GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Fakultät für Physik

Bachelorarbeit

Vorhersage raumzeitlicher Dynamik mittels Reservoir Computing

Preidicting spatio-temporal dynamics using reservoir computing

angefertigt von

Roland Simon Zimmermann

aus Recklinghausen

am Max-Planck-Institut für Dynamik und Selbstorganisation

Bearbeitungszeit: 1. Juni 2017 bis 15. September 2017

Erstgutachter/in: Prof. Dr. Ulrich Parlitz

Zweitgutachter/in: Prof. Dr. Florentin Wörgötter

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*
λ	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.
σ	
$\Delta\sigma$	
σ_{RBF}	Breite der gaußschen Basisfunktion
δ	Dimension der Delay Reconstruction
$[\cdot\,;\,\cdot]$	Vertikales Aneinanderfügen von Vektoren/Matrizen

1 Einführung

Im Zuge der fortschreitenden technologischen Weiterentwicklung wurden Wissenschaftler und Ärzte mit neuen Methoden ausgestattet um Herzen zu untersuchen. So ist es möglich bei sogenannten *in vitro* Experimenten ein Herz außerhalb des eigentlichen Körpers schlagen zu lassen, sodass sein Verhalten studiert werden kann. Dabei wird die Dynamik des Herzmuskels durch unterschiedliche Ionenarten beeinflusst. Diese erzeugen einen Stromfluss sowohl im Inneren als auch auf der Oberfläche des Herzens. Mit den aktuellen Techniken ist es möglich bei solchen Experimenten die mechanische Kontraktion des Herzen und auch den Spannungsverlauf auf der Oberfläche zu messen. Doch Informationen über die Ströme und Spannungen im Innersten des Herzens sind aufgrund der erschwerten Zugänglichkeit kaum messbar. Ebenso können kaum Ströme einzelner bestimmter Ionenarten vermessen werden.

Im Rahmen dieser Arbeit wird versucht eine Möglichkeit zu geben, um sowohl die im Inneren auftretenden Spannungen als auch weitere schwer messbare (verborgene) Systemvariablen mittels *Machine Learning* (maschinelles Lernen) zu approximieren. Dafür wird das *Reservoir Computing* anhand der *Echo State Networks* angewendet und anschließend mit bereits länger bestehenden Methoden verglichen. Dies sind die Approximation mittels der *nächsten Nachbarn* und *radialer Basisfunktionen.*

Da die Anwendung solcher Techniken auf echte Messdaten mit einigen Hindernissen verbunden ist, werden zuerst zwei Modellsysteme betrachtet. Diese Modelle sind in der Vergangenheit entwickelt worden, um eine mathematische Beschreibung der Funktionsweise der Herzen zu geben und ihre Dynamiken zu beschreiben. In dieser Arbeit werden das *Barkley*- und das *Mitchell-Schaeffer*-Modell untersucht.

Am Anfang wird ein theoretischer Überblick in Kapitel 2 gegeben. Dort werden zunächst die beiden verwendeten Modelle eingeführt und beschrieben, und im Anschluss die beiden klassischen Methoden vorgestellt. Darauffolgend wird in Kapitel

2.7 die Technik der *Echo State Networks* eingeführt und beschrieben. In Kapitel 3 werden diese Ansätze nun an drei verschiedenen Szenarien getestet. Zuerst wird eine verborgene Variable auf Grundlage einer gemessenen Variable approximiert in Kapitel 3.2. Danach werden die Ansätze genutzt um die Qualität der gemessenen Spannungsverläufe in 3.3 zu erhöhen. Darauffolgenden wird in Abschnitt 3.2 die Spannung im Inneren eines solchen Systems vorhergesagt.

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, 4].
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

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

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

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

ausreichend genau. Im Folgenden wird zudem, in Analogie zu [4], die Diffusionskonstante auf $D = 1/25$, die Gitterkonstante auf $\Delta x = 0.1$ und die Zeitkonstante auf $\Delta t = 0.01$ gesetzt.

2.2 Mitchell-Schaeffer Modell

Das *Mitchell-Schaeffer Modell* ist ebenso wie das *Barkley Modell* ein System aus gekoppelten partiellen Differentialgleichungen. Es ist vorgeschlagen worden, um eine phänomenologisches Beschreibung der Aktionspotentiale auf der Membran von Herzzellen zu liefern. Das Modell wird durch die Membranspannung $v(t)$ und eine Kontrollvariable $h(t)$, welche das Verhalten der beteiligten Ionenkanäle steuert, definiert. Hierbei wird die Spannung als dimensionslose Größe dargestellt, die Werte zwischen 0 und 1 annehmen kann [15].

Diese Dynamik wird durch die Gleichungen

$$\begin{aligned} \frac{\partial v}{\partial t} &= \nabla \cdot (D \nabla v) + \frac{h v^2 (1-v)}{\tau_{in}} - \frac{v}{\tau_{out}}, \\ \frac{\partial h}{\partial t} &= \begin{cases} \frac{1-h}{\tau_{open}}, & \text{wenn } v \leq v_{gate} \\ \frac{-h}{\tau_{close}}, & \text{wenn } v \geq v_{gate} \end{cases} \end{aligned} \tag{2.5}$$

beschrieben. Dabei stehen die Parameter $\tau_{in}, \tau_{out}, \tau_{open}, \tau_{close}$ für Zeitkonstanten, welche die Form des Aktionspotentials modifizieren. Die ersten beiden Konstanten beschreiben die Geschwindigkeit, mit der die Ionen durch die Membran strömen, und die letzten beiden die Geschwindigkeit mit der sich die verantwortlichen Ionenkanäle öffnen beziehungsweise schließen. Zusätzlich stellt die Konstante v_{gate} eine Grenzspannung dar. Beim Über- und Unterschreiten dieser Grenze ändert sich der der jeweilige Zustand der Ionenkanäle , indem $h(t)$ angepasst wird. Im Rahmen dieser Arbeit werden, soweit keine anderen Angaben vorhanden sind, die Parameter durch die Werte aus Tabelle 2.1 in Analogie zu [15] festgesetzt. Dabei ist allerdings τ_{open} auf 20 [2, S. 134ff.] reduziert worden, da mit dieser Wahl ein chaotischeres Verhalten, ähnlich zum *Barkley-Modell*, erzeugt wird. Dies erschwert die mögliche Vorhersage der Entwicklung, wodurch eine anspruchsvolle Herausforderung erzeugt wird.

τ_{in}	τ_{out}	τ_{open}	τ_{close}	v_{gate}
0.3	6.0	20	150	0.13

Tab. 2.1: Verwendete Zeitkonstaten und Grenzspannung v_{gate} für die Betrachtung des *Mitchell-Schaeffer Modells*

Der erste Summand der zeitlichen Ableitung von v beschreibt ein Diffusionsverhalten, welches durch die Diffusionsmatrix $\mathbf{D} = \text{diag}(D_x, D_y)$ beschrieben wird. Die Einführung dieser Matrix erlaubt im Allgemeinen die Verwendung von zwei verschiedenen Diffusionskontanten D_x, D_y, welche Richtungsabhängig sind [17]. Im Folgenden wird für diese allerdings der gleichen Wert $D_x = D_y = D$ gesetzt.

Die meisten, auf zellulärer Ebene aufgestellten, Gleichungen haben eine hohe Komplexität. Hierdurch werden numerische Berechnungen sehr aufwendig. In der Herleitung dieses Modells sind einige vereinfachende Annahmen eingeflossen, wodurch die Komplexität und somit auch der numerische Aufwand reduziert worden sind. Trotz des phänomenologischen Charakters des *Mitchell-Schaeffer Modells* besitzen die Parameter eine physiologische Interpretation. Zudem ist es in der Lage wichtige Eigenschaften des Aktionspotentials im Vergleich zu anderen Modellen gut wiederzugeben [17].

Analog zu der Betrachtung des *Barkley Modells* sind für die numerische Betrachtung die beiden *PDE*s erneut in einem expliziten Verfahren mittels

$$\frac{\partial v}{\partial t}_{i,j} = D \cdot \Sigma(t)_{i,j} + \frac{h(t)_{i,j} v(t)_{i,j}^2 (1 - v(t)_{i,j})}{\tau_{in}} - \frac{v(t)_{i,j}}{\tau_{out}}$$
$$\frac{\partial h}{\partial t}_{i,j} = \begin{cases} \frac{1-h(t)_{i,j}}{\tau_{open}}, & \text{wenn } v(t)_{i,j} \leq v_{gate} \\ \frac{-h(t)_{i,j}}{\tau_{close}}, & \text{wenn } v(t)_{i,j} \geq v_{gate} \end{cases} \tag{2.6}$$

diskretisiert worden. Dabei werden im Folgenden die Integrationskonstanten $\Delta x = 0.1, \Delta t = 0.01$ und die Diffusionskonstante $D_x = D_y = D = 5 \times 10^{-3}$ genutzt.

2.3 Delay Reconstruction

Die *Delay Reconstructions* (Verzögerung-Konstruktionen) können benutzt werden um Zeitreihen zu analysieren und den Phasenraum des ursprünglichen Systems zu rekonstruieren. Hierbei wird ein Signal $s(t)$ an diskreten Zeitpunkten betrachtet, sodass sich das diskrete Signal $s_n = s(n\Delta t)$ ergibt. Eine *Delay Reconstruction* erzeugt hier raus ein Signal, in welchem die Informationen δ vorheriger Zeitpunkte mit dem Abstand τ enthalten sind. Somit wird eine höhere dimensionale Zeitreihe $\vec{z}_n \in \mathbf{R}^\delta$ durch

$$\vec{z}_n = \left(s_{n-(\delta-1)\tau}, s_{n-(\delta-2)\tau}, \ldots, s_n\right) \tag{2.7}$$

konstruiert. Bei einer ausreichend hohen Wahl der Rekonstruktionsimension m ist es hiermit möglich den Phasenraum des Attraktors zu rekonstruieren. Für die Wahl der Verzögerungszeit τ gibt es keine rigorose mathematische Definition oder Beschreibung, sondern es existieren verschiedene Ansätze zur Ermittlung des optimalen Wertes. Ein populärer Ansatz, welcher in dieser Arbeit verwendet besteht darin τ durch das Auffinden der ersten Nullstelle von der Autokorrelationsfunktion

$$AUC(\tau) = \sum_{l}^{N-\tau} (s_l - \bar{s})(s_{l+\tau} - \bar{s}) \tag{2.8}$$

zu ermitteln. Dies lässt sich dadurch motivieren, dass durch das hinzunehmen des Signals der Zeitreihe die um diesen Wert von τ verschoben ist, am meisten neue Information hinzugefügt wird, da die Selbstähnlichkeit des Signals am geringsten ist [12, 30 ff.]

