Signalverarbeitung mit neuronalen Netzen vom Typ 'echo state networks'

von
Georg Fette
Matrikelnummer 948955

Diplomarbeit

in Informatik

vorgelegt der
Fakultät für Informatik der TU-Darmstadt
im Mai 2004

Betreuer:
Dr. Julian Eggert und Dr. Marc-Oliver Gewaltig, HRI-EU Offenbach

Gutachter:
Prof. Dr. Oskar von Stryk, Simulations and systems optimization, TU-Darmstadt

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Darmstadt, 27. Mai 2004
Georg Fette

Ich danke Dr. Julian Eggert, der mit viel Geduld meine unsauberen mathematischen Ausführungen auf den richtigen Weg gelenkt hat, sowie Dr. Mark-Oliver Gewaltig der mich mit vielen Anregungen unterstützt hat. Dafür dass ich meine Diplomarbeit in einem interessanten Umfeld anfertigen konnten danke ich der gesamten Belegschaft von Honda-HRI-EU und besonders Volker Willert, Inna Mikhailova, Rasvan Enache und Björn Schölling fuer ihre logistische Unterstützung zwischen Darmstadt und Offenbach. Meinen Eltern und allen meinen Mitstudenten und Freunden danke ich für ihre Hilfe während meines gesamten Studiums.

Inhaltsverzeichnis

1 Einleitung

2001 entwickelten Maass [1] und Jäger [4] unabhängig voneinander eine rekurrente Neuronale-Netz-Topologie, die unter bestimmten Bedingungen erstaunliche Berechnungseigenschaften aufwies. Diese Netze, von Maass 'liquid state machines' und von Jäger 'echo state networks' genannt, konnten Eingangssignale über lange Zeiträume zwischenspeichern und komplexe Verrechnungen der Eingangssignale vornehmen. In den letzten drei Jahren erschienen verschiedene Arbeiten (z.B. [5]), die die neu vorgeschlagene Netzwerkarchitektur von verschiedenen Seiten beleuchteten. Eine vollständige Analyse ist jedoch durch die enorme Komplexität der nichtlinearen rekurrenten Systeme kaum möglich. Ich möchte in dieser Arbeit untersuchen, wie derartige Netze funktionieren, indem ich sie experimentell teste und einige ihrer Eigenschaften analytisch herleite.
Im Folgenden verwende ich den Namen Echo-State-Networks ESN.
In diesem ersten Kapitel möchte ich den grundlegenden Aufbau von ESNs erklären und verschiedene Experimente vorstellen, an denen man ihre Fähigkeiten sehen kann. In Kapitel 2 und 3 untersuche ich die Netze mit linearer Transferfunktion und zeige unter dieser Vorraussetzung einige anschauliche Interpretationsmöglichkeiten wie und warum ESNs funktionieren. Im 4. Kapitel gehe ich auf die Speicherfähigkeit von ESNs ein und wie diese Speicherfähigkeit durch verschiedene Störeinflüsse beeinträchtigt wird. In Kapitel 5 zeige ich analytische Untersuchungen und Experimente um die Fähigkeiten von ESNs mit nichtlinearer Transferfunktion zu untersuchen und gebe Interpretationen der Ergebnisse. In Kapitel 6 wende ich die beschriebene Technik an um einen Geschwindigkeitserkenner auf einem kleinen zweidimensionalen rezeptiven Feld zu konstruieren. Kapitel 7 ist eine Zusammenfassung des Beschriebenen und ein allgemeiner Ausblick.

1.1 Genereller Aufbau

Ein ESN besteht aus einer Inputschicht u, einer Ausgangsschicht o und einer Zwischenschicht x, die im folgenden Echo-Schicht genannt wird. Der grundlegende Aufbau ist in Abbildung 1 gezeigt.
Die Echo-Schicht besteht aus n Neuronen, wobei n meist im Regime von ca. 100 bis 1000 Neuronen liegt, und ist durch die Netzwerk-Matrix A intern verbunden. A ist eine $n \times n$-Matrix, deren Einträge zufällig belegt sind. Dabei entscheidet ein Konnektivitätsparameter $P_{x,x}$ die gleichverteilte Wahrscheinlichkeit, dass eine Verbindung zwischen zwei Neuronen der Echo-Schicht besteht. Bei bestehender Verbindung wird das Gewicht gaussverteilt mit Mittelwert 0 und Varianz 1 gezogen. Jedes Neuron u_i der Eingangsschicht ist mit einer gleichverteilten Wahrscheinlichkeit $P_{u,x}$ mit einen gaussverteilten Gewicht mit den Neuronen der Echo-Schicht verbunden. Das Eingangs-Signal $\mathbf{u}(t)$ wird in das Netz eingespeist und erregt die Neuronen der Echo-Schicht, die mit der Eingangsschicht verbunden sind. Der Netzwerkzustand $\mathbf{x}(t)$ hängt zu jedem Zeitschritt immer von seinem eigenen letzten Zustand $\mathbf{x}(t-1)$, der Gewichtsmatrix des Netzes A und vom zusätzlichen Eingangssignal

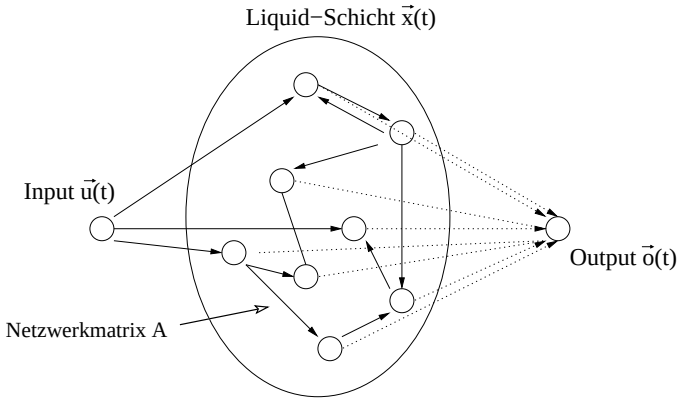

Abbildung 1: grundlegender Aufbau eines Echo-State-Networks

$\mathbf{u}(t)$ zum jeweiligen Zeitschritt ab. Es müssen keine direkten Gewichte vom Eingang zum Ausgang bestehen. Ebenso gibt es vor dem Lernen keine Gewichte von der Echo-Schicht zum Ausgang. Verbindungen vom Ausgang zurück in die Echo-Schicht werden im Kapitel 8.2 'Rückkopplung der Ausgänge' im Appendix behandelt.
Ich definiere folgende Terminologie:

$\mathbf{x}(t) \in \mathbb{R}^n$	Zustandsvektor der Neuronen der Echo-Schicht zum Zeitpunkt t
$\mathbf{x}(0) = (0, ..., 0)^T$	Zustand der Echo-Schicht zum Zeitpunkt 0
$\gamma : \mathbb{R} \to \mathbb{R}$	Transfer-Funktion, gültig für jedes Neuron
$A \in \mathbb{R}^{n \times n}$	Verbindungsmatrix der Echo-Neuronen
$\mathbf{u}(t) \in \mathbb{R}^m$	Eingangsvektor zum Zeitpunkt t
$B \in \mathbb{R}^{m \times n}$	Verbindungsmatrix zwischen den Eingängen und den Echo-Neuronen
$\Delta t = 1$	Zeitschrittweite

Die Dynamik des Netzwerks lässt sich darstellen durch:

$$\mathbf{x}(t) = \gamma(A\,\mathbf{x}(t-1) + B\,\mathbf{u}(t)) \tag{1}$$

Damit das Netz funktioniert muss eine Eigenschaft sichergestellt sein, die von Jäger als 'having fading memory' (äquivalent zu 'state forgetting', 'input forgetting oder 'having echo states' [4]) bezeichnet wurde. Er fordert, dass A einen Spektralradius $\rho_A = max(|\lambda_A|) < 1$ hat, wobei λ_A die Eigenwerte von A sind. Informell verlangt die Bedingung, dass das Netz auf jedes Eingabesignal mit einer Erregungsreaktion antwortet, die mit der Zeit abklingt und durch die man den Informationsgehalt des vergangenen Inputs durch geschicktes Auslesen wieder rekonstruieren kann. Dafür darf das Netzwerk niemals instabil, d.h. in einen übererregten Zustand wechseln, und somit unerregbar durch neue Input-Signale werden. Der Spektralradius kleiner 1 kann sichergestellt werden, indem man die gesamte Matrix durch den Betrag des grössten Eigenwertes teilt. Dadurch werden alle Eigenwerte um diesen Faktor kleiner skaliert und damit auch der Spektralradius.

Des Weiteren fordere ich aus Gründen der Ergebnisgüte der Netze, dass die Matrix A vollen Rang besitzt damit kein Netzwerkzustand auf den Nullvektor abgebildet wird. Jedes Signal klingt mit der Zeit ab, verschwindet jedoch niemals vollständig. Wenn die Netzwerkmatrix wie beschrieben zufällig konstruiert wird, ist die Rangvollständigkeit fast immer gewährleistet. Im Folgenden möchte ich den beschriebenen Aufbau als 'Jägernetz' bezeichnen. Später werde ich die Konstruktion variieren, um spezielle Eigenschaften zu untersuchen.

1.2 Lernen und Auslesen

Das Auslesen des Echo-Pools findet über einen Vektor $\mathbf{w}$ statt, der in einem Lernschritt nach einer Trainingsphase konstruiert wird.

t_{train}	Simulationslänge des Trainings
$\mathbf{w} \in \mathbb{R}^n$	Auslesevektor
$o(t) \in \mathbb{R}$	Ausgangsfunktion
$\mathbf{o} \in \mathbb{R}^{t_{train}}$	Vektor aller Netz-Ist-Ausgänge
$X \in \mathbb{R}^{t_{train} \times n}$	Matrix aller gemessenen Netzwerkzustände

$$o(t) = \mathbf{w}^T \mathbf{x}(t) \tag{2}$$

$$\mathbf{o}^T = \mathbf{w}^T X \tag{3}$$

$$X = (\mathbf{x}(0), \mathbf{x}(1), ..., \mathbf{x}(t_{train})) = \begin{pmatrix} x_1(0) & x_1(1) & \cdots & x_1(t_{train}) \\ x_2(0) & x_2(1) & \cdots & x_2(t_{train}) \\ \vdots & \vdots & \vdots & \vdots \\ x_n(0) & x_n(1) & \cdots & x_n(t_{train}) \end{pmatrix}$$

Der Vektor $\mathbf{o}$ bezeichnet in diesem Zusammenhang eine vektorisierte Variante der diskreten Funktion $o(t)$. Er ist folglich ein Vektor über die Zeit.
Die Konstruktion von $\mathbf{w}$ kann durch ein standard Regressionsverfahren, wie z.B. lineare Regression durchgeführt werden. Die Ausgangsgewichte sind dadurch derart justiert, dass zu jedem gesehenen Netzzustand derjenige Ausgangswert erzeugt wird, der am besten zum Sollausgang passt.

