweitgutachter/in: nicht vorhanden

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

Symbol	Bedeutung
N	Anzahl der Einheiten im Reservoir $\mathbf{W}$
α	Verlustrate des *Leaky Integrators*
β	Parameter für den Lernvorgang mittels *Tikhonov Regularisierung.*
$\rho(\mathbf{W}))$	Spektralradius von $\mathbf{W}$
$\sigma(\mathbf{W})$	Singulärwert von $\mathbf{W}$
ν_{max}	Maximalwert des hinzugefügten Rauschens $\nu(n)$ für die Erhöhung der Stabilität.
$[\cdot\,;\,\cdot]$	Vertikales Aneinanderfügen von Vektoren/Matrizen

1 Einführung

2 Theorie

2.1 Berkley Modell

Das *Barkley Modell*, welches 1990 von Dwight Barkley vorgestellt wurde, ist ein System aus gekoppelten Reaktionsdiffusionsgleichungen. Dies sind partielle Differentialgleichungen (*PDE*) zweiter Ordnung welche einen Diffusionsterm besitzen. Das *Barkley Modell* beschreibt ein erregbares und oszillierendes Medium. Das Modell besteht aus zwei Variablen u, v die den *PDEs*

$$\begin{aligned} \frac{\partial u}{\partial t} &= D \cdot \nabla^2 u + \frac{1}{\epsilon}(1-u)\left(u - \frac{v+b}{a}\right) \\ \frac{\partial v}{\partial t} &= u^\alpha - v, \end{aligned} \tag{2.1}$$

unterliegen. Dabei ermöglicht $\alpha = 1$ dem System *periodische* Wellenmuster auszubilden und $\alpha = 3$ bedingt ein *chaotisches* Verhalten. Die Variable u durchläuft hierbei eine schnellere Dynamik als die hemmende Variable v [1, 2].
Die Parameter ϵ, b und a charakterisieren das Verhalten des Systems und werden in der gesamten folgenden Arbeit - sofern keine anderen Angaben existieren - als

$$\begin{aligned} a &= 0.8, \\ b &= 0.01, \\ \epsilon &= 0.02 \end{aligned}$$

festgelegt.
Zudem wird das Modell in dieser Arbeit in zwei Dimensionen betrachtet, sodass $u(t, x, y)$ und $v(t, x, y)$ skalare zeitabhängige Felder sind.
Zur Simulation des System werden zunächst die *PDE*s zeitlich mit einem Zeitschritt Δt und örtlich mit einer Gitterkonstante Δx diskretisiert. Zur Beschreibung des

Diffusionstermes wird eine fünf-Punkte Methode

$$\nabla^2 u(t)_{i,j} \simeq u(t)_{i-1,j} + u(t)_{i+1,j} + u(t)_{i,j-1} + u(t)_{i,j+1} - 4u(t)_{i,j} =: \Sigma(t)_{i,j} \tag{2.2}$$

nach [1] verwendet. Dabei stehen die tiefergestellten Indizes für den diskretisierten Ort der *x-y-Ebene*. Für kleine Zeitschritte Δt ist ein *explizites Eulerverfahren*

$$\begin{aligned} u(t+1)_{i,j} &= u(t)_{i,j} + \Delta t \cdot \frac{\partial u}{\partial t}, \\ v(t+1)_{i,j} &= v(t)_{i,j} + \Delta t \cdot \frac{\partial v}{\partial t} \end{aligned} \tag{2.3}$$

mit

$$\begin{aligned} \frac{\partial u}{\partial t} &= D \cdot \Sigma(t)_{i,j} + \frac{1}{\epsilon}(1 - u(t)_{i,j}) \left(u(t)_{i,j} - \frac{v(t)_{i,j} + b}{a} \right) \\ \frac{\partial v}{\partial t} &= u(t)_{i,j}^3 - v(t)_{i,j} \end{aligned} \tag{2.4}$$

ausreichend genau. Im Folgenden wird zudem, in Analogie zu [2], die Diffusionskonstante auf $D = 1/50$, die Gitterkonstante auf $\Delta x = 0.1$ und die Zeitkonstante auf $\Delta t = 0.01$ gesetzt.