2.4 Nächster Nachbar Vorhersage

Das Ziel der *nächsten Nachbar Vorhersage* (im Folgenden als NN-Ansatz abgekürzt) 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 ein gewichteter Mittelwert der $\vec{y}_i$ genutzt wird. Hierzu kann jedem der k nächsten Nachbarn $(\vec{x}_i, \vec{y}_i)$ ein Gewicht $w_i(\vec{x})$ nach

$$w_i(\vec{x}) = \frac{v_i}{\sum_j v_j}, \text{ mit } v_i = \frac{1}{||\vec{x}_i - \vec{x}||}$$

zugeordnet werden. Diese Definition erfüllt $\sum_i w_i = 1$ und ordnen zudem fernen Nachbarn ein geringeres Gewicht zu. Die dabei auftretende Norm $|| \dots |||$ wird im Folgenden als euklidisch angenommen, sofern keine weiteren Angaben existieren. Somit ergibt sich für die gewichtete Vorhersage

$$F(\vec{x}) = \vec{y} = \sum_i^k \vec{y}_i \left(\sum_j \frac{||\vec{x}_i - \vec{x}||}{||\vec{x}_j - \vec{x}||} \right)^{-1} . \tag{2.9}$$

Der Schlüssel in der Bewältigung einer solchen Aufgabe liegt hauptsächlich in der Art und Weise, wie die k nächsten Nachbarn gesucht werden. Hierbei wird im Folgenden der Algorithmus K-D-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.

2.4.1 k-d-tree

Ein K-D-TREE ist eine Spezialform eines Binärbaumes, und eine oftmals genutzte Methode um Suchvorgänge in Datenstrukturen durchzuführen. Das zugrundeliegende Prinzip eines solchen Baumen ist, dass wenn der Punkt P_1 weit entfernt von P_2 liegt, aber P_3 nahe an P_2 liegt, so folgt daraus, dass P_1 und P_3 weit voneinander entfernt liegen müssen. Durch eine solche Argumentation muss bei einem solchem Suchvorgang die Distanz zweier Punkte seltener berechnet werden, wodurch Rechenzeit eingespart werden kann.

Der Suchvorgang besteht aus zwei Phasen. Zuerst wird die Aufbauphase des Baumes durchgeführt, bei der die Trainingsdaten einsortiert und damit ein Suchindex erzeugt wird. Anschließend folgt die Suchphase, bei der der zuvor erstellte Suchindex nach dem nächsten Nachbarn durchsucht wird.

In der Aufbauphase wird zuerst eine Dimensionen ausgewählt und der Median der Daten in dieser Dimension bestimmt. Anschließend werden die Datenpunkte anhand dieses Wertes in eine Menge unterteilt die nur größere oder nur kleinere Elemente bezogen auf jene Dimension beinhaltet. Die beiden Mengen bilden die ersten Äste des Baumes. Nun wird dieser Schritt rekursiv auf alle Äste angewendet, und die hierbei zum Vergleich genutzte Dimension iteriert [7]. Dieses Verfahren wird so lange wiederholt, bis eine bestimmte maximale Anzahl N_{max} an Knotenpunkten pro Ast erreicht wird. Ab dieser unteren Grenze wird das Erstellen des Binärbaumes beendet. Ab dieser Grenze benötigt der Zugriff auf die verschiedenen Elemente und das Aufteilen in neue Äste mehr Zeit, als das Berechnen der Abstände zwischen den verbleibenden N_{max} Knoten und dem Suchpunkt. Eine beispielhafte Darstellung des Verfahrens ist in Abbildung 2.1 zu finden.

Die Suchphase wird nun wieder rekursiv durchgeführt. Hierbei werden wieder iterierend die verschiedenen Dimensionen verglichen, und sich somit immer weiter im Suchbaum nach unten ein Weg gebahnt [7]. In der untersten Ebene, also wenn nur noch eine Suche zwischen maximal N_{max} Elemente durchgeführt werden muss, wird nun die *brute force*-Suche genutzt. In dieser Arbeit ist für alle Anwendungen diese Schwelle auf $N_{max} = 40$ gesetzt worden.

Diese Methode zeichnet sich durch eine Laufzeit, welche sich wie $\mathcal{O}(\log(N))$ verhält, aus [3]. Dies ist geringer, als die Laufzeit eines naiven Suchvorgangs, welche

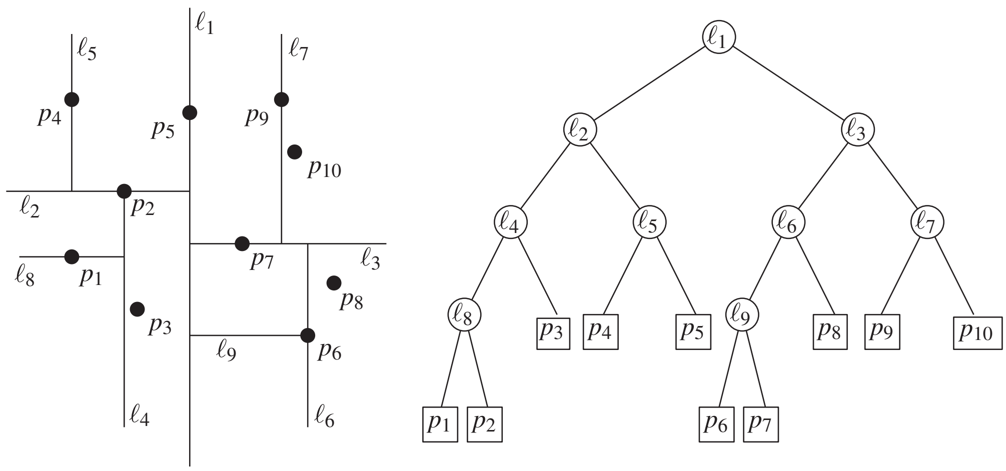


Abb. 2.1: Exemplarische Darstellung eines K-D-TREES für $d = 2$ Dimensionen. In der linkten Hälfte ist die graphische Interpretation der Aufteilung und in der rechten der Aufbau des entstehenden Baumes zu finden [7].

sich wie N verhält. Es hat sich allerdings gezeigt, dass wenn d hinreichend groß ist, diese Vorteile geringer werden, und ab einer gewissen Dimensionalität die Suche langsamer als ein naiver Suchvorgang abläuft.

2.5 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* (im Folgenden als RBF abgekürzt) 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 [6]. Eine mögliche Wahl der Basisfunktionen sind zum Beispiel Gaußfunktionen

$$\phi_i(\vec{x}) = \exp\left(-\frac{||\vec{x} - \vec{z}_i||}{\sigma^2_{RBF,i}}\right).$$

Die Linearkombination führt zu dem Ansatz

$$\vec{y} = F(\vec{x}) = \sum_{i=1}^{l} \vec{\omega}_i \phi(||\vec{x} - \vec{z}_i||), \tag{2.10}$$

wobei $\sigma_{RBF,i}$ für die Breite der i-ten Gaußfunktion steht. 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

$$\mathbf{Y} := \begin{pmatrix} y_{11} & \dots & y_{1m} \\ \vdots & & \vdots \\ y_{N1} & \dots & y_{Nm} \end{pmatrix}. \tag{2.11}$$

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

$$\mathbf{\Omega} := \begin{pmatrix} \omega_{11} & \dots & \omega_{1m} \\ \vdots & & \vdots \\ \omega_{l1} & \dots & \omega_{lm} \end{pmatrix}. \tag{2.12}$$

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

$$\mathbf{A} := \begin{pmatrix} A_{11} & \dots & A_{1m} \\ \vdots & & \vdots \\ A_{N1} & \dots & A_{Nm} \end{pmatrix}, \tag{2.13}$$

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

$$\mathbf{Y} = \mathbf{A} \cdot \mathbf{\Omega} \tag{2.14}$$

ausdrücken [6]. 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 [6]. Mit diesem Ansatz ergibt sich die Lösung zu

$$\mathbf{\Omega} = \mathbf{A}' \cdot \mathbf{Y}. \tag{2.15}$$

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

2.6 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 nichtlinearen Einheiten zusammengesetzt sind. Die erste dieser Ebenen wird zur Eingabe und die letzte zur Ausgabe eines Signals genutzt. Eine Schematische Darstellung ist im linken Teil der Abbildung 2.2 zu finden. 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 [5, S. 225-290].

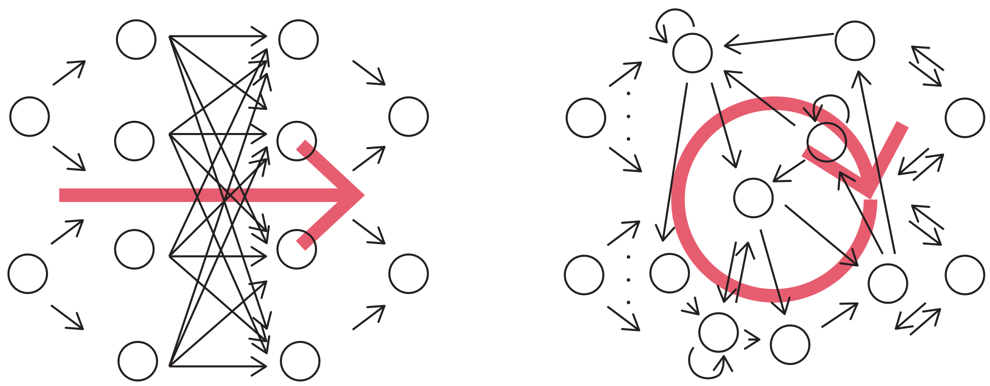


Abb. 2.2: Schematische Darstellung eines FFNN mit vier Ebenen (links) und eines RNN (rechts) mit ihren jeweiligen Verbindungen und der Eingangs- und Ausgangsebene. Der Informationsfluss ist in rot eingetragen (nach [9]).

Ein RNN hat einen ähnlichen Aufbau, doch hier können alle Einheiten an alle anderen Einheiten Signale weitergeben und von diesen erhalten. Die schematische Struktur ist im rechten Teil der Abbildung 2.2 dargestellt. 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 genutzt werden kann, sondern eine für RNNs abgewandelte Form genutzt werden muss. Für den prominenteste Algorithmus 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 und auch das Explodieren des Gradienten der Kostenfunktion deutlich wahrscheinlicher als bei der gewöhnlichen Backpropagation ist [9, 16].

2.7 Echo State Network

2.7.1 Überblick

Um die (Leistungs)Probleme der RNN zu umgehen, wurden als mögliche Lösung die Echo State Networks von H. Jäger vorgeschlagen [8]. 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 [14].

2.7.2 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 [13]. Im Folgenden wird der Aufbau und anschließend die Funktionsweise eines solchen Netzwerkes nach [11] 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 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

$$\vec{s}(n) = (1 - \alpha) \cdot \vec{x}(n-1) + \alpha \cdot f_{in} \left(\mathbf{W}_{in}[b_{in}; \vec{u}(n)] + \mathbf{W}\vec{x}(n-1) \right), \tag{2.16}$$

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

$$\vec{s}(n) = f_{in}\left(\mathbf{W}_{in}[b_{in}; \vec{u}(n)] + \mathbf{W}\vec{x}(n-1)\right). \tag{2.17}$$

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

$$\vec{s}(n) = (1-\alpha) \cdot \vec{x}(n-1)\alpha \cdot f_{in}\left(\mathbf{W}_{in}[b_{in}; \vec{u}(n)] + \mathbf{W}\vec{x}(n-1) + \mathbf{W}_{fb}\vec{y}(n)\right) \tag{2.18}$$

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

$$\vec{y}(n) = f_{out}\left(\mathbf{W}_{out}\vec{x}(n) = \mathbf{W}_{out}[b_{out}; \vec{s}(n); \vec{u}(n)]\right), \tag{2.19}$$

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 (2.18) genutzt wird, wird $\mathbf{W}_{fb}$ gleichartig konstruiert. Auf das Generieren der inneren Matrix $\mathbf{W}$ wird in Abschnitt 2.7.4 genauer eingegangen.