$\hat{\mathbf{o}} \in \mathbb{R}^{t_{train}}$	Vektor aller Netz-Soll-Ausgänge
X^+	Pseudoinverse von X

$$\mathbf{w}^T = \hat{\mathbf{o}}^T X^+ \tag{4}$$

Der derart konstruierte Vektor ergibt auf den Trainingsdaten den minimalen Fehler und führt auch bei Verrechnung folgender, bisher noch unbekannter Input-Signalfolgen mit bemerkenswerten Genauigkeiten auf den gewünschten Ausgangsvektor. Somit konnten trotz

des relativ simplen Aufbaus erstaunliche Fähigkeiten derartiger Netze experimentell nachgewiesen werden.
Bei Jäger sind nicht nur die Echo-Neuronen, sondern auch die Ausleseeinheiten sigmoide Neuronen. In diesem Fall muss $\hat{\mathbf{o}}$ elementweise mit γ^{-1} verrechnet werden um die Transfer-Funktion auf den Ausleseneuronen wieder zu revidieren, bevor auf den Ausgangswerten lineare Regression angewendet werden kann.

1.3 Erste Experimente

Im den folgenden Experimenten wird immer das identische Netz verwendet. Es wird bei jedem Versuch eine Initialisierungsphase durchlaufen in der sich die Echo-Schicht von ihrem Anfangszustand auf Eingangssignale einpendeln kann. Darauf folgt die Trainingsphase, in der die Zustände X der Echo-Schicht notiert werden und mit deren Hilfe der Ausgangsvektor bestimmt wird. Mit der Matrix X der geloggten Echo-Schicht-Zustände aus (3) wird die optimale Gewichtung der Ausgangsgewichte gemäss (4) bestimmt. Dann folgt eine Testphase, in der die Fähigkeit des trainierten Netzes zur Funktionsapproximation ausgetestet wird. Das Netz ist dabei wie beschrieben zufällig spärlich verbunden und zufällig gewichtet. Nach der Verteilung von Kanten und deren Gewichten wird die Echo-Matrix A durch den grössten absoluten Eigenwert geteilt um die 'fading memory'-Eigenschaft zu gewährleisten und dann durch den Skalierungsfaktor $\rho_A \in]0,1[$ auf einen gewünschten Spektralradius skaliert. Das verwendete Netz hat folgende Parameter:

$n = 100$
$P_{x,x} = 0.2 \quad P_{u,x} = 1.0$
Kantengewichte der Eingangs-Verbindungen $\quad b_{i,j} = 0.01$
Spektralradius $\quad \rho_A = 0.9$
Anzahl der Eingänge $\quad m = 1$
$t_{initLength} = 100 \quad t_{trainLength} = 1000 \quad t_{testLength} = 200$
$u(t), \hat{o}(t), o(t) \in \mathbb{R}$
$\gamma(x) = \tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$

1.3.1 Vergangenheits-Rekonstruktion VR

Ich betrachte nun folgende Aufgabe:
Das Netz wird mit einem Eingangssignal $u(t)$ gespeist, das zu jedem Schritt aus einer Gleichverteilung aus dem Intervall $[0, 1]$ gezogen wird. Dazu wird zu jedem $u(t)$ das paarweise passende $\hat{o}(t) = u(t - 60)$ definiert, d.h. das Netz soll das Eingangssignal 60 Schritte verzögert wiedergeben. Training und Test werden wie im letzen Abschnitt beschrieben ausgeführt. In Abbildung 2 sieht man das Testergebnis des Experiments. Wie man sieht

ist es der Regression gelungen die Ausgangsgewichte $\mathbf{w}$ derart zu setzen, dass aus der Echo-Schicht $\mathbf{x}$, in der stetig das eintreffende Eingangssignal vielfältig transformiert und übereinandergelagert wird, diejenigen Eigenschaften herauszufiltern, die benötigt werden, um das vergangene Eingangssignal wieder zu rekonstruieren. Man kann mit demselben Netz und denselben Eingängen auch einen weiteren Readout-Pool w_2, der auf die Input-Output-Paare $\hat{o}(t) = u(t-20)$ trainiert wird, konstruieren. Man beachte also, dass das Training der jeweiligen Readout-Pools unabhängig voneinander geschehen kann, da diese nur versuchen aus dem durch das Eingangs-Signal angeregten, 'hin- und herschwappenden' Echo-Pool die notwendigen Informationen herauszuextrahieren. Aus demselben Netz kann man zu jedem Zeitpunkt gleichzeitig die Eingangs-Signale in eine bis zu 80 Zeitschritte und weiter reichende Vergangenheit herauslesen, wobei die Qualität der Rekonstruktion mit weiter zurückliegender Vergangenheit stetig abnimmt.

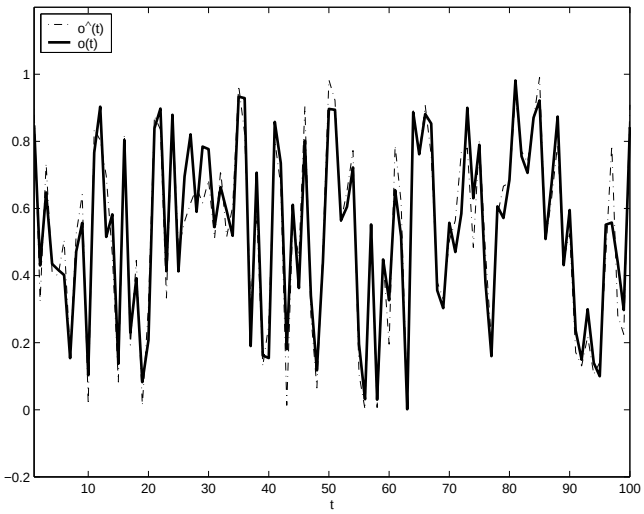

Abbildung 2: $\hat{o}(t)$ und $o(t)$ des Testnetzes. Die Kurven sind fast deckungsgleich.

1.3.2 Mustererkennung

Wir untersuchen nun eine weitere Aufgabe für das gleiche Netz, die wir als Mustererkennung bezeichen. $u(t)$ ist wieder Rauschen aus dem Intervall $[0, 1]$ in das mit einer Wahrscheinlichkeit von 0.02 zu jedem Zeitschritt ein Muster eingefügt wird. Das Muster selbst besteht aus einer 20 Zeitschritte langen Sequenz aus gleichverteilten Zufallszahlen aus dem Intervall $[0, 1]$. Es wird beim Einfügen sichergestellt, dass sich die eingefügten Versionen des Musters nicht überlappen, indem bei den Zeitpunkten an denen das Muster eingefügt wird keine Zufallsziehung zum erneuten Einsetzen vorgenommen werden.
Das gewünschte Ausgangs-Signal soll fast überall 0 sein, nur nach dem jeweiligen Präsentieren des letzten Teils des Musters soll $\hat{o}(t) = 1$ sein. In Abbildung 3 sieht man die Resultate

dieses Experiments. Man kann erkennen, dass zwar ein permanentes Untergrundrauschen an den Nicht-Muster-Stellen auftritt, aber die eigentlichen Detektor-Peaks sind klar zu erkennen und könnten durch einen Schwellwert herausgefiltert werden. Es fielen dabei einige

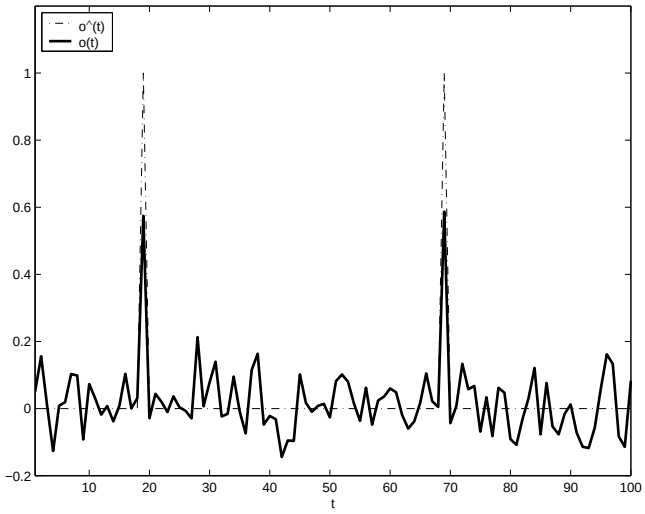

Abbildung 3: $\hat{o}(t)$ und $o(t)$ des Testnetzes. Die Peaks reichen zwar nicht bis 1, die Fehl-Erkennungen haben aber nur einen maximalen Ausschlag von 0.2

Eigenschaften auf, die auch von Jäger experimentell belegt wurden :
- Je länger die Trainingsphase ist, desto besser sind die Ergebnisse des trainierten Netzes. Ab einer gewissen Trainingslänge kann die Qualität jedoch nicht durch noch längeres Training weiter gesteigert werden.
- Man kann einen einzelnen Readout-Pool sogar darauf trainieren auf zwei unterschiedliche Patterns zu reagieren.
- Wenn man während des Trainings das eingefügte Patterns stets ein wenig verrauscht, dann generalisiert das Netz auch auf Variationen des zu erkennenden Musters.
- Man kann den Readout-Pool auch darauf trainieren nicht zu generalisieren, sondern gezielt zu diskriminieren, indem man zusätzlich variierte Versionen des Patterns in den Untergrund einfügt, die aber nicht mit einer 1 im Output finalisiert werden. Die lineare Regression fixiert ihre Optimierung dann speziell auf das reine Basis-Pattern.