2.2 Delay Coordinates

2.3 Nächste Nachbar Vorhersage

Das Ziel der *nächsten Nachbar Vorhersage* ist es den funktionalen Zusammenhang $F : X \to Y$ zu finden, welcher Daten der Menge $X \in \mathbb{R}^n$ auf Elemente aus $Y \in \mathbb{R}^m$ eindeutig abbildet. Hierfür wird angenommen, dass die Funktion F lokal stetig ist. Zudem werden hierfür Daten benötigt, anhand derer der Zusammenhang erlernt werden kann.
Zu Beginn werden Paare $(\vec{x}, \vec{y}) \in X \times Y$ aus einem *Trainingsdatensatz* gebildet und eine Suchstruktur über die x-Werte gebildet. Nun kann diese Struktur genutzt werden, um für ein gegebenes $\vec{x}$ den wahrscheinlichsten Wert $\vec{y}$ zu suchen. Hierfür werden, unter der Annahme der lokalen Stetigkeit, die Datenpunkte $\vec{x}_1, ..., \vec{x}_k$ aus der zuvor angelegten Suchstruktur ausgewählt, welche den geringsten Abstand $d(\vec{x}, \vec{x_i})$ zu $\vec{x}$ besitzen.
Diesen k Datenpunkten ist jeweils ein eindeutiger Wert $\vec{y}_i$ zuvor zugeordnet werden.

Damit kann nun eine Approximation für den zu $\vec{x}$ gehörigen Wert $\vec{y}$ erstellt werden, indem beispielsweise der Mittelwert der $\vec{y_i}$ genutzt wird.

Der Schlüssel in der Bewältigung einer solchen Aufgabe liegt hauptsächlich in der Art und Weise, wie die nächsten Nachbarn gesucht werden. Hierbei werden im Folgenden die beiden Algorithmen $k-d-tree$ und $ball-tree$ ebenso betrachtet wie ein naiver Ansatz. Bei diesem naiven Ansatz (*brute force*) werden die nächsten Nachbarn aus dem unsortierten Trainingsdatensatz durch pures Ausprobieren aller Möglichen Punkte ermittelt.

2.3.1 k-d-tree

2.3.2 ball-tree

2.4 Radiale Basisfunktionen

Eine weitere Methode um einen solchen funktionalen Zusammenhang $F : X \to Y$ zu finden, welcher Daten der Menge $X \in \mathbb{R}^n$ auf Elemente aus $Y \in \mathbb{R}^m$ eindeutig abbildet, bieten die *radialen Basisfunktionen* an. Auch hierfür Daten benötigt, anhand derer der Zusammenhang erlernt werden kann. Diese Trainingsdaten sollen im Folgenden aus N Datensätzen bestehen.

Bei diesem Ansatz wird die gesuchte Funktion F als Linearkombination aus vielen radialen Funktionen approximiert. Dafür werden l Elemente $\{\vec{x}_i\}, i = 1, ..., l$ aus den Trainingsdaten ausgewählt und diese als so genannte *Zentren* $\{\vec{z}_i\}$ genutzt. Hiermit lassen sich die Funktionen als $\phi_i(\vec{x}) = \phi(||\vec{x} - \vec{z}_i||), i = 1, \ldots, l$ darstellen. Die dabei auftretende Norm $||\ldots|||$ wird im Folgenden als euklidisch angenommen [4]. Eine mögliche Wahl der Basisfunktionen sind zum Beispiel Gaußfunktionen

$$\phi_i(\vec{x}) = \exp\left(-\frac{||\vec{x} - \vec{z}_i||}{\sigma_i^2}\right).$$

Die Linearkombination führt zu dem Ansatz

$$\vec{y} = F(\vec{x}) = \sum_{i=1}^{l} \vec{\omega}_i \phi(||\vec{x} - \vec{z}_i||). \tag{2.5}$$

Die $\vec{\omega}_i \in \mathbb{R}^m$ stehen hierbei für die *Gewichtsvektoren* der einzelnen Basisfunktionen

ϕ_i im Rahmen der Linearkombination.