2.7.3 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

$$\mathbf{Y} = \mathbf{W}_{out}\mathbf{X} \tag{2.20}$$

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

$$\mathbf{W}_{out} = \mathbf{Y}\mathbf{X}^T \left(\mathbf{X}\mathbf{X}^T + \beta I\right)^{-1} \tag{2.21}$$

ist sehr leistungsstark, aber auch teilweise numerisch instabil. Bei geeigneter Wahl von β können die besten Ergebnisse hinsichtlich der Genauigkeit der Vorsage erzielt werden [13]. Deshalb wird in dieser Arbeit auch nur dieses Lösungsverfahren verwendet. Die weiteren Lösungsansätze für das Gleichungssystem sind nur aus Gründen der Vollständigkeit angegeben.

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

$$\mathbf{W}_{out} = \mathbf{Y}\mathbf{X}' \tag{2.22}$$

folgt. Dieses Verfahren ist zwar sehr rechenaufwendig aber dafür numerisch stabil [10, 13]. Nichts desto trotz, kann allerdings auf Grund des Fehlens einer Regularisierung leicht der Effekt des Overfittings auftreten. Auf Grund dessen wird es in dieser Arbeit nicht verwendet.

Um den Effekt des Overfittings bei der Verwendung der Psuedoinversen zu reduzieren, kann in der Zustandsgleichung (2.16) beziehungsweise (2.18) eine leichte normalverteilte Störung $\vec{\nu}(n)$ der Größenordnung 1×10^{-1} bis 1×10^{-5} addiert wird. Falls die *Tikhonov Regularisierung* zur Lösung verwendet wird, erhöht die Verwendung der zufälligen Störung die Stabilität der Vorhersage des System. Dieser Ansatz beruht auf Empirie, da eine mathematische Begründung hierfür noch nicht vollständig gelungen ist [8, 13]. Anschaulich lässt sich das Vorgehen dadurch motivieren, dass hierdurch künstliche Datenpunkte in der nähe der vorhandenen Trainingsdaten emuliert werden, und somit eine größere Vielfalt an Daten während der Trainingsphase beobachtet wird.

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 $n > T$

Zusätzlich zu diesen Eigenschaften wird die Dynamik des Reservoirs auch von dessen Größe N bestimmt. Es kann gezeigt werden, dass die Gedächtnisleistung eines Reservoirs stark von dieser abhängt. Somit ist es ratsam für Aufgaben, die eine lange Gedächtnisleistung benötigen, ein großes und für Aufgaben, die nur ein Kurzzeitgedächtnis benötigten, ein kleines Reservoir zu benutzen [9].

2.7.4 Theoretischer Hintergrund

Um die mathematischen Eigenschaften beschreiben zu können, sind zuerst zwei Definitionen nötig [18].

Definition 1 (Kompatibler Zustand)**.** *Sei $S : X \times U \to X$ ein* ESN *mit der Zustandsgleichung $\vec{x}_{n+1} = F\left(\vec{x}_n, \vec{u}_{n+1}\right)$. Eine Folge von Zuständen $(\vec{x}_n)_n$ ist kompatibel mit der Eingangsfolge $(\vec{u}_n)_n$, wenn $\vec{x}_{n+1} = F\left(\vec{x}_n, \vec{u}_{n+1}\right), \forall n \leq 0$ 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 [9].

Nun stellt sich die Frage, wann ein Netzwerk diese Eigenschaft besitzt. Es wird schnell klar, dass dies hauptsächlich 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 [8]. Hieraus ergibt sich, dass die *ESP* vorliegt, wenn

$$|1 - \alpha(1 - \sigma_{max}(\mathbf{W}))| < 1 \tag{2.23}$$

erfüllt ist, wobei $\sigma_{max}(\mathbf{W})$ der größte Singulärwert ist [11].
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 [8, 11].

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 [18]. Stattdessen konnte gezeigt werden, dass eine hinreichende Bedingung durch

$$\rho(\alpha|\mathbf{W}| + (1 - \alpha)\mathbf{I}) < 1 \tag{2.24}$$

gegeben ist - wobei als Betrag der Matrix hier das elementweise Betragsnehmen gemeint ist. Diese Bedingung ist weniger einschränkend als Gleichung (2.23) [18].

Weitergehend hat sich in Experimenten gezeigt, dass eine dünnbesetze Gewichtsmatrix $\mathbf{W}$ zu reicheren Dynamiken innerhalb des Reservoirs führen kann [8]. Eine solche dünnbesetze Matrix bedeutet, dass nicht mehr jedes Neuron mit jedem anderen Neuron verbunden ist, sondern dass nur noch ein relativer Anteil ϵ dieser Verbindungen vorhanden ist. Da durch eine größere Anzahl an verschiedenen internen Dynamiken vielfältigere Funktionen besser approximiert werden können, kann die Vorhersagequalität durch einen Geringen ϵ Wert erhöht werden.

Darauf basierend kann nun eine Methode nach [18] angegeben werden, um die Gewichtsmatrix $\mathbf{W}$ zu konstruieren:

1. Generiere zufällige Matrix $\mathbf{W}$ mit $|\mathbf{W}| = \mathbf{W}$ bei der in jeder Zeile nur ϵ Einträge ungleich 0 sind.
2. Skaliere $\mathbf{W}$, sodass Gleichung (2.24) 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

Die zuvor in Kapitel 2 eingeführten Methoden werden nun durch drei verschiedene Szenarien ausprobiert und verglichen. Hierbei liegt der Fokus auf der Verwendung und Erprobung der ESNs. Da die klassischen Methoden der *nächsten Nachbarn* (NN) und der *radialen Basisfunktionen* (RBF) bereits seit längerer Zeit bekannt sind und populäre Lösung solcher Problemfälle darstellen, dienen sie als Bezugsgröße.

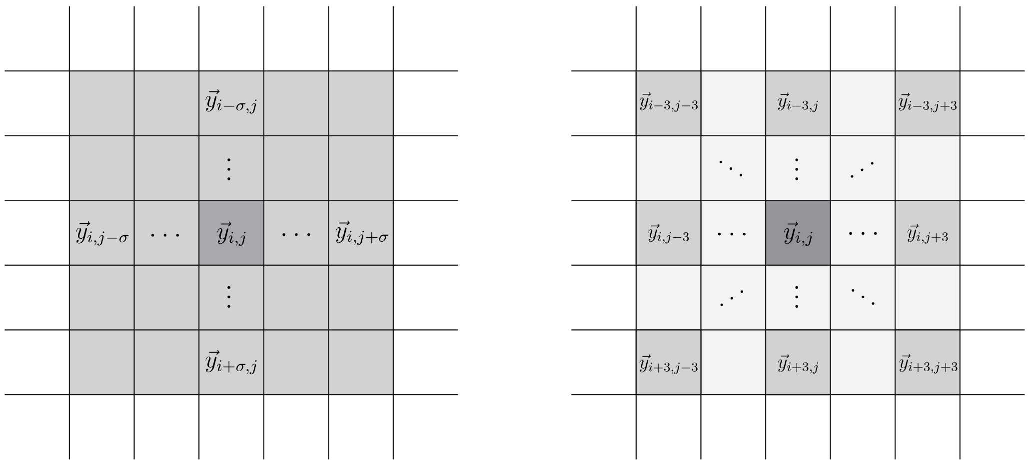
Jedes der drei Szenarien wird sowohl auf ein *Barkley*-System als auch auf ein System nach dem *Mitchell-Schaeffer*-Modell angewendet. Diese Systeme bestehen aus 150 Gitterpunkten und nutzen die zuvor beschriebenen Parameter. Für ihre Startverteilung werden die Felder der beiden Systemvariablen in 100 Quadrate unterteilt, und diese mit Zufallswerten zwischen 0 und 1 initialisiert. Anschließend werden die Systeme über 2000 Zeitschritte ($\widehat{=}$400.0 Zeiteinheiten) simuliert um ein transientes Verhalten abzuwarten. Durch das weitere Simulieren der Systeme werden die Test und Trainingsdaten ermittelt. Dabei wird für das *Barkley* eine Samplingzeit von 0.1 und für das *Mitchell-Schaeffer*-Modell von 1.0 Zeiteinheiten benutzt.

Die erste Aufgabe besteht darin aus der Kenntnis einer der beiden Systemvariablen die andere Unbekannte zu ermitteln. Dabei wird die Spannungsvariable als Quelle genutzt. Dies ist in den zuvor eingeführten Modellen jeweils die Größe, welche den Diffusionsterm beinhaltet; also die u-Variable im *Barkley*-Modell und die v-Variable im *Mitchell-Schaeffer*-Modell.
Im zweiten Szenario werden die Techniken verwendet um aus Messdaten einer simulierten Fernfeldmessung der Spannungsvariable diese wiederherzustellen. Diese Fernfeldmessung wird durch eine gaußsche Unschärfe simuliert.
Abschließend wird die Spannungsvariable der inneren Punkte eines Quadrates nur durch die Kenntnis der Randwerte des Systems vorhergesagt.

Add subchapter?

3.1 Allgemeines Vorgehen

Das Ziel aller drei Aufgaben besteht jeweils darin ein zweidimensionales Feld vorherzusagen. Eine naheliegende Möglichkeit dies zu schaffen besteht darin wirklich den gesamten Inhalt des 150×150 Einheiten großen Feldes auf einmal vorherzusagen. Da dabei die Ausgabe der Vorhersage aus einem 22500-dimensionalen Vektor besteht werden sehr viele Trainingsdaten benötigt, um genügend Informationen über eine solch hochdimensionale Ausgabe zu erhalten. Um dieses Problem zu umgehen wird stattdessen ein Verfahren benutzt, bei dem jeder Punkt einzeln vorhergesagt wird. Dies hat zudem den Vorteil, dass aus einer monströsen Vorhersage, welche mitunter viel Arbeitsspeicher verbrauchen würde, in viele kleine Vorhersagen aufteilt. Hierdurch sinkt der zur Berechnung benötigte Bedarf an Arbeitsspeicher drastisch.



(a) Messsonde ohne Abstände zwischen den Messpunkten

(b) Messonde mit einem Abstand von zwei Einheiten zwischen den Messpunkten

Abb. 3.1: Illustration der verwendeten *Messsondentechnik*. Abbildung 3.1a deutet an, wie aus einem σ^2 großem Quadrat um den eigentlichen Messpunkt Daten für die Vorhersage genutzt werden. Dagegen ist in Abbildung 3.1b das Verfahren für $\sigma = 5$ und $\Delta\sigma = 2$ dargestellt, sodass insgesamt die Information aus 9 Punkten genutzt wird.