1.3.3 Nichtlineare Funktionsapproximierung

Wir betrachten nun eine weitere Variation von nichtlinearen Aufgaben. Das Eingangssignal $u(t)$ wird gleichverteilt aus der binären Menge $\{0, 1\}$ gezogen. Das Soll-Ausgangs-Signal $\hat{o}(t)$ soll 1 sein, wenn in den letzten z Zeitschritten im Eingangs-Signal k mal die 1 erschien, ansonsten soll der Ausgang 0 zeigen. In den Bildern in Abbildung 4 sieht man die Ergebnisse dieser Experimente. Die Bilder zeigen, dass das Netz auch diese Aufgabe zufriedenstellend lösen kann.

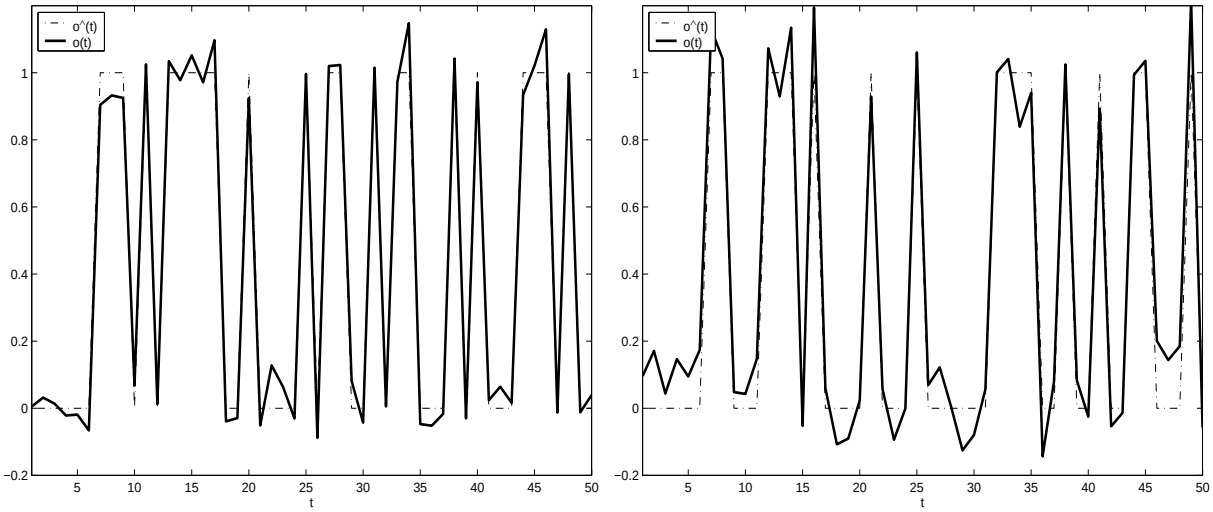

Abbildung 4: $\hat{o}(t)$ und $o(t)$ des Testnetzes. Als Aufgabe ist zu detektieren: (Links) ob in den letzen 2 Eingängen einmal die 1 erschien (auch als XOR bekannt); (Rechts) ob in den letzen 3 Eingängen zweimal die 1 erschien.

1.4 Vorläufige Folgerungen

ESNs sind fähig in ihrem Zustandspool nicht nur eine gewisse Menge der zuletzt eingetroffenen Signale zwischenzuspeichern, sondern diese auch in einer Art zu verrechnen, dass man mit ihnen komplexere nichtlineare Berechnungen durchführen kann.
Die Berechnungsfähigkeiten hängen dabei stark von den verwendeten Parametern ab:

1. Anzahl n der verwendeten Echo-Neuronen:
   Je mehr Neuronen man verwendet, desto längere Vergangenheiten kann man rekonstruieren und desto kompliziertere Funktionen kann man mit besserer Ergebnisgüte approximieren.

2. Skalierung des Input-Signals:
   Die Grösse der Skalierung der Eingänge wirkt sich stark auf die Fähigkeit aus, Vergangenheiten zu rekonstruieren. Zu hohe Eingangsgewichte führen zu fast vollständigem Verlust der Errinnerungskapazität. Zu kleine Eingänge führen zur Beeinträchtigung der Fähigkeit nichtlineare Funktionen zu berechnen. Die Skalierung des Eingangs wirkt sich, wie wir später sehen werden, nur für nichtlineare Netze aus.

3. Spärlichkeit der Echo-Verbindungsmatrix:
   Die Spärlichkeit der Verbindungsmatrix A wirkt sich fast nicht auf die Qualitäten des Netzes aus. Ist die Verbindungswahrscheinlichkeit zwischen den Neuronen jedoch so gering, dass das Netz fast ganz an internem Zusammenhalt verliert, dann verliert

es seine Fähigkeiten.

4. Wahl der Transfer-Funktion γ:
   Die Transfer-Funktion der Neuron ist entscheidend für die Funktionalität des Netzes. Z.B. führt die Identität zu einem enormen Anstieg der Erinnerungskapazität, der $\tanh(x)$ ist dagegen vorteilhaft für nichtlineare Aufgaben.

2 Analytische Untersuchung von ESNs im Linearen

Zur Vereinfachung betrachte ich zunächst nur lineare Netze, bzw. ich verwende die Identität $\gamma(x) = x$ als Transfer-Funktion. Das System wird durch folgende Iterations-Vorschrift beschrieben

$$\mathbf{x}(t) = A\,\mathbf{x}(t-1) + B\,\mathbf{u}(t) \tag{5}$$

2.1 Diagonalisierung

Jede Matrix mit vollem Rang lässt sich mit $A = T^T\,D\,T$ diagonalisieren

$$\begin{aligned} D \in \mathbb{C}^{n\times n}, d_{ij} = 0, i \neq j \qquad & \text{Diagonalmatrix} \\ T \in \mathbb{C}^{n\times n} \qquad & \text{orthonormale Transformationsmatrix} \end{aligned}$$

$$\begin{aligned} \mathbf{x}(t) &= T^T\,D\,T\,\mathbf{x}(t-1) + B\,\mathbf{u}(t) \\ T\,\mathbf{x}(t) &= D\,T\,\mathbf{x}(t-1) + T\,B\,\mathbf{u}(t) \\ \tilde{\mathbf{x}}(t) &= D\,\tilde{\mathbf{x}}(t-1) + \tilde{B}\,\mathbf{u}(t) \\ \Rightarrow \quad \tilde{x}_i(t) &= \tilde{x}_i(t-1)\,d_{ii} + (\tilde{B}\,\mathbf{u}(t))_i \qquad (6) \\ o(t) &= \mathbf{w}\,\mathbf{x}(t) = \mathbf{w}\,T^T\,\tilde{\mathbf{x}}(t) = \tilde{\mathbf{w}}\,\tilde{\mathbf{x}}(t) \qquad (7) \end{aligned}$$

Jedes lineare ESN lässt sich daher in eine Form überführen, in der jedes Neuron nur von seinem eigenen letzten Zustand und vom Eingang der es erregt abhängig ist.
Zum Zwecke der Übersichtlichkeit nennen wir alle $\tilde{\mathbf{x}}$ nun $\mathbf{x}$, alle $\tilde{B}$ nun B und alle $\tilde{\mathbf{w}}$ nun $\mathbf{w}$. Die Matrizen T und T^T sind in die Eingangsmatrix B und die Ausgangsgewichte $\mathbf{w}$ hereinmultipliziert worden. Dies ändert das Verhalten des Netzes nicht, da die Veränderung nur eine Basistransformation im Zustandsraum des Netzes ist.
Da die diagonalisierte Matrix D auch bei reellen Matrizen A selten reellwertig ist, sind nun D, $\mathbf{w}$, $\mathbf{x}$ und B komplex. Auch die Wertebereiche der Ein- und Ausgänge erweitere ich von nun an auf komplexe Werte. Sind sie weiterhin reell, dann versucht die lineare Regression den komplexen Anteil auf 0 zu minimieren. Da im Linearen die Matrizen durch Basistransformationen reell gemacht werden können (siehe Appendix 8.3 'Ersatzschaltbilder'), können die Betrachtungen aber auch weiterhin vollständig im Reellen durchgeführt werden.

2.2 Kernel

Um den zeitlichen Verlauf der Erregung eines einzelnen rekurrent verschalteten Neurons zu beschreiben, untersuche ich zunächst eine zeitliche Dynamik mit diskreter Schrittweite

Δt. In jedem Zeitschritt wird die Erregung mit dem Gewichtungsfaktor d multipliziert, so dass die Dynamik ohne Input lautet:

$$x(t + \Delta t) = x(t)\, d(\Delta t) \tag{8}$$

Da die gemessenen Funktionswerte eine Exponential-Funktion beschreiben, bestimme ich für diese deren homogene Lösung.

$$d(\Delta t) = d^{\Delta t} \tag{9}$$

$$\text{Taylor-Entwicklung um } \Delta t = 0: \quad d^*(\Delta t) = 1 + ln(d)\, \Delta t + \mathcal{O}(\Delta t^2) \tag{10}$$

$$\begin{aligned}
x(t + \Delta t) &= x(t)\, d(\Delta t) \\
x(t + \Delta t) - x(t) &= x(t)\, (d(\Delta t) - 1) \\
\lim_{\Delta t \to 0} \frac{x(t + \Delta t) - x(t)}{\Delta t} &= \lim_{\Delta t \to 0} x(t)\, \frac{d(\Delta t) - 1}{\Delta t} \\
\Rightarrow \quad \lim_{\Delta t \to 0} \frac{x(t + \Delta t) - x(t)}{\Delta t} &= \lim_{\Delta t \to 0} x(t)\, \frac{d^*(\Delta t) - 1}{\Delta t} \\
\text{mit (10)} \quad \Rightarrow \lim_{\Delta t \to 0} \frac{x(t + \Delta t) - x(t)}{\Delta t} &= x(t)\, ln(d) \\
\Rightarrow \quad \dot{x}(t) &= x(t)\, ln(d) \\
\Rightarrow \quad g(t) &= e^{ln(d)\, t} = d^t
\end{aligned} \tag{11}$$

Diese Funktion wird fortan als Kernel $g : \mathbb{R} \to \mathbb{C}$ des Neurons bezeichnet und beschreibt die Antwort des Neurons auf einen Dirac $\delta(t)$.

$$\begin{aligned}
\delta(t) &= 0 \qquad (\text{für } t \neq 0) \\
\int_{-\infty}^{\infty} \delta(t) dt &= 1
\end{aligned} \tag{12}$$

Es ist zu bemerken, dass sich auf der Diagonalen der Diagonalmatrix D die Eigenwerte von A befinden und A, wie anfänglich verlangt, einen Spektralradius kleiner eins sowie vollen Rang besitzt. Daher gilt $0 < |d_{ii}| < 1$. Je näher der Betrag des Gewichts d an eins liegt, desto langsamer fällt die e-Funktion ab. g kann dabei entweder rein reell gedämpft abfallen, wenn d reell ist, oder oszillatorisch gedämpft, wenn d komplex ist. Das Verhalten von $g(t)$ wird vollständig durch die Parameter Dämpfung $\alpha = |d|$ und Winkelfrequenz $\omega = \arg(d)$ beschrieben. In den Bildern in Abbildung 5 sieht man die Graphen einiger Kernel-Funktionen mit unterschiedlichen Gewichten.