Das Ziel besteht jetzt darin die Gewichtsvektoren ω_i approximativ zu bestimmen. Dafür werden zunächst drei Matrizen definiert, durch die das Problem ausgedrückt werden kann.
Die Matrix $\mathbf{Y} \in \mathbb{R}^{N \times m}$ repräsentiert die Funktionswerte der Abbildung und beinhaltet als Zeilen die N verschiedenen Funktionswerte $\vec{y}_i$ der Trainingsdaten

$$\mathbf{Y} := \begin{pmatrix} y_{11} & \dots & y_{1m} \\ \vdots & & \vdots \\ y_{N1} & \dots & y_{Nm} \end{pmatrix}. \tag{2.6}$$

Die Matrix $\mathbf{\Omega} \in \mathbb{R}^{l \times m}$ beinhaltet dagegen als Zeilen die Gewichtsvektoren

$$\mathbf{\Omega} := \begin{pmatrix} \omega_{11} & \dots & \omega_{1m} \\ \vdots & & \vdots \\ \omega_{l1} & \dots & \omega_{lm} \end{pmatrix}. \tag{2.7}$$

Die dritte Matrix $\mathbf{A} \in \mathbb{R}^{N \times m}$ repräsentiert Anwendungen der radialen Basisfunktionen auf die Trainingsdaten

$$\mathbf{A} := \begin{pmatrix} A_{11} & \dots & A_{1m} \\ \vdots & & \vdots \\ A_{N1} & \dots & A_{Nm} \end{pmatrix}, \tag{2.8}$$

wobei die einzelnen Elemente als $A_{ij} := \phi(||\vec{x}_i - \vec{y}_j||)$ definiert sind.
Damit lässt sich das Problem durch

$$\mathbf{Y} = \mathbf{A} \cdot \mathbf{\Omega} \tag{2.9}$$

ausdrücken [4]. Da die Matrizen $\mathbf{Y}$ und $\mathbf{A}$ konstruiert sind, besteht die Aufgabe lediglich darin die Matrix $\mathbf{\Omega}$ der Gewichte zu ermitteln. Der naheliegende Ansatz, das direkte ermitteln der Inversen $\mathbf{A}^{-1}$ stellt sich dafür aus ungeeignet heraus, da das Problem meistens überkonditioniert ist. Daraus ergibt sich eine schlechtere Voraussage der zukünftigen Funktionswerte. Stattdessen ist es geschickter das Problem als eine lineare Optimierungsaufgabe zu betrachten, bei der der Fehler $||\mathbf{A}\omega_i - \vec{y}_i||^2$ minimiert werden soll.

Durch die Verwendung der *Moore-Penrose Pseudoinversen* $\mathbf{A}'$ wird hierbei zugleich gewährleistet, dass die Lösung ausgewählt wird, die zudem auch die kleinsten Gewichte besitzt. Dies hilft den Effekt des *Overfittings* zu vermeiden [4]. Mit diesem Ansatz ergibt sich die Lösung zu

$$\mathbf{\Omega} = \mathbf{A}' \cdot \mathbf{Y}. \tag{2.10}$$

Um nun Funktionswerte vorherzusagen wird der Zusammenhang aus für die zuvor ermittelten Gewichte genutzt.

2.5 Neural Networks

2.5.1 Überblick über 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.

2.5.2 Feed Forward Neural Networks

Ein FFNN besteht aus mehreren Ebenen, welche jeweils aus verschiedenen nichtlinearen Einheiten zusammengesetzt sind. Die erste dieser Ebenen wird zur Eingabe und die letzte zur Ausgabe eines Signals 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, müssen die Gewichte in einem Trainingsvorgang angepasst werden. Dies wurde durch die Entwicklung des Backpropagation-Algorithmus stark vereinfacht. Hierbei werden die Gewichte so angepasst, dass eine Kostenfunktion minimiert wird [3, S. 225-290].