Des Weiteren kann angenommen werden, dass die Dynamiken einen ausgeprägten lokalen Charakter besitzen, sodass zumindest bei den ersten beiden Aufgaben weit entfernte Punkte keinen unmittelbaren Einfluss auf die Vorhersage haben. Darauf

basierend kann eine sogenannte *Messsondentechnik* entwickelt und für diese genutzt werden. Hierbei werden nicht nur die Informationen an einem Punkt (i, j) für die Vorhersage, sondern auch die benachbarten Punkte, welche in einem Quadrat um (i, j) liegen, genutzt. Eine Veranschaulichung ist in 3.1a zu finden. Die Größe des Quadrates wird durch den Parameter σ bestimmt, und ergibt sich zu σ^2. Da direkt Nachbarn unter Umständen durch den geringen Abstand sehr ähnliche Informationen beinhalten können, wird zudem ein Parameter $\Delta\sigma$ eingeführt, welche den Abstand zweier benachbarter Punkte, deren Information simultan verwendet werden, angibt. Eine beispielhafte Darstellung hiervon ist für $\sigma = 5, \Delta\sigma = 2$ in Abbildung 3.1b dargestellt. Dabei werden nur die Zeitreihen der Gitterpunkte genutzt, welche dunkelgrau hinterlegt sind, und die hellgrauen Informationen verworfen. Die Parameter, welche für die ersten beiden Aufgaben überprüft werden, sind in Tabelle 3.1 aufgelistet. Dabei ist anzumerken, dass die Diskretisierung des Diffusionstermes in den Differentialgleichungen einem Wert $\sigma = 3$ entsprechen würde.
Durch dieses Vorgehen kann für jeden Gitterpunkt ein $\left\lceil \frac{\sigma}{\Delta\sigma} \right\rceil^2$-dimensionaler Eingabevektor erstellt und für die ersten beiden Vorhersage-Aufgaben genutzt werden.

σ	1	3	5		7		
$\Delta\sigma$	1	1	1	2	1	2	3

Tab. 3.1: In den ersten beiden Aufgaben verwendete Parameter σ und $\Delta\sigma$ für die *Messsondentechnik*.

Der Trainingsvorgang wird jeweils über $N_{Training} = 15000$ Zeitschritte durchgeführt und der Anschließende Evaluationsdurchgang auf $N_{Testing} = 2000$ Zeitschritte. Zur Bewertung der Leistung einer Vorhersage werden die beiden Fehlergrößen MSE und NRMSE eingeführt. Im Allgemeinen ist der MSE (*Mean Squared Error*) durch

$$MSE(y) = \sum_{i}^{m} \sum_{t}^{N_{Testing}} (y(t)_i - \hat{y}(t)_i)^2 \tag{3.1}$$

definiert und charakterisiert die Genauigkeit einer Vorhersage $\hat{y}$ im Vergleich zu dem tatsächlichen Wert $y \in \mathbb{R}^m$ über den Zeitraum $N_{Testing}$. Der NRMSE normiert diesen Fehler noch auf eine Vorhersage, bei der der Mittelwert $\langle y \rangle$ über die Trainingsphase als vorhergesagten Wert genutzt wird. Er ist als

$$NRMSE(y) = \sqrt{\frac{MSE(y)}{MSE(\langle y \rangle)}} \tag{3.2}$$

definiert. Zusätzlich zu diesen Fehlermaßen werden im Folgenden oftmals auch die Laufzeiten der Ansätze angegeben. Hierbei ist zu beachten, dass diese nicht über mehrere Ausführungen des identischen Programmes gemittelt worden sind, und deshalb nicht als statistisch relevante Information sondern nur als ein Hinweis gesehen werden können.

Unter der Kenntnis, dass in den Modellen nur Werte zwischen 0 und 1 angenommen werden dürfen beziehungsweise angenommen werden, werden die Vorhersagen auf das Intervall $[0, 1]$ beschränkt. Dafür werden die Werte beider Variablen der Systeme sowohl nach unten als auch nach oben hin nach

$$x = \begin{cases} 0, & \text{wenn } x \leq 0 \\ x, & \text{wenn } x \geq 0 \wedge \leq 1 \\ 1, & \text{wenn } x \geq 1 \end{cases} \tag{3.3}$$

abgeschnitten, wobei x für eine der beiden Variablen in dem jeweiligen Modell steht.

3.1.1 Echo State Network

Echo State Networks besitzen viele verschiedene Hyperparameter, welche die Qualität der Vorhersage beeinflussen können. Dazu zählen nach 2.7 die Reservoirgröße N, der Spektralradius ρ, die Verlustrate α, die Amplitude der zufälligen Störung ν, die Stärke der Regularisierung λ und der Anteil der vorhandenen internen Verbindungen ϵ. Da es zum aktuellen Zeitpunkt noch keinen zufriendenstellenden mathematischen Algorithmus für das das selbstständige optimale Einstellen eines ESNs gibt, müssen die Parameter manuell ermittelt werden. Hierfür wird in dieser Arbeit eine GridSearch benutzt. Bei diesem Verfahren wird der Hyperparameterraum in festgelegten Schritten abgetastet und die Leistung des somit entstehenden Netzwerke evaluiert und somit die besten Parameter ermittelt. Durch die hohe Anzahl der einstellbaren Hyperparameter und die nicht zu vernachlässigende Rechenzeit für das Trainieren und Evaluieren eines Netzwerkes, ist es nicht sinnvoll diese Suche

für alle Komponenten des hochdimensionalen Zielvektors gleichzeitig durchzuführen. Stattdessen wird zuerst unter der Annahme, dass die Dynamik sich lokal an allen Punkten ähnlich verhält, ein Punkt in der Mitte des Feldes ausgewählt, und nur versucht diesen einen einzelnen Punkt vorherzusagen. Diese Aufgabe kann deutlich schneller berechnet werden, sodass nun die optimalen Hyperparameter mit einer GridSearch gesucht werden können. Im Anschluss können die Hyperparameter des zuvor ermittelten ESN für die Vorhersage aller Punkte genutzt werden. Abschließend wird noch einmal Versucht das gefundene Reservoir manuell zu verbessern, indem die Parameter N und λ noch einmal variiert werden.
Es ist zu erwarten, dass die hierbei gefundenen Hyperparameter eine akzeptable Leistung für die jeweiligen Probleme erzielen können. Da allerdings bei dem zuvor beschriebenen Verfahren bei weitem nicht alle sinnvollen Hyperparameter getestet werden können, besteht die Möglichkeit, dass es noch besser geeignete Reservoirs mit anderen Hyperparametern gibt, welche eine noch höhere Leistung erzielen können.

3.1.2 Klassische Methoden

Die klassischen Methoden sind nicht von alleine aus in der Lage zeitlich ausgeprägte Dynamiken vorherzusagen, da den Methoden a priori keine Informationen über die vorherigen Zustände vorliegen. Um dieses Problem zu lösen können Verzögerungs-Koordinaten mittels der in Abschnitt 2.3 beschriebenen *Delay Reconstruction* für die in Abschnitt 3.1 beschriebenen Vektoren aufgestellt werden. Die über die Autokorrelation ermittelte zeitliche Verzögerung τ ist für beide Systeme in Tabelle 3.2 dargestellt.

$\tau_{Barkley}$	$\tau_{Mitchell-Schaeffer}$
0.64 Zeiteinheiten	2.38 Zeiteinheiten

Tab. 3.2: Verwendete zeitliche Verzögerung τ für die *Delay Reconstruction* für das *Mitchell-Schaeffer*- und das *Barkley*-Modell

3.2 Kreuz-Prädiktion

Momentan ist es durch invitro Experimente bereits möglich die Ausbreitung der elektrischen Erregung auf der Oberfläche des Herzmuskels experimentell aufzuzeichnen. Nun stellt sich die Frage, ob anhand beispielsweise der Messung der Membramspannung weitere Variablen des Systems wie die Kalium-Konzentration oder ähnliches ermittelt werden kann. Diese Fragestellung wird in der ersten Aufgabe betrachtet. Es wird die Vorhersage von der Spannungsvariable auf die zweite Variable des jeweiligen Modells sowohl für das *Barkley*- als auch für das *Mitchell-Schaeffer*-Modell durchgeführt. Dabei wird zuerst die Nächste Nachbar Methode, anschließend die radialen Basisfunktionen und schlussendlich die *ESN*s verwendet. Es werden sowohl die einzelnen Ergebnisse präsentiert als auch ein abschließender Vergleich durchgeführt.

3.2.1 Nächste Nachbar Vorhersage

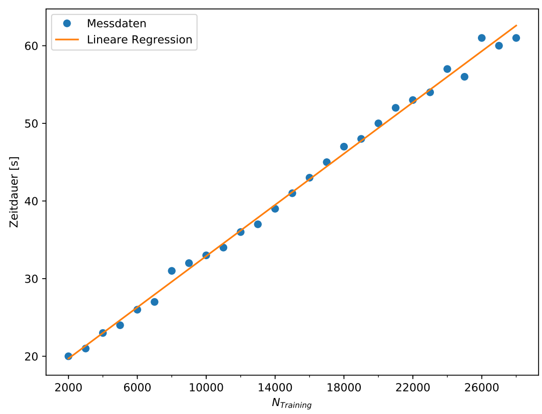
Die Ergebnisse für die optimalen Hyperparameter sind in Tabelle 3.3 zu finden. Dabei sind sowohl die verwendeten Parameter als auch die erzielten Fehler MSE und NRMSE aufgelistet.

	Barkley	Mitchell-Schaeffer
σ	1	7
$\Delta\sigma$	1	1
δ	3	3
k	5	5
Laufzeit [s]	40	5252
MSE	**0.00105**	**0.01353**
NRMSE	**0.1367**	**0.7438**

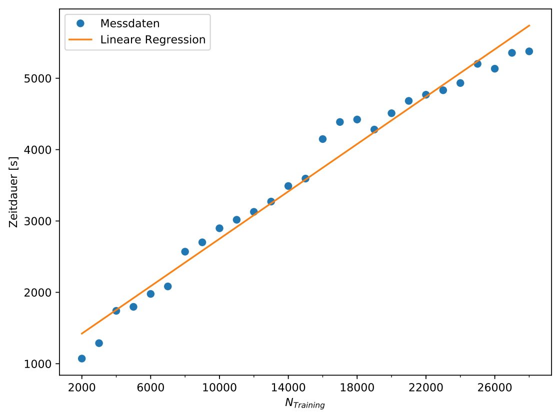
Tab. 3.3: Gefundene Hyperparameter der nächsten Nachbar Vorhersage für das *Mitchell-Schaeffer*- und das *Barkley*-Modell, welche zu den geringsten Fehlern führen.