Im Folgenden verwenden wir um Echo-State-Netze auf konventionellen Computern simulieren zu können stets diskretisierte e-Funktionen mit Schrittweite $\Delta t = 1$ statt kontinuierlichen Funktionen, statt Integralen über den Zeitbereich Summen über die einzelnen Zeitschritte und statt Diracs diskrete $\hat{\delta}$-Pulse :

$$\hat{\delta}(t) = \begin{cases} \frac{1}{\Delta t} & (\text{für } t = 0) \\ 0 & (\text{für } t \neq 0) \end{cases}$$

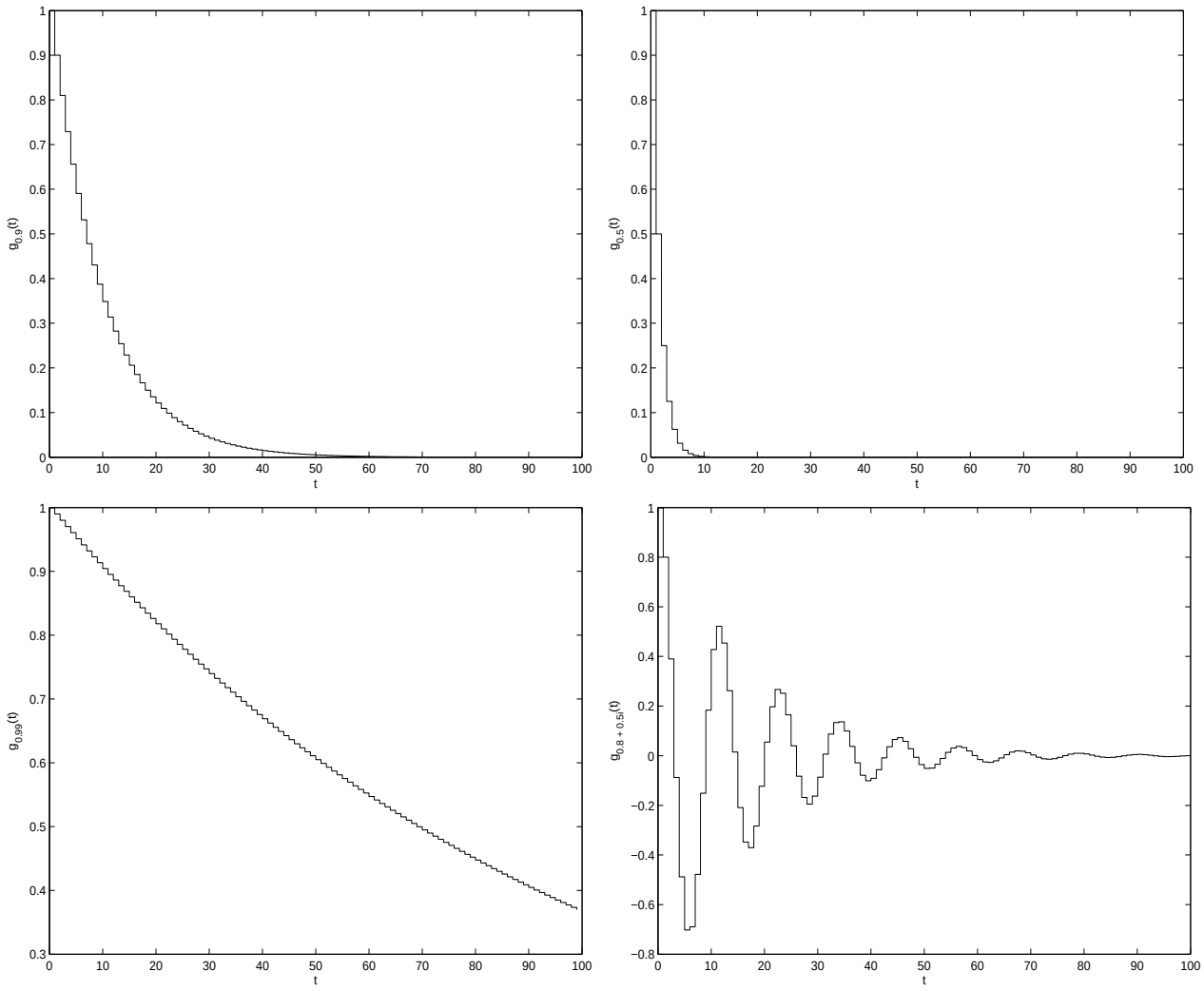

Abbildung 5: Kernel von Neuronen mit Gewichten l.o. 0.9, r.o. 0.5, l.u. 0.99, r.u. 0.8+0.5i

Der Übersichtlichkeit wegen spreche ich im Folgenden trotz der Diskretisierung von δ-Pulsen. Es sind damit jedoch stets $\hat{\delta}$-Pulse gemeint.

Die Reaktion der Neuronen mit rein reellen Exponenten in Form einer Exponential-Funktion erinnert an ein Verhalten, das man bei mehr biologisch orientierten Neuronenmodellen beobachten kann: Neuronen vom Typ Integrate-and-Fire (I&F). Diese besitzen kapazitive Eigenschaften und daraus abgeleitet eine Membranzeitkonstante τ, die das Verhalten des Neurons durch folgende Gleichung festlegt:

$$g(t) = e^{-\frac{t}{\tau}} \tag{13}$$

Man kann Gleichung (11) so umschreiben, dass das rekurrente Gewicht d einer Membranzeitkonstanten entspricht.

$$\begin{aligned} e^{t\,ln(d)} &= e^{-\frac{t}{\tau}} \\ \Rightarrow \quad \tau &= -\frac{1}{ln(d)} \end{aligned} \tag{14}$$

Ein Gewicht von 0.9 entspricht somit einer Membranzeitkonstanten von circa 9.5 Zeitschritten.

2.3 Faltungen

Verallgemeinert man vom δ-Puls als Eingangssignal auf beliebige Funktionen, dann entspricht der Erregungsverlauf eines Neurons einer Faltung seines Eingangssignals mit seinem Kernel (inhomogene Lösung der Differentialgleichung).

$$h_v(t) = (B\,\mathbf{u}(t))_v \quad \text{Eingang am Neuron } v$$

$$x_v(t) = (h_v * g_v)(t) = \sum_{t'=0}^{t} h_v(t')\, g_v(t - t') \tag{15}$$

In den Bildern in Abbildung 6 sieht man das Antwortverhalten eines Neurons mit $d = 0.9$ auf eine Serie von δ-Pulsen.

2.4 Ausgänge

Ein Ausgangsneuron berechnet seine Erregung durch die gewichtete Summation der Erregungen der Echo-Neuronen.

$\mathbf{w} \in \mathbb{C}^n$	Gewichtsvektor von der Echo-Schicht zum Ausgangsneuron
$o(t) \in \mathbb{C}$	Wert der Auslese-Einheit zum Zeitpunkt t

$$o(t) \;=\; \mathbf{w}^T\,\mathbf{x}(t) = \sum_{v=1}^{n} w_v\, x_v(t) \tag{16}$$

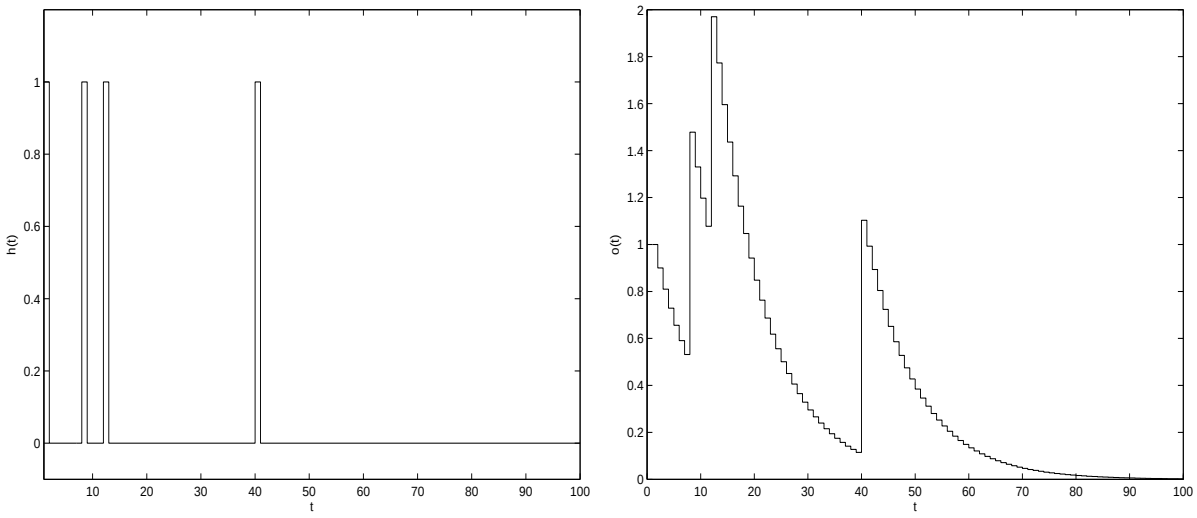

Abbildung 6: Erregungsverlauf eines Neurons mit d = 0.9 (rechts) auf eine Folge von δ-Pulsen (links)