2.5.3 Recurrent Neural Networks

Ein RNN hat einen ähnlichen Aufbau, doch hier können 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. Ein Nachteil ist, dass zum Trainieren nicht mehr der einfachere Backpropagation-Algorithmus Algorithmus genutzt werden kann, sondern eine für RNNs abgewandelte Form genutzt werden muss. Dafür werden die verschiedenen Zustände die das RNN im Laufe der Signal-Propagation annimmt nacheinander betrachtet und auf diese zeitliche Entwicklung anschließend der Backpropagation-Algorithmus angewendet. Diese Methode ist unter dem Namen Backpropagation through Time (BTT) bekannt. Sie ist zum einen rechenaufwendiger und zum anderen auch instabiler, da das Verschwindenden des Gradienten der Kostenfunktion deutlich wahrscheinlicher als bei der gewöhnlichen Backpropagation ist [11].

2.6 Echo State Network

2.6.1 Überblick

Um die (Leistungs)Probleme der RNN zu umgehen, wurden als mögliche Lösung die Echo State Networks von H. Jäger vorgeschlagen [5]. Etwa zeitgleich wurde von W. Maas das Modell der *Liquid State Machines* vorgeschlagen. In diesem Modell steht der biologische Hintergrund im Fokus, doch sind die Ergebnisse denen der Echo State Networks sehr ähnlich [10].

2.6.2 Aufbau

Ein ESN ist eine Spezialform eines RNNs. Hierbei wird eine auf dem ersten Blick eigenartige Entscheidung getroffen: Während des gesamten Trainingsvorganges werden die Verbindungen der einzelnen Einheiten größtenteils nicht verändert. Es wird versucht durch das *Echo* der vorherigen Signale, welche noch im Netzwerk gespeichert sind, diese Signale zu rekonstruieren - hieraus ergibt sich auch der Name [9]. Im Folgenden wird der Aufbau und anschließend die Funktionsweise eines solchen Netzwerkes nach [8] beschrieben.

Allgemein bildet das Netzwerk S ein zeitliches Signal $\vec{u}(n) \in \mathbb{R}^{N_u}$ auf eine zeit-

lich 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 werden 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

$$\vec{s}(n) = (1-\alpha) \cdot \vec{x}(n-1) + \alpha \cdot f_{in}\left(\mathbf{W}_{in}[b_{in}; \vec{u}(n)] + \mathbf{W}\vec{x}(n-1)\right), \tag{2.11}$$

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

$$\vec{s}(n) = f_{in}\left(\mathbf{W}_{in}[b_{in}; \vec{u}(n)] + \mathbf{W}\vec{x}(n-1)\right). \tag{2.12}$$

Da für manche Anwendungsfälle auch eine direkte Rückkopplung wünschenswert ist, kann 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

$$\vec{s}(n) = (1-\alpha) \cdot \vec{x}(n-1)\alpha \cdot f_{in}\left(\mathbf{W}_{in}[b_{in}; \vec{u}(n)] + \mathbf{W}\vec{x}(n-1) + \mathbf{W}_{fb}\vec{y}(n)\right) \tag{2.13}$$

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. Im Folgenden wird nur der Fall einer Linearkombination betrachtet, da sich für die an-

deren Methoden ein analoges Verfahren ergibt. In diesem Fall berechnet sich die Ausgabe mittels

$$\vec{y}(n) = f_{out}\left(\mathbf{W}_{out}\vec{x}(n) = \mathbf{W}_{out}[b_{out}; \vec{s}(n); \vec{u}(n)]\right), \tag{2.14}$$

wobei f_{out} die Transferfunktion der Ausgabe ist.

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 zufällige Anordnung von zufälligen Gleitkommazahlen zwischen -1.0 und 1.0 als geschickt herausgestellt. Falls ein Feedback gewünscht ist, also Gleichung (2.13) genutzt wird, wird $\mathbf{W}_{fb}$ gleichartig konstruiert. Auf das Generieren der inneren Matrix $\mathbf{W}$ wird in Abschnitt 2.6.4 genauer eingegangen.

2.6.3 Trainingsvorgang

Nachdem der Aufbau des Netzwerkes beschrieben ist, ergibt sich nun die Frage, wie der Trainingsvorgang durchgeführt wird.