Dabei ist die stark unterschiedliche Laufzeit der beiden Vorhersagen auffällig. Dies lässt sich allerdings durch die verschiedenen Dimensionalitäten der Quellvariable erklären: Während beim *Barkley*-Modell lediglich ein 3-dimensionaler Vektor für die Vorhersage die besten Ergebnisse erzielt konnte beim *Mitchell-Schaeffer*-Modell durch die Verwendung eines 147-dimensionalen Quellvektors die besten Ergebnisse erzielt werden. Da, wie in Abschnitt 2.4 erwähnt, die benötigte Zeit für eine Vorhersage sehr stark mit der Dimension zunimmt, lässt sich somit der Anstieg von 40 auf 5252 Sekunden erklären.

Da eine Nächsten Nachbar Vorhersage nur anhand der in der Trainingsphase gesehenen Datenpunkte eine Vorhersage erstellt, ist anzunehmen, dass die Qualität dieser sehr stark von der Länge der Trainingsphase abhängt. Um dies zu untersuchen ist für die zuvor ermittelten Hyperparameter eine Vorhersage für verschiedene Trainingslängen $N_{Training}$ durchgeführt und die dabei auftretenden MSEs und die benötigte Laufzeit gemessen worden. Hierbei können zwei Effekte beobachtet werden. Bei der Betrachtung der grafischen Darstellung der benötigten Laufzeit in Abbildung 3.2 ist zu erkennen, dass ein linearer Zusammenhang zwischen der $N_{Training}$ und Laufzeit existiert. Der erzielte Fehler verhält sich dagegen anders und sinkt asymptotisch gegen eine untere Schranke ab nach Abbildung 3.3. Anzumerken ist, dass die Sättigung des Fehlers im *Barkley*-Modell schon ab etwa $N_{Training} = 15000$ eintritt, doch beim *Mitchell-Schaeffer*-Modell erst deutlich später. Dies ist ein Hinweis darauf, dass die Dynamiken im letzteren chaotischer und unregelmäßiger als bei ersten ablaufen. Zusammenfassend lässt sich somit die Wahl der Trainingslänge von $N_{Training} = 15000$ für alle Szenarien und alle drei Methoden damit begründen, dass man für die Nächste Nachbar Vorhersage, welche am empfindlichsten auf diese Länge reagiert, eine akzeptablen Kompromiss zwischen der Rechenzeit und der Genauigkeit erhält.

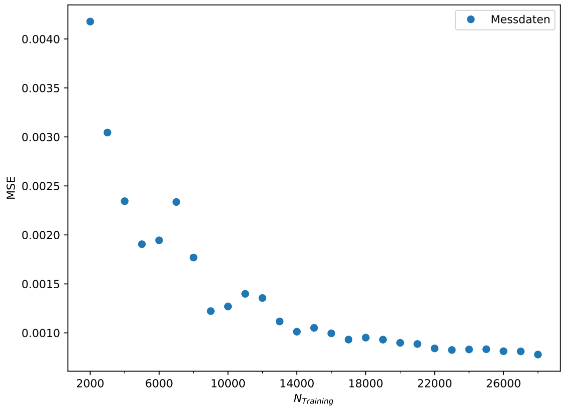


(a) Abhängigkeit für das *Barkey*-Model.

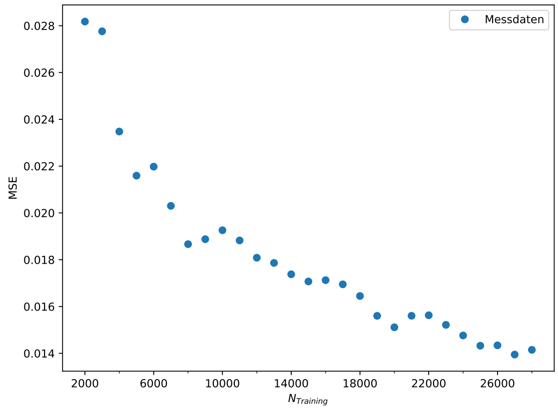


(b) Abhängigkeit für das *Mitchell-Schaeffer*-Model.

Abb. 3.2: Darstellung der Abhängigkeit des benötigten Laufzeit von der verwendeten Anzahl an Trainingsdaten $N_{Training}$ für das *Barkley*-Modell (links) und für das *Mitchell-Schaeffer*-Modell (rechts) bei der Verwendung einer nächsten Nachbar Vorhersage.



(a) Abhängigkeit für das *Barkey*-Model.



(b) Abhängigkeit für das *Mitchell-Schaeffer*-Model.

Abb. 3.3: Darstellung der Abhängigkeit des MSE von der verwendeten Anzahl an Trainingsdaten $N_{Training}$ für das *Barkley*-Modell (links) und für das *Mitchell-Schaeffer*-Modell (rechts) bei der Verwendung einer nächsten Nachbar Vorhersage.

3.2.2 Radiale Basisfunktionen

Bei der Verwendung radialer Basisfunktionen stellt zudem die Breite σ_{RBF} der Gaußfunktionen als auch die Anzahl der Basisfunktionen l einen wichtigen Parameter dar. Im Rahmen dieser Arbeit ist die Anzahl der Basisfunktionen auf $l = 100$ festgelegt worden - diese Wahl wird im Folgenden weiter motiviert werden. Um die anderen Parameter zu finden, sind σ, $\Delta\sigma$ wie oben beschrieben, $\delta \in [3, 4, 5]$ und $\sigma_{RBF} \in [0.5, 1.0, 3.0, 5.0, 7.0, 9.0]$ variiert worden. In Tabelle 3.4 sind die dadurch gefundenen optimalen Parameter, die damit erreichten Fehler und die benötigte Laufzeit erneut für beide Modelle aufgelistet. Hierbei ist zu bemerken, dass die optimalen Werte für σ, $\Delta\sigma$ und δ mit denen für die NN-Vorhersage übereinstimmen.

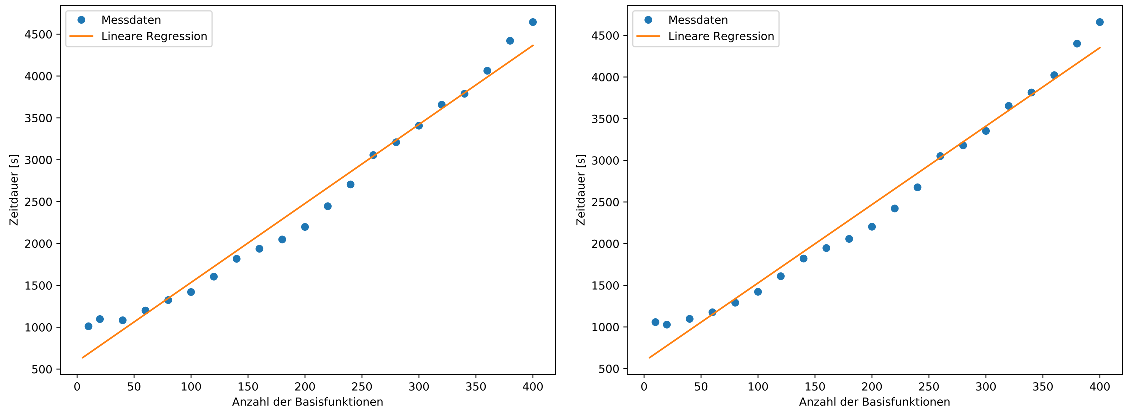
Analog zu der Untersuchung des Einflusses der Trainingslänge $N_{Training}$ bietet es sich für die radialen Basisfunktionen an, den Einfluss der Anzahl der verwendeten Basisfunktionen l auf die Genauigkeit und die benötigte Laufzeit zu untersuchen. Dabei werden jeweils wieder die besten zuvor ermittelten Hyperparameter verwendet. Hierfür sind die gemessenen Laufzeiten gegen die Anzahl der Basisfunktionen in Abbildung 3.4 aufgetragen worden. Es ist erneut anzunehmen, dass ein linearer Zusammenhang zwischen den beiden Größen existiert.

Der Zusammenhang zwischen dem MSE und der Anzahl der Basisfunktionen ist in Abbildung 3.5 zusehen. Zum einen kann ein ein asymptotischer Anteil erkannt

	Barkley	Mitchell-Schaeffer
σ	1	7
$\Delta\sigma$	1	1
δ	3	3
σ_{RBF}	0.5	5
Laufzeit [s]	1430	1434
MSE	**0.00064**	**0.00890**
NRMSE	**0.1069**	**0.6034**

Tab. 3.4: Gefundene Hyperparameter der radialen Basisfunktionen für das *Mitchell-Schaeffer*- und das *Barkley*-Modell, welche zu den geringsten Fehlern führen.

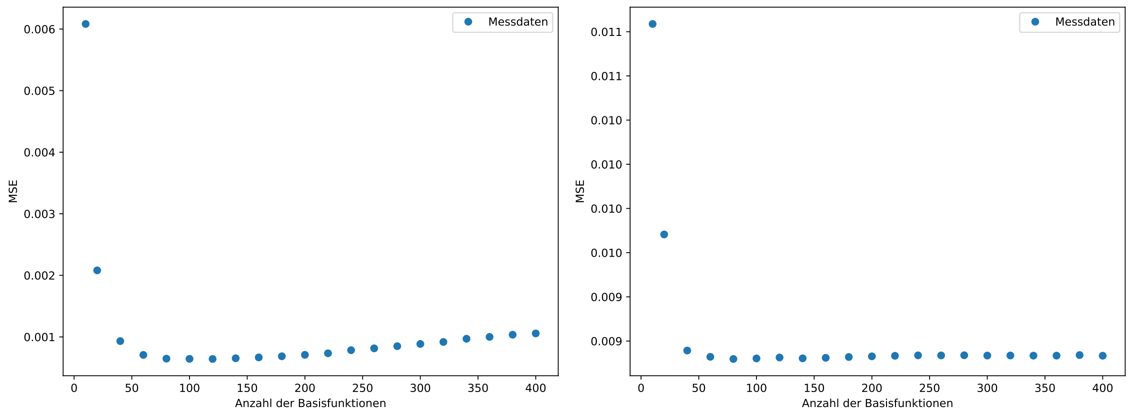
werden, sodass der Fehler erst einmal für mehr Basisfunktionen abnimmt. Allerdings lässt Abbildung 3.5a erahnen, dass es hierbei einen optimalen Wert gibt, ab dem der Fehler wieder ansteigt. Dies kann durch eine schlechtere Generalisierung der Dynamik und ein zu starkes Anpassen und die Trainingsphase (auch bekannt als *Overfitting*) erklärt werden. Zusammenfassend zeigt sich, dass die Wahl von 100 Basisfunktionen eine akzeptable Abschätzung ist, sodass der Fehler möglichst gering ist und die Laufzeit auch gering gehalten wird. Diese Annahme wird im Folgenden ohne weitere qualitative Untersuchungen auf die anderen beiden Probleme übertragen, um den benötigten Rechenaufwand für die Parametersuche in einem angebrachten Rahmen zu halten.



(a) Abhängigkeit für das *Barkey*-Model.

(b) Abhängigkeit für das *Mitchell-Schaeffer*-Model.

Abb. 3.4: Darstellung der Abhängigkeit des benötigten Laufzeit der Basisfunktionen l für das *Barkley*-Modell (links) und für das *Mitchell-Schaeffer*-Modell (rechts) bei der Verwendung radialer Basisfunktionen.