2.5 Eingänge

Es ist günstig, wenn die Eingangsmatrix B im diagonalisierten System bestimmten Bedingungen genügt. Wenn zwei unterschiedliche Eingänge mit demselben rekurrenten Neuron verbunden wären, könnte dieses nicht unterscheiden, von welchem der beiden Eingänge ein eingetroffener Impuls stammt. Für die Aufrechterhaltung der getrennten Informationen der verschiedenen Eingänge sollte jeder Eingang einen separaten Pool an rekurrenten Neuronen besitzen. Wir trennen das Netzwerk deshalb in m disjunkte Bereiche, wobei jeder Bereich für einen Eingang verantwortlich ist.
Die Eingangsmatrix B muss somit von folgender Form sein.

$$B = \begin{pmatrix} b_1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ b_{n/m} & 0 & & \\ 0 & b_{n/m+1} & \cdots & \vdots \\ \vdots & \vdots & & \\ & b_{2n/m} & \cdots & \vdots \\ \vdots & 0 & & 0 \\ & \vdots & & b_{n-m} \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & b_n \end{pmatrix} \tag{17}$$

Da jedes Echo-State-Neuron nur einen Eingang erhält, kann dieses Eingangsgewicht auf 1 normiert werden, da dessen Variabilität durch das Ausgangsgewicht des Echo-Neurons

ausgedrückt werden kann, wie folgend gezeigt wird:

$$\begin{aligned} o(t) &= \sum_{v=1}^{n} w_v \, x_v(t) \\ &= \sum_{v=1}^{n} w_v \sum_{t'=0}^{\infty} g_v(t-t') \, h_v(t') \\ &= \sum_{t'=0}^{\infty} \sum_{v=1}^{n} w_v \, g_v(t-t') \, (b_v \, u_{\lfloor v/m \rfloor}(t')) \\ &= \sum_{t'=0}^{\infty} \sum_{v=1}^{n} (w_v \, b_v) \, g_v(t-t') \, u_{\lfloor v/m \rfloor}(t') \\ &= \sum_{t'=0}^{\infty} \sum_{v=1}^{n} \tilde{w}_v \, g_v(t-t') \, u_{\lfloor v/m \rfloor}(t') \end{aligned} \tag{18}$$

Die Matrix B aus (17) lässt sich somit ohne Einschränkungen vereinfachen zu

$$B = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 1 & 0 & & \\ 0 & 1 & \cdots & \vdots \\ \vdots & \vdots & & \\ & 1 & \cdots & \vdots \\ \vdots & 0 & & 0 \\ & \vdots & & 1 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{19}$$

2.6 Gesamtkernel

Statt den Ausgang durch die gewichtete Aufsummierung von gefalteten Kernelfunktionen zu berechnen, kann man aus den einzelnen Kerneln mit ihren jeweiligen Gewichten einen Gesamtkernel $p(t)$ errechnen, den man mit dem Eingangssignal $u(t)$ falten kann.

$p : \mathbb{C} \to \mathbb{C} \quad = \sum_{v=1}^{n} w_v \, g_v$ Gesamtkernelfunktion

$p(t)$ Antwort der Auslese-Einheit zum Zeitpunkt t nach einem δ-Puls

$$o(t) = \sum_{t'=0}^{t} \sum_{v=1}^{n} w_j \, g_v(t-t') \, h_v(t')$$

$$= \sum_{t'=0}^{t} p(t-t')\, h_v(t')$$
$$= (h * p)(t) \tag{20}$$

Es wird hier angenommen, dass die Eingangsgewichte normiert wurden, damit alle Echo-Neuronen denselben Eingang $h(t)$ erhalten. Diese Betrachtung bezieht sich ohne Einschränkung der Allgemeinheit nur auf einen eindimensionalen Input, da Neuronen, die zu einem weiteren Inputstrom gehören, mit einem unterschiedlichen $h(t)$ gespeisst werden. Weitere Eingänge können mit derselben Methodik separat betrachtet werden und sind unabhängig voneinander.
In Abbildung 7 sieht man ein Beispiel eines Gesamtkernels, der aus der Summation der Einzelkernel g_1 ($d_{1,1} = 0.7$), g_2 ($d_{2,2} = 0.8$) und g_3 ($d_{3,3} = 0.9$) mit den Gewichtungsfaktoren $w_1 = 1$, $w_2 = -2$ und $w_3 = 1$ zusammengesetzt ist. Man sieht, dass sich beim Eintreffen eines δ-Pulses das Maximum der Reaktion erst 10 Zeitschritte später zeigt. Wenn man das Netzwerk nun mit beliebigem Input stimuliert, wird der Output zum grössten Teil vom Input 10 Zeitschritte vorher bestimmt. Ich habe damit ein Netzwerk konstruiert mit der Fähigkeit die Vergangenheit in guter Nähe zu rekonstruieren, bzw. verzögert wiederzugeben.

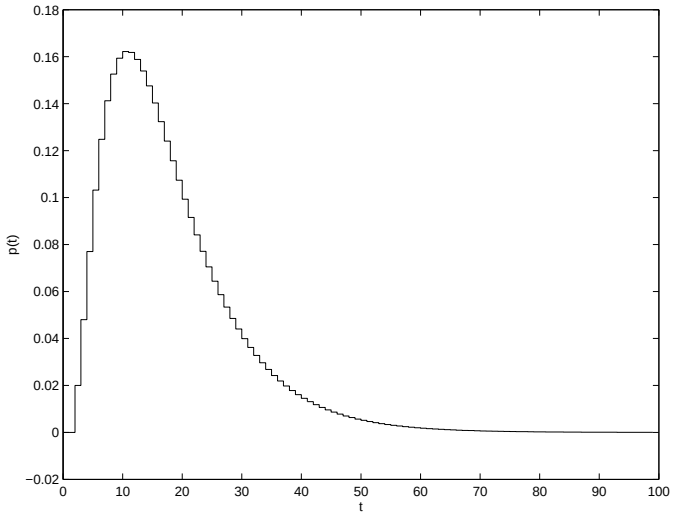

Abbildung 7: kombinierter Kernel aus 3 Einzelkernels

Der Gesamtkernel ist eine Summation aus verschiedenen (mehr oder weniger) oszillierenden e-Funktionen q_v, die sich aus der komplexen Gewichtung der Einzelkernel bilden. Die unterscheidenden Parameter dieser Funktionen sind Anfangsamplitude und Abfallgeschwindigkeit der e-Funktion, sowie Frequenz und Phase der Oszillation. Die Abfallgeschwindigkeit wird wie bereits erwähnt ausgedrückt durch α und die Frequenz durch ω. Die Anfangsamplitude wird durch den Betrag des Ausgangsgewichtes $|w|$, die Phase durch die Winkelfre-

quenz des Ausgangsgewichtes $\arg(w)$ festgelegt.

$q_v(t) \in \mathbb{C}$ Anteil eines Neurons v am Gesamtausgang o(t)

$$\begin{aligned} q_v(t) &= g_v(t)\, w_v \\ &= e^{t\, ln(d_v)}\, e^{ln(w_v)} \\ &= e^{t\, ln(d_v)_r + Re(ln(w_v)) + i\,(t\, ln(d_v)_i + Im(ln(w_v)))} \\ &= |w_v|\, \alpha_v^t\, e^{i\,(t\,\omega_v + \arg(w_v))} \end{aligned} \tag{21}$$

3 Wiederholung der Experimente mit neuem Fokus

3.1 Vergangenheits-Rekonstruktion VR

Ich untersuche nun die anfänglichen Experimente der VR genauer. Ich wiederhole das Experiment mit geändertem Aufbau, indem ich nun ein rein rekurrent verschaltetes Netz konstruiere, was bedeutet, dass A von vornherein eine komplexe Diagonalmatrix ist. Ich verteile die Eigenwerte der zu konstruierenden Matrix willkürlich gleichmässig im komplexen Raum mit $\alpha_v \in [0, 2\pi[$ und $\omega_v \in]0, 1[$.
Nach dem Lernschritt kann man den Kernel des Netzes untersuchen, indem man sich den gewichtet aufsummierten Gesamtkernel errechnet und plottet (Abbildung 8). Bei einer Lernaufgabe, wie z.B. das Eingangssignal in $s = 30$ Zeitschritten Vergangenheit zu rekonstruieren, hat der Kernelgraph eines Netzes mit 50 komplexen Echo-Neuronen einen Ausschlag von 1 bei genau $t = 30$ und sonst nahezu 0. Die lineare Regression hat die verschiedenen e-Funktionen derart kombiniert, dass an allen Stellen ausser $t = 30$ sich die Ausschläge der Einzelfunktionen auslöschen, und nur bei $t = 30$ zu einem Peak aufsummieren.
Das VR-Experiment lässt sich auch wiederholen, indem man Neuronen verwendet, die statt

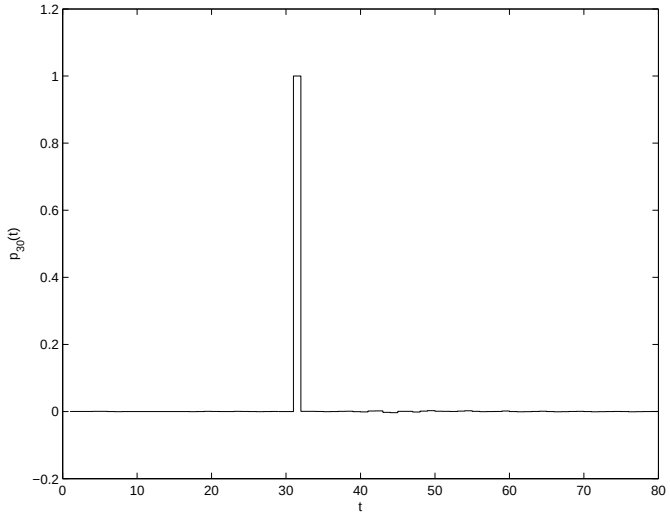

Abbildung 8: Kernel, der auf jede Erregung erst in 30 Schritten in der Zukunft reagiert

einem rekurrentem Gewicht eine Integratorfähigkeit (Membranzeitkonstante) besitzen. Der übrige Netzaufbau bleibt wie gehabt. Diese Konstruktionsweise schränkt die Mächtigkeit der Gesamtkernelbildung stark ein, da nur e-Funktionen mit reellem Exponenten verwendet werden. In Abbildung 9 sieht man einen Kernel eines Netzes mit 60 Neuronen mit rein reellen Gewichten, das darauf trainiert wurde die Vergangenheit $s = 20$ zu rekonstruieren. Man sieht deutlich, wie die lineare Regression versucht den Peak auf $t = 20$ zu setzen und ansonsten zu minimieren.