Hierfür wird für die Zeiten $n = 0, ..., T_0$ das ESN mit dem Signal $\vec{u}(n)$ betrieben, wobei T_0 die *transiente Zeit* beschreibt. Hierdurch soll das System aus seinem zufällig gewähltem Anfangszustand in einen charakteristischen Zustand übergehen. Anschließend wird das System für Zeiten $n < T$ weiter 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

$$\mathbf{Y} = \mathbf{W}_{out}\mathbf{X} \tag{2.15}$$

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* mittels der Regularisierung $\beta \cdot ||\vec{W}_{out,i}||^2$ der Gewichtsmatrix mit der Konstante β erhalten werden. Hierbei steht $\vec{W}_{out,i}$ für die jeweils i-te Zeile der

Gewichtsmatrix. Das Verfahren

$$\mathbf{W}_{out} = \mathbf{Y}\mathbf{X}^T \left(\mathbf{X}\mathbf{X}^T + \beta I\right)^{-1} \tag{2.16}$$

ist sehr leistungsstark, aber auch teilweise numerisch instabil. Bei geeigneter Wahl von β können die besten Ergebnisse hinsichtlich der Genauigkeit der Vorsage erzielt werden [9].

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

$$\mathbf{W}_{out} = \mathbf{Y}\mathbf{X}' \tag{2.17}$$

folgt. Dieses Verfahren ist zwar sehr rechenaufwendig aber dafür numerisch stabil [7, 9]. Nichts desto trotz, kann allerdings auf Grund des Fehlens einer Regularisierung leicht der Effekt des Overfittings auftreten. Dieser kann umgangen werden, indem in der Zustandsgleichung (2.11) beziehungsweise (2.13) eine leichte normalverteilte Störung $\vec{\nu}(n)$ der Größenordnung 1×10^{-1} bis 1×10^{-5} addiert wird. Dieser Ansatz beruht auf Empirie, da eine mathematische Begründung hierfür noch nicht vollständig gelungen ist [5, 9].

Zusammenfassend ergibt sich somit der folgende Funktionsablauf für die Anwendung 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 $n > T$

2.6.4 Theoretischer Hintergrund

Um die mathematischen Eigenschaften beschreiben zu können, sind zuerst zwei Definitionen nötig [12].

Definition 1 (Kompatibler Zustand)**.** *Sei* $S : X \times U \to X$ *ein* ESN *mit der Zustandsgleichung* $\vec{x}_{n+1} = F(\vec{x}_n, \vec{u}_{n+1})$*. Eine Folge von Zuständen* $(\vec{x}_n)_n$ *ist kompatibel mit der Eingangsfolge* $(\vec{u}_n)_n$*, wenn* $\vec{x}_{n+1} = F(\vec{x}_n, \vec{u}_{n+1}), \forall n \leq 0$ *erfüllt ist.*

Definition 2 (Echo State Eigenschaft (ESP))**.** *Ein* ESN $S : X \times U \to X$ *besitzt die Echo State Eigenschaft genau dann wenn eine Nullfolge* $(\delta_n)_{n\geq 0}$ *existiert, sodass für alle Zustandsfolgen* $(\vec{x}_n)_n, (\vec{x}'_n)_n$ *die kompatibel mit der Eingangsfolge* $(\vec{u}_n)_n$ *sind gilt, dass* $\forall n \geq 0 ||x_n - x'_n|| < \delta_n$

Dies bedeutet, dass nachdem das Netzwerk lang genug betrieben worden ist, der Zustand nicht mehr von dem beliebig gewähltem Anfangszustand abhängt. Diese Eigenschaft ist notwendig, damit das ESN Vorhersagen treffen kann [6].