(a) Abhängigkeit für das *Barkey*-Model.

(b) Abhängigkeit für das *Mitchell-Schaeffer*-Model.

Abb. 3.5: Darstellung der Abhängigkeit des MSE von der verwendeten Anzahl der Basisfunktionen l für das *Barkley*-Modell (links) und für das *Mitchell-Schaeffer*-Modell (rechts) bei der Verwendung radialer Basisfunktionen.

3.2.3 Echo State Network

Abschließend ist dieses Problem nun mit den *ESN*s gelöst worden. Dazu wurden die Hyperparameter nach Abschnitt 3.1.1 gesucht worden. Die gefundenen Parameter und die damit erreichten Ergebnisse sind in Tabelle 3.5 aufgelistet. Es ist auffällig, dass die optimalen Werte für σ und $\Delta\sigma$ hier von denen der NN- und der RBF-Vorhersage abweichen.

	Barkley	Mitchell-Schaeffer
σ	3	3
$\Delta\sigma$	1	1
N	400	400
$\rho(\lvert\mathbf{W}\rvert)$	0.95	0.95
α	0.05	0.05
ϵ	0.1	0.1
ν_{max}	1×10^{-4}	1×10^{-4}
λ	5×10^{-6}	5×10^{-6}
Laufzeit [s]	3710	3733
MSE	1.95×10^{-7}	**0.00075**
NRMSE	3.47×10^{-6}	**0.1753**

Tab. 3.5: Gefundene Hyperparameter des ESN für das *Mitchell-Schaeffer*- und das *Barkley*-Modell, welche zu den geringsten Fehlern führen.

Auffällig ist, dass für beide Modelle die gleichen Hyperparameter die höchste Genauigkeit erzielen.

Add more details?

3.2.4 Vergleich

Abschließend kann nun ein Vergleich der drei verwendeten Methoden hinsichtlich ihrer Laufzeit und der erzielten Genauigkeiten durchgeführt werden. Dieser ist in Tabelle 3.6 zu finden. Die jeweils besten Ergebnisse sind hervorgehoben. Die *ESN*s erzielen für beide Modelle den geringsten Fehler, also die höchste Genauigkeit. Dabei ist der NRMSE für das *Barkley*-Modell mehrere Größenordnung kleiner als bei den Konkurrenz-Ansätzen. Diese überaus hohe Genauigkeit ist für das *Mitchell-Schaeffer*-Modell nicht erreicht worden. Hier beträgt der Fehler trotzdem etwa nur ein Drittel von dem der anderen Ansätze. Im Austausch für diese hohe Genauigkeit ist allerdings die benötigte Zeit für die Vorhersage höher als bei den Konkurrenten.

Unter der Voraussetzung, dass die Rechenzeit nur eine untergeordnete Rolle spielt, so ergeben sich die *ESN*s als bester Ansätze für die Kreuz-Prädiktion.

	Barkley			Mitchell-Schaeffer		
	NN	RBF	ESN	NN	RBF	ESN
Laufzeit [s]	**40**	1430	3710	5252	**1434**	3733
MSE	0.00105	0.00064	1.95×10^{-7}	0.01353	0.00890	**0.00075**
NRMSE	0.1367	0.1069	3.47×10^{-6}	0.7438	0.6034	**0.1753**

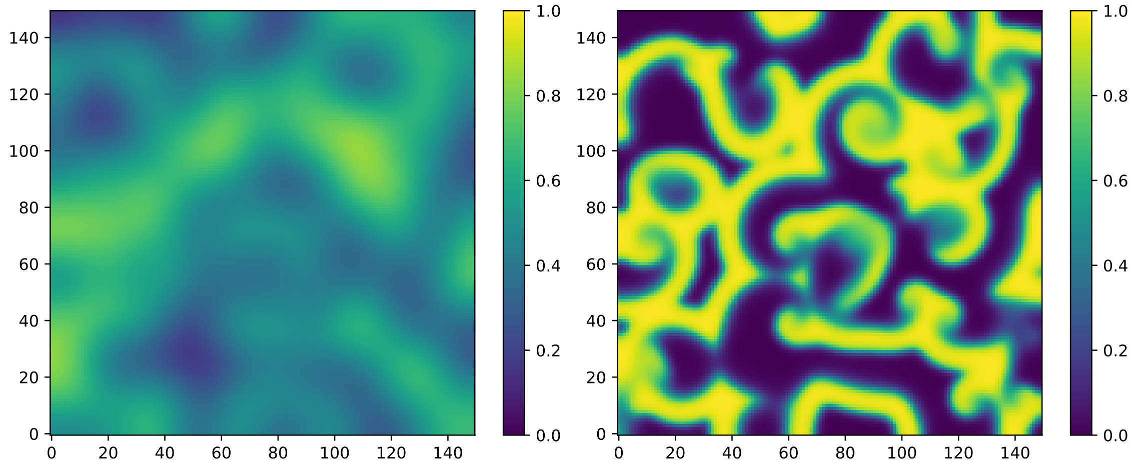
Tab. 3.6: Vergleich der benötigten Laufzeit und der erreichten Fehlers der drei Ansätze für das *Mitchell-Schaeffer*- und das *Barkley*-Modell, welche zu den geringsten Fehlern führen.

3.3 Prädiktion der Dynamik durch das Fernfeld

Bei der Durchführung von invitro Experimenten mit Herzen gibt es verschiedene Möglichkeiten die Messung der elektrischen Erregung auf der Herzoberfläche durchzuführen. Zum einen können Elektroden zur Messung benutzt werden, zum anderen allerdings auch Fluoreszenzmessungen durchgeführt werden. Bei der Verwendung von Elektroden wird effektiv nicht das unmittelbare elektrische Feld auf der Herzoberfläche gemessen, sondern ein Fernfeld dessen. Es stellt sich nun die Frage, ob aus der Kenntnis dieses Fernfeldes die korrekte Erregung auf der Oberfläche bestimmt werden kann. Eine experimentelle Untersuchung dieser Fragestellung wird im Folgenden durchgeführt. Hierfür müssen zuerst diese Fernfeldaufnahmen für das *Barkley*- und für das *Mitchell-Schaeffer*-Modell erzeugt werden. Dabei wird das Fernfeld nicht korrekt simuliert, sondern durch eine gaußsche Unschärfe emuliert. Dazu wird auf das gesamte Feld der Spannungsvariable beider Modelle eine solche Unschärfe mit einer Breite $\sigma_{Blur} = 8.0$ mittels einer Faltung angewendet. Eine exemplarische Darstellung des emulierten Fernfeldes und des tatsächlichen Feldes ist in Abbildungen 3.6 und 3.7 zu finden.

3.3.1 Nächste Nachbar Vorhersage

Zuerst wird diese Aufgabe erneut mit dem NN-Ansatz betrachtet. Die besten gefundenen Hyperparameter dafür sind in Tabelle 3.7 aufgelistet. Bemerkenswert ist erneut die geringe Laufzeit dieses Ansatzes. Dies wird durch die verhältnismäßig



(a) Emuliertes Fernfeld

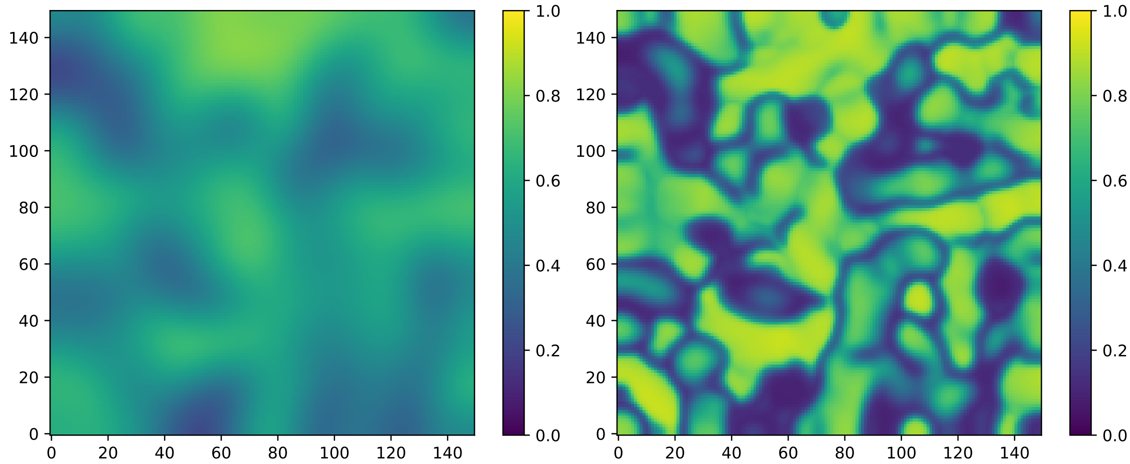
(b) Echte Erregung des Modells

Abb. 3.6: Graphische Darstellung der u-Variable des *Barkley*-Modells. Links ist das emulierte Fernfeld und rechts das tatsächliche u-Feld des Modells zu sehen.

geringe Dimensionalität des Eingabe-Vektors begünstigt. Allerdings sind die Fehlerwerte sehr hoch, sodass die Vorhersage kaum besser ist, als eine Schätzung mit dem Mittelwert als Vorhersage.

	Barkley	Mitchell-Schaeffer
σ	1	1
$\Delta\sigma$	1	1
δ	4	3
k	5	5
Laufzeit [s]	53	42
MSE	**0.10089**	**0.06217**
NRMSE	**0.8227**	**0.9136**

Tab. 3.7: Gefundene Hyperparameter der nächsten Nachbar Vorhersage für das *Mitchell-Schaeffer*- und das *Barkley*-Modell, welche zu den geringsten Fehlern führen.



(a) Emuliertes Fernfeld (b) Echte Erregung des Modells

Abb. 3.7: Graphische Darstellung der v-Variable des *Mitchell-Schaeffer*-Modells. Links ist das emulierte Fernfeld und rechts das tatsächliche v-Feld des Modells zu sehen.

3.3.2 Radiale Basisfunktionen

Als nächstes sind nun die radialen Basisfunktionen ebenfalls auf das Problem angewendet worden. Die dabei gefundenen Hyperparameter sind in Tabelle 3.8 präsentiert.

	Barkley	Mitchell-Schaeffer
σ	3	5
$\Delta\sigma$	1	2
δ	3	3
σ_{RBF}	5.0	9.0
Laufzeit [s]	1840	1842
MSE	**0.03899**	**0.03252**
NRMSE	**0.5114**	**0.6913**

Tab. 3.8: Gefundene Hyperparameter der radialen Basisfunktionen für das *Mitchell-Schaeffer*- und das *Barkley*-Modell, welche zu den geringsten Fehlern führen.

3.3.3 Echo State Network

Nachdem die klassischen Methoden bereits auf dieses Problem angewendet worden sind, kann das Problem nun mittels der ESNs erneut betrachtet werden. Hierfür sind die verwendeten Hyperparameter erneut nach Abschnitt 3.1.1 gesucht worden. Die Ergebnisse sind in Tabelle 3.9 zu finden. Auffällig ist hierbei, dass die optimale Größe N des Reservoirs für beiden Modelle unter der maximal betrachteten Größe $N \leq 400$ liegt. Dies kann ein Anzeichen dafür sein, dass für das Bewältigen der Aufgabe kein ausgeprägtes Langzeitgedächtnis vorhanden sein muss, da diese nach Abschnitt 2.7 mit der Größe N des Reservoirs skaliert.