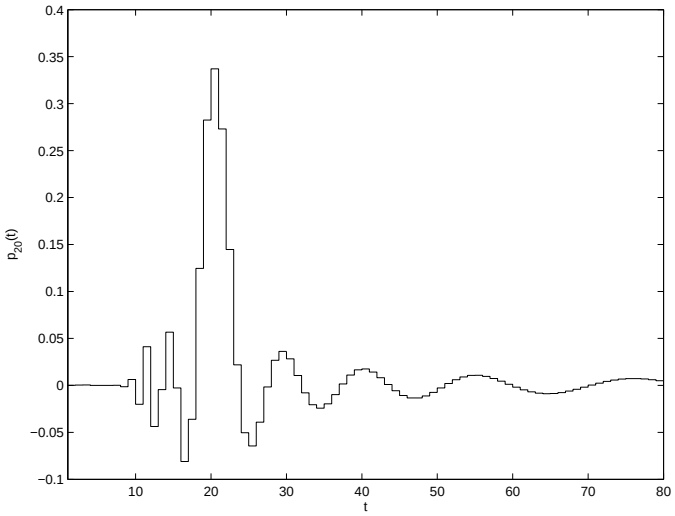

Abbildung 9: Kernel, der versucht erst in 20 Schritten in der Zukunft zu reagieren, dazu aber nur aus e-Funktionen mit reellen Exponenten aufgebaut ist

3.2 Konstruktion von Gesamtkerneln p_s für die VR

Kann man die Ausgangsgewichte, die zur Konstruktion des Gesamtkernels $p_s(t)$ beitragen, auch ohne die lineare Regression konstruieren? Möchte man dies tun, so muss man sich überlegen, was das Kernelbild repräsentiert. Da die einzelnen Neuronen gedämpfte Sinus- und Kosinus-Schwingungen mit unterschiedlichen Frequenzen repräsentieren, kann man sich den Netz-Aufbau als einen diskreten Fourier-Transformations-Apparat vorstellen. Jedes Neuron v detektiert im Eingangs-Signal das Vorkommen seiner Frequenz ω_v in einer Zeitspanne, die seinem Dämpfungsabfall entspricht. Da es sich beim gewünschten Gesamtkernel um einen verschobenen δ-Puls handelt, müssen theoretisch alle Frequenzen mit Amplitude 1 und einer Phasenverschiebung entsprechend der Verschiebung des Diracs beteiligt sein [7].

$$s \in \mathbb{N} \qquad \delta\text{-Puls-Verschiebung}$$

$$\text{verschobener Dirac} \quad \delta_s(t) = \delta(t-s)$$

$$\text{Faltung} \quad \hat{\delta}_s(\omega) = \int_{-\infty}^{+\infty} \delta(t-s) e^{i\,\omega t} dt = e^{-i\,\omega\,s} \tag{22}$$

Fuer die weiteren Erläuterungen mache ich folgende Definitionen:

1. Eine Menge an Winkeln $\omega_v = 2\pi\, v/n, \quad v \in [1, n]$ nenne ich 'homogen'.

2. Ein ESN ist genau dann 'homogen', wenn die Menge der Winkel seiner Eigenwerte 'homogen' ist, die Beträge α_v seiner Eigenwerte alle identisch sind und seine Eingangsgewichte alle 1 sind.

Ich konstruiere nun ein homogenes Netzwerk mit $\alpha = 1$. Jedes Neuron regiert auf eine Erregung ungedämpft mit einer der Frequenzen von 0 bis 2π. Moechte man als Kernel den gewünschten Dirac erhalten, dann muss man diese Frequenzen mit der in (22) verlangten Gewichtung versehen.

$$w_\omega = e^{-i\,\omega\,s} \tag{23}$$

Da der diskrete Dirac an der Stelle s die Amplitude 1 haben soll, müssen alle Gewichte durch die Anzahl der Neuronen geteilt werden.

$$\Rightarrow \quad w_\omega = \frac{e^{-i\omega s}}{n} \tag{24}$$

Gewichtet man die Ausgangsgewichte gemäss (24), wobei $s \in [0, n-1]$ (die Fälle $s \geq n$ werden später im Kapitel 4.3.1 'Störungen durch Geisterbilder' behandelt), so erhält man ein Kernel-Bild (Abbildung 10), das fast dem des durch die Regression Konstruierten entspricht. Der Unterschied besteht darin, dass der Puls sich periodisch alle n Zeitschritte wiederholt. Möchte man die Periodizität entfernen, so kann man den Gewichtsbetrag α auf

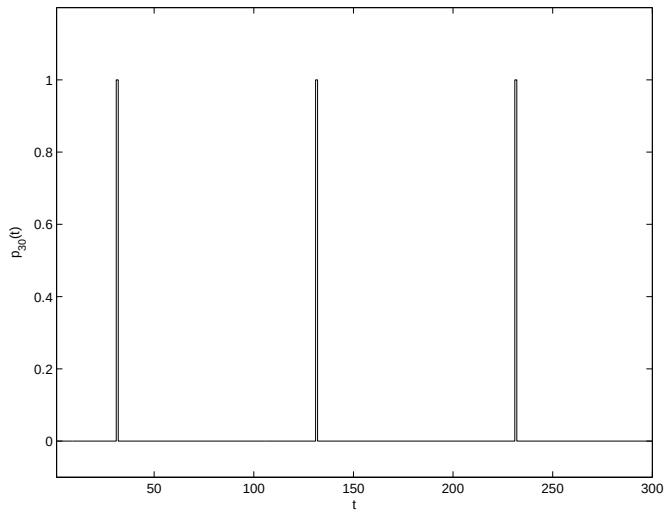

Abbildung 10: Konstruierter Kernel mit 100 Neuronen und Diracs bei 30 + 100n. Alle Neurone haben ein rekurrentes Gewicht mit Betrag 1. Der Dirac ist periodisch ohne Dämpfung.

einen Wert kleiner 1 setzen. Bei $\alpha = 0.9$ ist bei 100 Neuronen der Puls bei 130 auf 0.9^{130} abgesunken. Hiermit ist jedoch der gewünschte Peak bei 30 ebenfalls auf 0.9^{30} reduziert. Da man eigentlich ein Kernelbild $\delta_s(s) = 1$ und $\delta_s(t) = 0, \quad (t \neq s)$ haben möchte muss man die

Ausgangsgewichte entsprechend strecken, um den quadrierten Abstand zum Wunschkernel zu minimieren.

$$\begin{aligned} err(|w|) &= \sum_{t=0}^{\infty} (\delta_s(t) - p_s(t))^2 \\ &= (1 - \alpha^s\, n\, |w|)^2 + \sum_{k=0}^{\infty} (\alpha^{s+k\,n}\, n\, |w|)^2 \\ \frac{\delta err(|w|)}{\delta |w|} = 0 \quad \Rightarrow \quad |w| &= \frac{1}{\alpha^s\, n} + \frac{1}{\alpha^{s-2\,n}\, n} \end{aligned} \tag{25}$$

Für grosse n und einem $\alpha < 1$ führt dies auf einen Streckungsfaktor von $|w| = \frac{1}{\alpha^s\, n}$.

$$\Rightarrow \quad w_\omega = \frac{e^{-i\omega s}}{\alpha^s\, n} \tag{26}$$

Durch Gleichung (26) hat der Puls bei s wieder den Wert 1. Durch die Streckung werden zwar auch alle periodischen Wiederholungen in der ferneren Vergangenheit gestreckt, deren Wert ist jedoch durch die zusätzliche Stauchung von $\alpha^{n\,k}$ im Vergleich zum ersten Puls, bei einer günstigen Wahl von α so gering, dass er gegen 0 läuft. Ich habe in meinen Experimenten α meist auf einen Wert gesetzt, so dass der erste periodisch wiederholte Puls nur noch eine Amplitude von 0.001 des Originals besitzt. Dies war ausreichend um zufrieden stellend kleine Fehler zu erhalten. Da die Entfernung der beiden Pulse direkt von der Anzahl der verwendeten Frequenzen, bzw. Neuronen, abhängt ergibt sich der Zusammenhang $\alpha = 0.001^{1/n}$. Abbildung 11 zeigt einen Kernel in dem mit $\alpha = 0.96$ die Periodizitäten nahezu entfernt wurden.

Da die resultierenden Funktionen nahezu keinen Unterschied mehr zum gewünschten Kernel aufweisen, spreche ich von nun an im Bezug auf den konstruierten Kernel wieder von δ-Funktionen.