Nun stellt sich die Frage, wann ein Netzwerk diese Eigenschaft besitzt. Es wird schnell klar, dass dies nur durch die Gewichtsmatrix $\mathbf{W}$ bestimmt wird. Betrachtet man die Zustandsgleichung des Netzwerkes, so lässt sich auf Grund des *Banachschen Fixpunktsatzes* erkennen, dass die *ESP* für alle Eingänge $\vec{u}_n$ vorliegt, sobald $||\vec{x}_{n+1} - \vec{x}'_{n+1}|| < ||\vec{x}_n - \vec{x}'_n||$ für zwei kompatible Zustände $\vec{x}_n \neq \vec{x}'_n$ erfüllt ist [5]. Hieraus ergibt sich, dass die *ESP* vorliegt, wenn

$$|1 - \alpha(1 - \sigma_{max}(\mathbf{W}))| < 1 \tag{2.18}$$

erfüllt ist, wobei $\sigma_{max}(\mathbf{W})$ der größte Singulärwert ist [8].
Weitergehend ist bekannt, dass für Systeme bei denen der Spektralradius $\rho(\mathbf{W}) > 1$ ist diese Eigenschaft nicht vorliegen kann, sofern $\vec{u}_n = 0$ möglich ist [5, 8].

Hieraus ergab sich lange Zeit die falsche Annahme, dass für Systeme mit $\rho(\mathbf{W}) < 1$ die Eigenschaft stets garantiert ist. Wie allerdings gezeigt werden konnte, ist dies nicht der Fall [12]. Stattdessen konnte gezeigt werden, dass eine hinreichende Bedingung durch

$$\rho(\alpha|\mathbf{W}| + (1 - \alpha)\mathbf{I}) < 1 \tag{2.19}$$

gegeben ist - wobei als Betrag der Matrix hier das elementweise Betragsnehmen gemeint ist. Diese Bedingung ist weniger einschränkend als Gleichung (2.18) [12].

Darauf basierend kann nun eine Methode nach [12] angegeben werden, um die

Gewichtsmatrix $\mathbf{W}$ zu konstruieren:

1. Generiere zufällige Matrix $\mathbf{W}$ mit $|\mathbf{W}| = \mathbf{W}$
2. Skaliere $\mathbf{W}$, sodass Gleichung (2.19) erfüllt ist.
3. Wechsel zufällig das Vorzeichen von ungefähr der Hälfte aller Einträge.

Statt dieser Vorschrift wurde zuvor oftmals $\mathbf{W}$ zufällig generiert und anschließend nur $\rho(\mathbf{W})$ statt $\rho(|\mathbf{W}|)$ skaliert, was mit unter zu instabilen Systemen geführt hat. Da allerdings auch für Systeme mit einem Spektralradius > 1 die *ESP* beobachtet werden kann für nicht verschwindende Eingänge $\vec{u}_n$, ist es ratsam auch effektive Spektralradien jenseits 1 auszuprobieren.

3 Anwendungen

4 Diskussion

5 Fazit

6 Ausblick

7 Danksagungen

Literaturverzeichnis

[1] D. Barkley. Barkley model. *Scholarpedia*, 3(11):1877, 2008. doi: 10.4249/scholarpedia.1877. revision #91029.

[2] Sebastian Berg, Stefan Luther, and Ulrich Parlitz. Synchronization based system identification of an extended excitable system. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 21(3):033104, 2011.

[3] Cristopher M. Bishop. *Pattern Recognition andMachine Learning*. Springer, Cambridge, 2006. ISBN 0-387-31073-8.

[4] D.S. Broomhead and D Lowe. Multi-variable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.

[5] H. Jäger. The "echo state" approach to analysing and training recurrent neural networks - with an erratum note. *GMD Report*, 148, 2001/2010.

[6] H. Jäger. A tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach. *GMD Report*, 159:48 ff., 2002.

[7] H. Jäger. Long short-term memory in echo state networks: Details of a simulation study. Technical report, Jacobs University Bremen - School of Engineering and Science, 2012.

[8] H. Jäger, M. Lukoševičiusa, D. Popovici, and U. Siewert. Optimization and applications of echo state networks with leaky- integrator neurons. *Neural Networks*, 20:335–352, 2007.

[9] M. Lukoševičiusa and H. Jäger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[10] Wolfgang Maass. Liquid State Machines: Motivation, Theory, and Applications. In *Computability in Context*, pages 275–296. Imperial College Press, 2011.

[11] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on Machine Learning*, 28, 2013.

[12] I. Yildiz, H. Jäger, and S. Kiebel. Re-visiting the echo state property. *Neural Networks*, 35:1–9, 2012.