Add more details on:long time memory vs N dependency in theory.

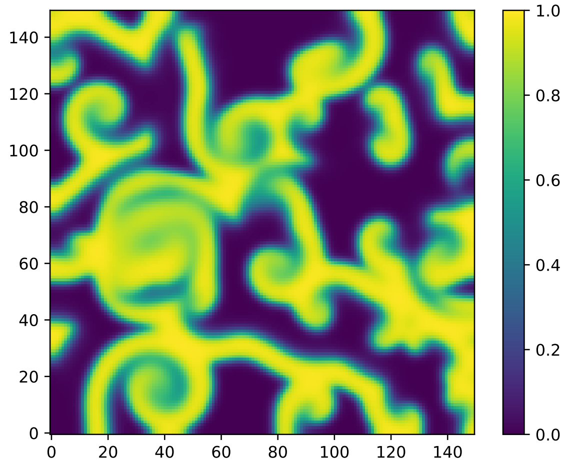
	Barkley	Mitchell-Schaeffer
σ	7	7
$\Delta\sigma$	1	1
N	200	50
$\rho(\lvert\mathbf{W}\rvert)$	1.50	0.10
α	0.20	0.05
ϵ	0.1	0.1
ν_{max}	1×10^{-5}	1×10^{-4}
λ	5×10^{-10}	5×10^{-6}
Laufzeit [s]	1603	1540
MSE	**0.02347**	**0.02449**
NRMSE	**0.3968**	**0.3599**

Tab. 3.9: Gefundene Hyperparameter des ESN für das *Mitchell-Schaeffer*- und das *Barkley*-Modell, welche zu den geringsten Fehlern führen.

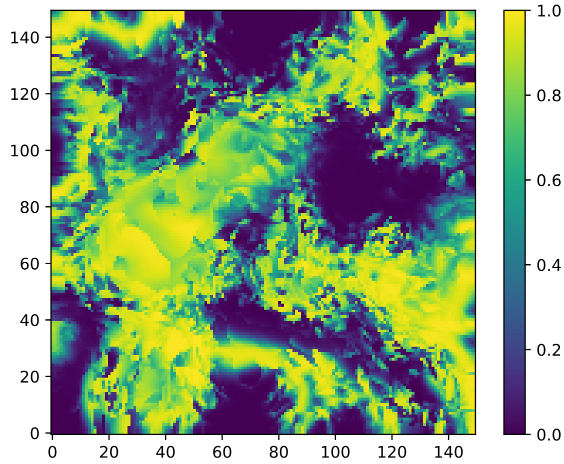
3.3.4 Vergleich

Zusammenfassend können nun die Ergebnisse der drei Ansätze erneut verglichen werden. Eine vergleichende Übersicht ist in Tabelle 3.9 zu finden. Dort ist erneut zu bemerken, dass die ESNs die geringsten Fehlerwerte erzeugt, doch der NN-Ansatz deutlich schneller berechnet werden kann.

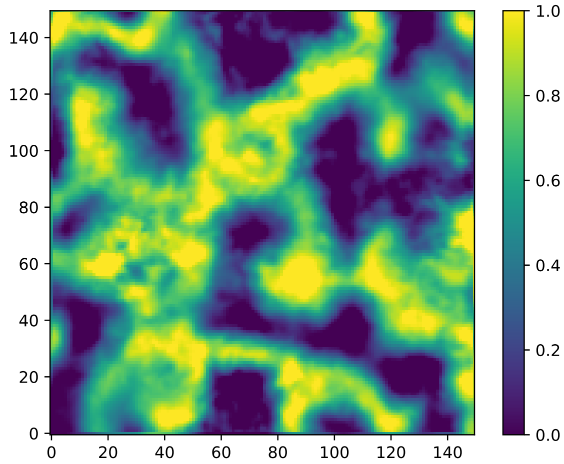
Zusätzlich zu der Tabelle ist noch ein exemplarischer grafischer Vergleich der Resultate der drei Ansätze mit dem Ziel in Abbildung 3.8 dargestellt. Dort fällt auf, dass die Vorhersage des NN-Ansatzes selbst die Struktur der Dynamik kaum korrekt auflöst. Im Vergleich dazu ist die Vorhersage des RBF-Ansatzes und des ESN deutlich feiner und beinhaltet sogar die Makrostruktur der Dynamik. Des Weiteren ist zu bemerken, dass diese mit dem ESN leicht feiner aufgelöst worden ist, als mit RBF-Ansatz. Zwar stimmen hier auch nicht die feinen Details der Dynamik mit dem Original überein, doch ist eine starke Verbesserung im Vergleich zu dem emulierten Fernfeld zu bemerken. Unter Umständen wäre es für zukünftige Arbeiten bei dieser Aufgabe angebracht eine andere Fehlermetrik als die mittlere quadratische Abweichung zu benutzen, welche die Ähnlichkeit zwischen den Strukturen der Felder stärker berücksichtigt.



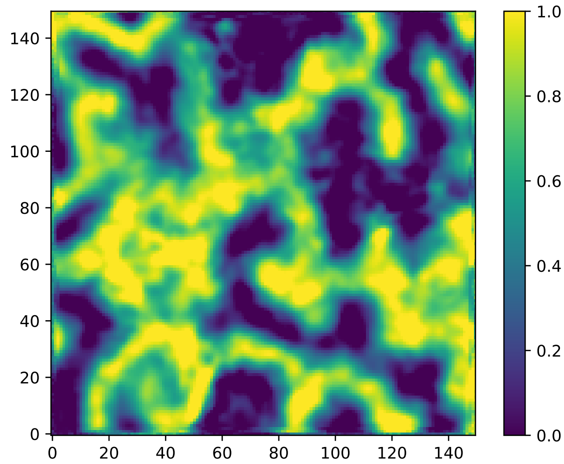
(a) Echte Erregung des Modells



(b) Vorhersage des NN-Ansatzes



(c) Vorhersage des RBF-Ansatzes



(d) Vorhersage des ESN

Abb. 3.8: Graphische Darstellung der u-Variable des *Barkley*-Modells für den 100. Zeitschritt des Testdatensatzes. Oben links ist das tatsächliche Feld des Modells zu sehen. Danach folgenden im Uhrzeigersinn die Vorhersagen des NN-Ansatzes, des RBF-Ansatzes und des ESN.

	Barkley			Mitchell-Schaeffer		
	NN	RBF	ESN	NN	RBF	ESN
Laufzeit [s]	**53**	1840	3604	**42**	1842	3823
MSE	0.10089	0.03899	**0.02347**	0.06217	0.03252	**0.02449**
NRMSE	0.8227	0.5114	**0.3968**	0.9136	0.6913	**0.3599**

Tab. 3.10: Vergleich der benötigten Laufzeit und der erreichten Fehlers der drei Ansätze für das *Mitchell-Schaeffer*- und das *Barkley*-Modell, welche zu den geringsten Fehlern führen.

3.4 Kreuz-Prädiktion innere Dynamiken

Bei Messungen der elektrischen Erregung des Herzens können nach aktuellen Stand meistens nur die Erregungen auf der Herzoberfläche gemessen werden. Die Ausbreitungen im Inneren des räumlich ausgedehnten Herzens bleiben somit verborgen. Zudem ist anzunehmen, dass die Gesamtdynamik nicht nur durch die Oberfläche, sondern auch durch die Erregung im Inneren bestimmt und charakterisiert wird. Somit wird die Frage aufgeworfen, ob die innere Erregung des Herzens nur durch die Kenntnis der Oberflächendynamik vorhergesagt werden kann. In diesem Abschnitt soll versucht werden, diese Fragestellung erneut mit den ESNs und den klassischen Methoden zu untersuchen. Dabei wird diese Frage statt an einem dreidimensionalen Systems an den zuvor bereits benutzten zweidimensionalen Modellen untersucht.

Hierbei wird nur das Feld der Spannungsvariable betrachtet. In diesem $N \times N$ Einheiten großem Feld wird ein Quadrat mit der Seitenlänge a ausgewählt, für dessen Pixel die Spannungsvariable bestimmt werden soll. Dazu wird um das innere Quadrat ein Rahmen der Breite b gewählt und die Spannungsvariable der beinhalteten Pixel als Quelle genutzt. Eine graphische Illustration dieses Aufbaus ist in Abbildung 3.9 dargestellt. Somit wird die Spannung im Inneren für a^2 Punkte durch die Kenntnis der $(a + 2b)^2 - a^2$ umgebenden Pixel bestimmt.

Dieses Szenario ist für die in Tabelle 3.11 angegebenen Parameterkombinationen durchgeführt worden.

3.4.1 Nächste Nachbar Vorhersage

Für die letzte Aufgabe ist erneut zuerst der NN-Ansatz getestet worden. Es ist anzumerken, dass dies nicht alle Parameterkombinationen a, b aus Tabelle 3.11 durch-

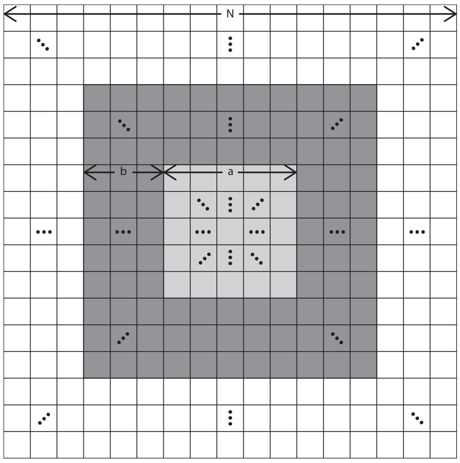


Abb. 3.9: Darstellung des Aufbaus. Das gesamte $N \times N$ große Feld der Spannungsvariable ist in weiß, wohingegen der vorherzusagende Bereich der Größe $a \times a$ in hellgrau dargestellt ist. Drumherum liegt der dunkelgraue Rahmen der Größe $b \times b$ dessen Pixel für die Vorhersage des Inneren genutzt werden.

a	4			8			16			32			64			128*			146*		148*
b	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	2	1	1

Tab. 3.11: Verwendete Parameter a und b für die Abmessungen des inneren und äußeren Quadrates. Die markierten Werte sind nur für die ESNs untersucht worden.

geführt worden ist, da der Rechenaufwand teilweise zu groß geworden ist. Dies liegt daran, dass die Dimension der Eingabevariablen mit

$$(a + 2b)^2 - a^2 = 4b^2 + 4ab$$

skaliert. Da zudem die Rechenzeit für diesen Ansatz nach 2.4 für wachsende Dimensionen sehr stark zunimmt, kann diese Aufgabe für große Abmessungen des vorherzusagendenen Bereiches nicht mehr in einer angebrachten Zeit berechnet werden. Die optimalen gefundenen Hyperparameter und die damit erreichten Ergebnisse sind in Tabelle 3.12 aufgelistet.