Der Konstruktion der verschobenen δ-Pulsen ist jedoch eine Grenze gesetzt. Die diskrete Fourier-Transformation setzt voraus, dass das transformierte Signal sich periodisch nach n Schritten wiederholt. Hat man, wie in unserem Falle, ein aperiodisches Eingangssignal, so vermischen sich die Informationen der Zeitpunkte die den periodischen Wiederholungen entsprechen. Diese Vermischung ist irreversibel und führt zu Störungen. Daher müssen diese 'Geisterbilder' durch den Abfall der e-Funktionen ausgelöscht und der Verfall des gewünschten δ-Pulses durch die Streckung des Ausgangsgewichts wieder revidiert werden. Es ist nicht möglich einen einzelnen Puls zu einem Zeitpunkt $s \geq n$, z.B. bei $\hat{n} > n$, zu erstellen, da es eine periodische Wiederholung mit grosser Amplitude bei $\hat{n} - n$ gibt und diese beiden nicht unabhängig voneinander gestreckt oder gestaucht werden können. Ab $s = n$ treten daher mit dieser Technik grosse Fehler im Kernel und somit in der VR-Fähigkeit auf. Mit einem homogenen Netzwerk mit einem derart gewählten α, dass die periodisch auftretenden δ-Pulse im Kernelbild nahezu verschwinden, kann man dagegen für jede Verschiebung $s \in [0, n-1]$ einen nahezu exakten Dirac konstruieren. Wir sprechen dann davon, dass das Netz fast n Informationseinheiten rekonstruieren kann.

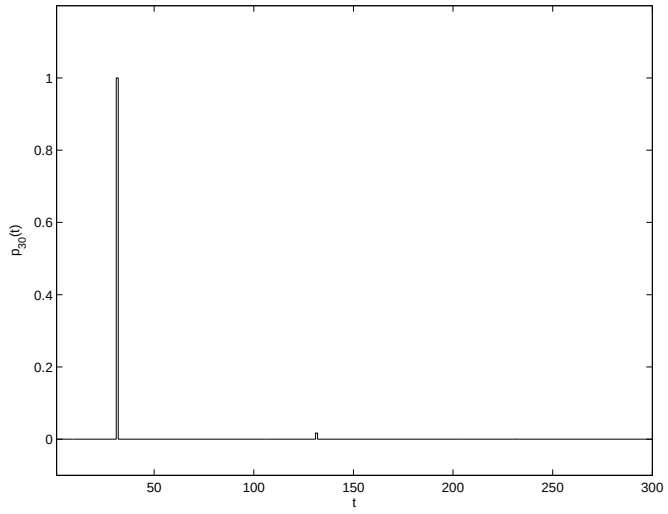

Abbildung 11: Konstruierter Kernel mit 100 Neuronen und Pulsen bei 30 + 100n. Alle Neurone haben $\alpha = 0.96$. Nur der Puls bei 30 hat noch eine Amplitude 1

Diese Aussagen treffen nur zu, wenn das Netzwerk homogen ist, da dann die DFT exakt ist. Sind die α_v unterschiedlich oder die ω_v nicht regelmässig verteilt, dann treten Fehler in der Konstruktion der Kernel auf (siehe Kapitel 4 'Speicherfähigkeit'). Belegt man die Neuronen z.B. mit längeren Frequenzen, mit deren Hilfe man Aussagen über eine fernere Vergangenheit machen könnte, dann geht dies immer zu Lasten der Exaktheit der DFT, wie man in Abbildung 12 sieht.

3.3 Regression auf dem gewünschten Kernel

Da eine einfache analytische Konstruktionsanweisung eines gewünschten Kernels für eine Vergangenheit s nur für homogene Netze gefunden werden kann, muss man sich bei nicht-homogenen Netzen weiterhin der linearen Regression bedienen.
Die Errechnung eines optimalen Kernels p_s lässt sich jedoch durch folgende Systematik effizienter gestalten:
Da sich die Eigenschaften eines linearen Systems vollständig durch seine Impulsantwort beschreiben lassen, kann man die Regression dadurch vereinfachen, dass man sie nicht mehr auf einem Strom von Input-Output-Paaren mit stochastischen Werten ausführt, sondern es reicht, wenn man man eine Regression auf den gewünschten Kernel hin durchführt. Man

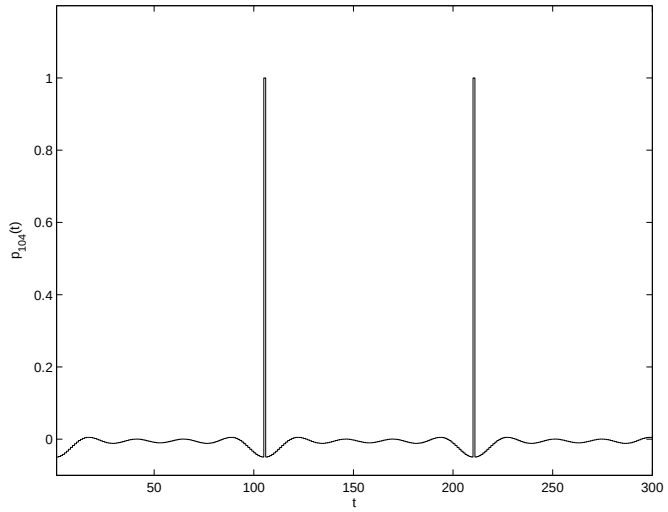

Abbildung 12: Konstruierter Kernel mit 100 Neuronen und δ-Pulsen bei 104 + 105n. Die Frequenzen sind hier gleichmässig zwischen [1/105..100/105] verteilt. Die 'Geisterbilder' wiederholen sich erst alle 105 Schritte. Die DFT ist nicht mehr exakt.

muss also nur noch auf k Zeitschritte weit eine Regression durchführen auf den Paaren

$$\{(u(t), \hat{o}(t))\}\,(t \in [0, k[) \;=\; \left\{ \begin{array}{c} (u(0) = 1,\, \hat{o}(0) = 0), \\ (u(1) = 0,\, \hat{o}(1) = 0), \\ \ldots, \\ (u(s) = 0,\, \hat{o}(s) = 1), \\ \ldots, \\ (u(k-1) = 0,\, \hat{o}(k-1) = 0) \end{array} \right\}$$

k muss dabei so gross gewählt sein, dass möglichst keine Restaktivität in den einzelnen Neuronen mehr vorhanden ist, so dass das Regressionsergebnis nicht durch unzureichende Simulationslänge verfälscht wird. Dabei habe ich stets die willkürliche Grenze $k = \lfloor log(0.0001)/log(max(\alpha_v)) \rfloor$ gewählt, damit ab einer maximalen Restimmanenz von kleiner als 0.0001 der weitere Kernelverlauf als gleich 0 angenommen werden kann. Dies ist zulässig da der Kernelverlauf durch den exponentiellen Verfall monoton fallend ist. Der Vorteil ist, dass diese direktere Regressionskonstruktion im Gegensatz zur Regression auf den Daten des verarbeiteten stochastischen Inputs um einige Grössenordnungen schneller ausgeführt werden kann. Wir werden im Folgenden bei linearen Netzen diese Art der Kernel-Errechnung anwenden.

3.4 Mustererkennung

Da wir uns im Folgenden wieder erst auf lineare Systeme einschränken, darf man die Ausgangsgewichte für einzelne, individuell gestreckte δ-Pulse addieren, um die δ-Pulse zu

superpositionieren. Damit ist die Konstruktion von beliebigen Kernel-Funktionen möglich. Somit kann man sich Funktionen widmen, die zu ihrer Berechnung nicht nur einen bestimmten Punkt in der Vergangenheit benötigen, sondern ganze Zeitabschnitte verrechnen, wie z.B. in den Experimenten zur Mustererkennung.
Es tritt hierbei jedoch das Problem auf, dass man keinen linearen Kernel konstruieren kann, der auf das Muster mit einem maximalen Ausschlag reagiert und auf alle übrigen möglichen Inputs mit einer geringeren Reaktion. Dies ist hierdurch zu zeigen:
Der Kernel bildet zu jedem Zeitschritt die Korrelation der vergangenen Inputs mit dem Muster durch das Ausführen des Skalarproduktes dieser Vergangenheit mit dem Kernel. Bei einer Eingangssignalfolge, die das Muster darstellt, kann man für jeden Zeitpunkt in der Vergangenheit das Eingangssignal erhöhen (bzw. senken, wenn das Kernelgewicht dort negativ ist) und würde einen entsprechend höheren Klassifikationswert erhalten. Da man keine nichtlinearen Effekte zur Verfügung hat um mit diesem Makel das Eingangssignal auf das Auftreten des Musters zu überprüfen, muss man sich einiger Alternativen bedienen.
Eine Möglichkeit das Dilemma zu umgehen ist die Einführung eines weiteren Eingangs, der zu jedem Zeitpunkt das Quadrat des ersten Eingangs zu diesem Zeitpunkt liefert, sowie eines Bias-Neurons in der Echo-Schicht, das mit einer konstanten 1 gespeist wird und ebenso ein trainierbares Ausgangsgewicht besitzt. Der zweite Eingang muss mit einer eigenen Neuronen-Phalanx ausgestattet sein, genauso wie sein lineares Korrelat.
Setzt man den Kernel p_1 des linearen Eingangs auf das zeitlich gespiegelte zu detektierende Muster multipliziert mit -2, den Kernel p_2 des quadratischen Eingangs auf einer Breite, die der Breite des Musters entspricht, auf 1 und sonst auf 0 und weist dem Bias-Ausgang ein Gewicht entsprechend den aufsummierten Quadraten des Musters M zu, so ergibt sich folgender Zusammenhang:

$$
\begin{aligned}
& p_1 : (-2\,M(t), -2\,M(t-1), ..., -2\,M(2), -2\,M(1), 0, 0, ...) \\
& p_2 : (1, 1, ..., 1, 1, 0, 0, ...) \\
o(t) &= (u * p_1)(t) + (u^2 * p_2)(t) + \sum_{i=0}^{t} M(i)^2 \\
&= \sum_{i=0}^{t} u(t-i)\, p_1(i) + \sum_{i=0}^{t} u(t-i)^2\, p_2(i) + \sum_{i=0}^{t} M(i)^2 \\
&= \sum_{i=0}^{t} u(t-i)\,(-2)\, M(t-i) + \sum_{i=0}^{t} u(t)^2\, 1 + \sum_{i=0}^{t} M(t-i)^2 \\
&= \sum_{i=0}^{t} u(t-i)\,(-2)\, M(t-i) + u(t-i)^2 + M(t-i)^2 \\
&= \sum_{i=0}^{t} (u(t-i) - M(t-i))^2 \\
&= \sum_{i=0}^{t} (u(i) - M(i))^2 \qquad (27)
\end{aligned}
$$