Dabei ist zu erkennen, dass die Qualität der Vorhersage mit steigendem a stark abnimmt. So kann für das *Barkley*-Modell nur für $a \in \{4, 8\}$ ein NRMSE der deutlich unter 0.50 erreicht werden. Für größere Bereiche steigt der NRMSE sogar auf > 1.0 an, sodass die Vorhersage nicht besser ist als wenn man den Mittelwert als Vorhersage nimmt. Die Vorhersagen des *Mitchell-Schaeffer*-Modells zeigen eine gleichartige

	Barkley				
a	4	8	16	32	64
b	1	1	1	1	1
δ	4	4	4	3	3
k	5	5	5	5	5
Laufzeit [s]	≈ 1	8	287	1809	14754
MSE	**0.00231**	**0.00891**	**0.07097**	**0.18961**	**0.24599**
NRMSE	**0.0155**	**0.0596**	**0.4779**	**1.3032**	**1.7009**

	Mitchell-Schaeffer				
a	4	8	16	32	64
b	1	1	1	1	1
δ	3	3	3	4	4
k	5	5	5	5	5
Laufzeit [s]	≈ 1	17	194	2482	20272
MSE	**0.14221**	**0.02465**	**0.06460**	**0.08744**	**0.09283**
NRMSE	**0.2663**	**0.4052**	**0.9779**	**1.3564**	**1.4012**

Tab. 3.12: Gefundene Hyperparameter der nächsten Nachbar Vorhersage für das *Barkley*-Modell (oben) und das *Mitschell-Schaeffer*-Modell (unten) für verschiedene Größen a des vorherzusagenden Bereichs, welche zu den geringsten Fehlern führen.

Tendenz, doch starten die Fehler hier bereits deutlich stärker.

3.4.2 Radiale Basisfunktionen

Analog zu den vorherigen Ausführungen sind die radialen Basisfunktionen ebenfalls auf dieses Problem angewendet worden. Dabei ist mit einer analogen Begründung wie bei den nächsten Nachbarn nur der eingeschränkte Wertebereich für a durchlaufen worden. Die dafür gefundenen Hyperparameter und die Fehler können in Tabelle 3.13 gefunden werden.

Es ist anzumerken, dass der NRMSE für beide Modelle und alle betrachteten Größen a kleiner als 1.0 bleibt. Nichtsdestotrotz steigt er ebenfalls mit wachsendem a an, wie schon bei den nächsten Nachbarn. Für die größten beiden a-Werte ist in beiden Modellen der Fehler allerdings schon so groß, dass die Vorhersage kaum nützliche Informationen liefert.

	Barkley				
a	4	8	16	32	64
b	1	1	1	1	1
δ	4	4	4	3	3
σ_{RBF}	9	5	9	9	7
Laufzeit [s]	≈ 2	7	41	279	1845
MSE	**0.00051**	**0.00450**	**0.04009**	**0.08783**	**0.13615**
NRMSE	**0.0586**	**0.1735**	**0.5196**	**0.7769**	**0.9703**

	Mitchell-Schaeffer				
a	4	8	16	32	64
b	1	1	1	1	1
δ	3	3	3	4	4
σ_{RBF}	9	9	9	5	7
Laufzeit [s]	≈ 1	7	43	237	1756
MSE	**0.00064**	**0.00497**	**0.02220**	**0.04745**	**0.05588**
NRMSE	**0.1094**	**0.2857**	**0.5797**	**0.8580**	**0.9184**

Tab. 3.13: Gefundene Hyperparameter der radialen Basisfunktionen für das *Barkley*-Modell (oben) und das *Mitchell-Schaeffer*-Modell (unten) für verschiedene Größen a des vorherzusagenden Bereichs, welche zu den geringsten Fehlern führen.

3.4.3 Echo State Network

Im Gegensatz zu den anderen beiden Methoden wächst die benötigte Rechenzeit bei der Verwendung der ESNs nicht so schnell an, sodass hiermit auch deutlich größere innere Felder betrachtet werden können. Zur Optimierung des Ansatzes ist erneut das in Abschnitt 3.1.1 beschriebene Verfahren durchgeführt worden. Dabei ist ersichtlich geworden, dass der untersuchte Bereich der Regularisierung $\lambda \in [5 \times 10^{-2}, 5 \times 10^{-6}]$ zu gering ist, da der optimale Wert stets am linken Rand des Intervalls gefunden worden ist. Deswegen ist noch einmal eine Suche auf dem größeren Parameterbereich $\lambda \in [5 \times 10^{-4}, 5 \times 10^{4}]$ durchgeführt worden.

Es ist anzunehmen, dass für die Vorhersage eines Punktes der weit von den bekannten Randwerten entfernt liegt, nicht nur durch sein vorheriger Wert und die aktuellen Randwerte benötigt werden. Vielmehr werden die vergangenen Randwerte einen starken Einfluss nehmen. Dies kann an einem Beispiel schnell deutlich gemacht werden: Würde beispielsweise eine ebene Welle durch das Feld propagieren, so können weit entfernte Punkte erst deutlich nachdem die Welle durch die Ränder hindurch gelaufen ist, hiervon beeinflusst werden. Somit benötigt das System eine ausgeprägte Gedächtnisleistung. Bei den ESNs skaliert nach 2.7 die Gedächtnisleistung mit der Größe N des Netzwerkes. Es wäre also anzunehmen, dass möglichst große Reservoirs eine optimale Leistung erzielen können. Dies kann experimentell nicht bestätigt werden. So erzielen zwar teilweise die größtmöglichen Reservoirs ($N = 400$) die besten Ergebnisse, doch tritt gibt es auch Werte für a bei denen kleinere Reservoirs ($N = 50$) besser arbeiten.

Hier kann erneut der Trend beobachtet werden, dass der Fehler mit steigendem a ebenso zunimmt. Für das *Mitchell-Schaffer*-Modell sind alle NRMSE-Werte kleiner als 1.0. Dies ist beim *Barkley-Modell* nicht mehr der Fall für die beiden größten Werte von a.

3.4.4 Vergleich

Für einen Vergleich der drei Methoden bietet es sich an die Ergebnisse für die größten Wert für a durchzuführen, der mit allen drei Ansätzen betrachtet worden ist. Dies entspricht dem Wert $a = 64$. Eine Übersicht der verschiedenen Ergebnisse ist in Tabelle 3.15 zu finden.

Es zeigt sich erneut, dass die Vorhersagen der klassischen Methoden einen höheren Fehler haben. Während der NN-Ansatz bei beiden Modellen schlechtere Ergebnisse als die Vorhersage mittels des Mittelwertes liefert, sind die radialen Basisfunktionen und die ESNs leicht besser als diese. Dabei erreichen die ESNs und radialen Basisfunktionen in etwa die gleiche Genauigkeit. Diese reicht allerdings auch nicht aus, um dies Methoden tatsächlich sinnvoll verwenden zu können.

Des Weiteren wächst der Fehlers mit steigender Größe des vorherzusagenden Bereiches auch bei allen drei Ansätze an. Es ist somit zu vermuten, dass dies nicht nur eine reine Beschränkung der einzelnen Methoden ist, sondern dass es womöglich eine Eigenschaft der betrachteten Modelle ist.

Add comparison images

	Barkley						
a	4	8	16	32	64	128	148
b	1	1	1	1	1	1	1
N	400	400	50	200	400	200	50
$\rho(\|\mathbf{W}\|)$	0.8	0.5	0.5	1.5	1.5	3.0	0.8
α	0.70	0.50	0.20	0.05	0.05	0.20	0.05
ϵ	0.2	0.2	0.1	0.2	0.2	0.2	0.1
ν_{max}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-4}	1×10^{-5}	1×10^{-5}	1×10^{-5}
λ	5×10^{-4}	5×10^{-1}	5×10^{-1}	5×10^{3}	5×10^{4}	5×10^{3}	5×10^{-6}
Laufzeit [s]	5	15	20	131	1206	3318	3010
MSE	**0.00048**	**0.00200**	**0.03460**	**0.08561**	**0.15392**	**0.17220**	**0.18431**
NRMSE	**0.0179**	**0.1156**	**0.4827**	**0.7671**	**0.9472**	**1.0250**	**1.1153**

	Mitchell-Schaeffer						
a	4	8	16	32	64	128	148
b	1	1	1	1	1	1	1
N	400	50	200	50	400	200	200
$\rho(\|\mathbf{W}\|)$	1.5	3.0	3.0	0.8	3.0	3.0	3.0
α	0.95	0.50	0.05	0.95	0.20	0.05	0.05
ϵ	0.1	0.1	0.2	0.1	0.1	0.1	0.2
ν_{max}	1×10^{-4}	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-4}	1×10^{-5}	1×10^{-5}
λ	5	5×10^{-2}	5×10^{-1}	5×10^{3}	5×10^{3}	5×10^{4}	5×10^{4}
Laufzeit [s]	5	15	20	131	1206	3318	3010
MSE	**0.00025**	**0.00220**	**0.01889**	**0.04240**	**0.05389**	**0.06596**	**0.06488**
NRMSE	**0.0687**	**0.1902**	**0.5343**	**0.8110**	**0.9019**	**0.9981**	**0.9844**

Tab. 3.14: Gefundene Hyperparameter der ESNs für das *Barkley*-Modell (oben) und das *Mitchell-Schaeffer*-Modell (unten) für verschiedene Größen a des vorherzusagenden Bereichs, welche zu den geringsten Fehlern führen.

	Barkley			Mitchell-Schaeffer		
	NN	RBF	ESN	NN	RBF	ESN
Laufzeit [s]	20272	1756	**1089**	14754	1845	**1206**
MSE	0.09284	0.05588	**0.05389**	0.24599	0.13615	**0.12975**
NRMSE	1.1837	0.9184	**0.9019**	1.3042	0.9703	**0.9472**

Tab. 3.15: Vergleich der benötigten Laufzeit und der erreichten Fehlers der drei Ansätze für das *Mitchell-Schaeffer*- und das *Barkley*-Modell, welche zu den geringsten Fehlern führen, für $a = 64$.

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] Ezio Bartocci, Pietro Lio, and Nicola Paoletti. *Computational Methods in Systems Biology: 14th International Conference, CMSB 2016, Cambridge, UK, September 21-23, 2016, Proceedings*, volume 9859. Springer, 2016.

[3] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[4] 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.

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

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

[7] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 2000.

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

[9] 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.

[10] 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.

[11] 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.

[12] Holger Kantz and Thomas Schreiber. *Nonlinear time series analysis*, volume 7. Cambridge university press, 2004.

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

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

[15] Colleen C Mitchell and David G Schaeffer. A two-current model for the dynamics of cardiac membrane. *Bulletin of mathematical biology*, 65(5):767–793, 2003.

[16] 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.

[17] Hugo Talbot, Stéphanie Marchesseau, Christian Duriez, Maxime Sermesant, Stéphane Cotin, and Hervé Delingette. Towards an interactive electromechanical model of the heart. *Interface focus*, 3(2):20120091, 2013.

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

Notes