D.h. diese Wahl der Kernel berechnet den aufsummierten quadratischen Fehler des Eingangssignals zum Muster.
Durch den zusätzlichen nichtlinearen Eingang konnte darauf verzichtet werden Nicht-Linearitäten in die Struktur des Netzwerks einzuführen, dafür sind doppelt so viele Echo-Neuronen notwendig. Wenn man die Gesamtanzahl der Neuronen nicht erhöhen will, dann muss man die Anzahl der Neuronen in den jeweiligen Pools reduzieren. Führt man, wie hier, einen Pool ein, der genauso viele Neuronen beinhalten soll wie der lineare Pool, dann haben beide nur die Hälfte der Speicherfähigkeit des ursprünglich doppelt so grossen Pools. Das Einführen von Nichtlinearitäten durch zusätzliche Eingänge mit gleich bleibender Neuronenzahl geht somit indirekt zu Lasten der Speicherfähigkeit.
Der sich ergebende Ausgang bei gezeigtem Netz-Aufbau ist um einiges besser als das einfache Modell mit rein linearem Eingang. Es besitzt die Eigenschaft, dass nur beim Präsentieren des Musters der grösste Ausschlag erreicht wird. Jedoch sind auch bei allen anderen Eingangssignal-Folgen, die dem Muster relativ ähnlich scheinen, der Ausgang relativ hoch, da der Ausgang nichts Anderes als der aufsummierte quadratische Fehler ist. Man kann dies als einen angenehmen Effekt der Generalisierung betrachten, den Anspruch der exakten Erkennung des Musters und ansonstigem Null-Ausgang erfüllt diese Netztopologie jedoch nicht. Um diesem Anspruch gerecht zu werden kann man weitere Korrelationen des Eingangs mit sich selbst und seiner Vergangenheiten als weitere Eingänge zuführen. Man kann z.B. folgenden Aufbau verwenden:

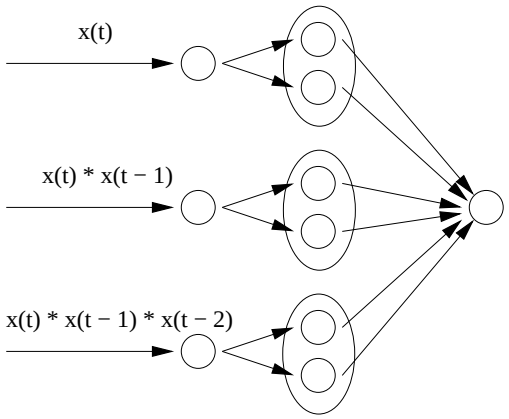

Abbildung 13: Aufbau eines linearen Netzes, dass mit nichtlinear korrelierten Eingängen arbeitet.

Ein Netz, erhält 5 Eingänge ($x_1(n) = x(n)$, $x_2(n) = x(n)\,x(n-1)$, $x_3(n) = x(n)\,x(n-2)$, $x_4(n) = x(n)\,x(n-3)$, $x_5(n) = x(n)\,x(n-1)\,x(n-2)$) mit Echo-Pools mit jeweils 20 Neuronen (Abbildung 13). Der Lernschritt wird wieder per linearer Regression ausgeführt. In den Bildern in Abbildung 14 kann man sehen, wie ein solches trainierte Netz mit unbekannten Daten getestet im Vergleich mit anderen Netzen abschneidet. Das Netz mit nichtlinearer Transfer-Funktion bringt zwar die besten Ergebnisse, das lineare Netz mit nichtlinearen Eingängen schneidet jedoch entscheidend besser ab als das nur mit li-

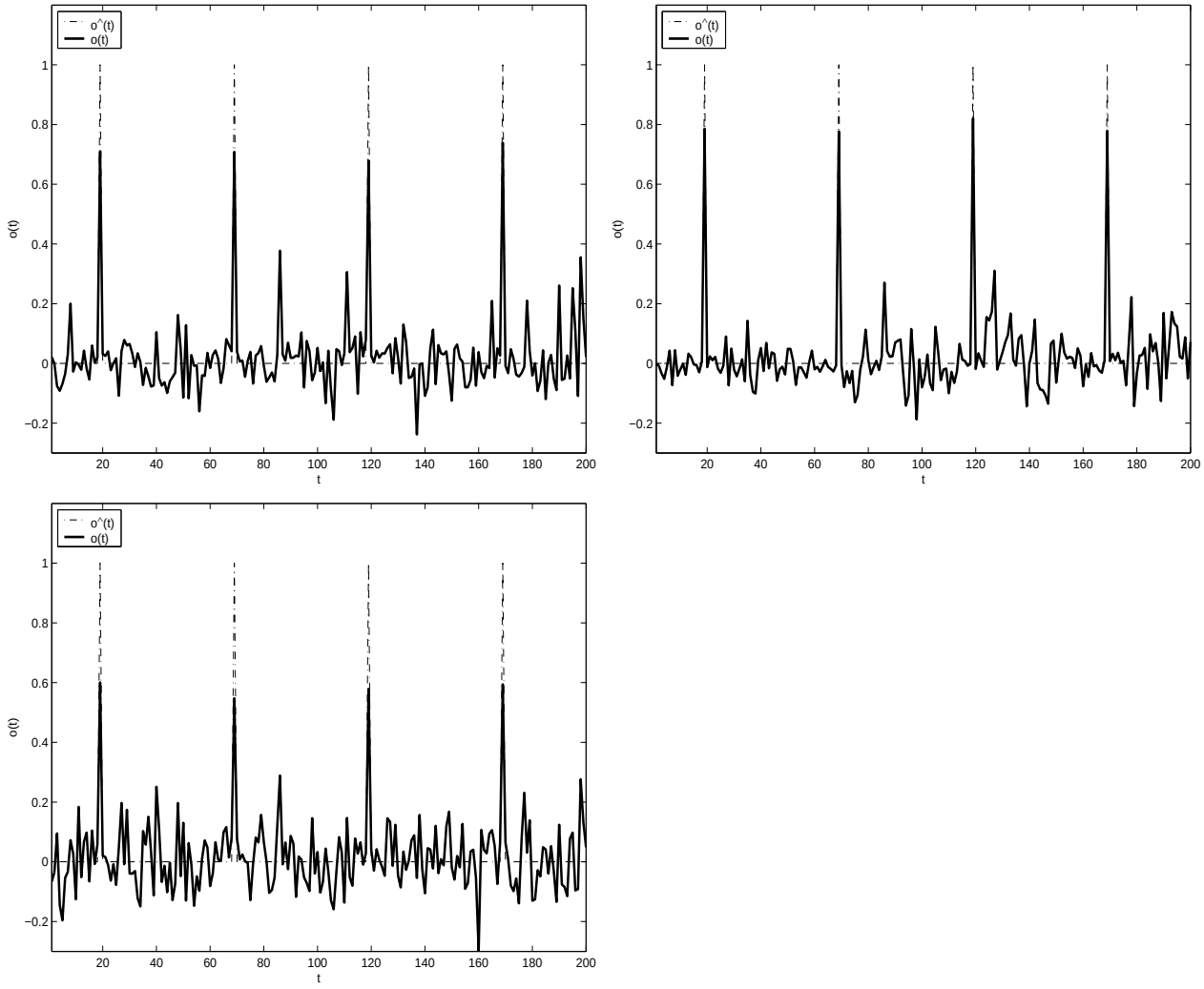

Abbildung 14: $o(t)$ und $\hat{o}(t)$ verschiedener Varianten trainierter Netze bei der Mustererkennung: mehrere nichtlinear korrelierte Eingänge, lineare Transfer-Funktion (l.o.); linearer Eingang, nichtlineare Transfer-Funktion (r.o.); linearer Eingang, lineare Transfer-Funktion (l.u.).

nearem Eingang gespeiste Netz. Es kann also vermutet werden, dass lineare Netze mit Eingängen, die ausreichend korreliert sind (z.B. wie hier durch Multiplikationen mit vergangenen Eingängen), äquivalente Berechnungsstärke wie nichtlineare Netze besitzen können, bzw. dass nichtlineare Netze Berechnungen durchführen können, die Korrelationen ähneln.

4 Speicherfähigkeit

Wir haben im letzten Kapitel gesehen, dass homogene ESNs aus n Neuronen theoretisch die Fähigkeit besitzen annähernd genau n Informationszustände der Vergangenheit zu speichern. Durch Effekte, wie z.B. Rechenungenauigkeiten oder wenn das Netz nicht homogen ist, kann diese Fähigkeit beeinträchtigt werden. Ich möchte in diesem Kapitel untersuchen, wie stark sich die verschiedenen Störungsquellen auf die Speicherfähigkeit auswirken. Dafür möchte ich die Definition der 'memory capacity' MC von Jäger [3] verwenden, sowie den durchschnittlichen quadratischen Fehler $mse(s)$.

4.1 Memory capacity (MC)

Die partielle memory capacity für eine Vergangenheit s ist definiert als [1]

$$mc(s) = \frac{(cov(\hat{\mathbf{o}}_s, \mathbf{o}_s))^2}{\sigma^2(\hat{\mathbf{o}}_s)\,\sigma^2(\mathbf{o}_s)} \tag{28}$$

Die $mc(s)$ ist die normierte Korrelation zwischen zwei Signalen. Ihr Wert liegt stets zwischen 0 und 1, wobei 1 auftritt bei Ergebnissen, bei denen Soll- und Ist-Ausgang bis auf Skalierungen identisch sind, sowie 0 bei Ergebnissen, bei denen Soll- und Ist-Ausgang keinerlei Abhängigkeit besitzen. Es ist zu beachten, dass der Wert unabhängig von Streckungen der einzelnen Signale ist.

Die vollständige memory capacity ist definiert als die Summe über alle partiellen memory capacities.

$$MC = \sum_{s=0}^{\infty} mc(s) \tag{29}$$

Jäger bewies in Proposition 4 in [3]

$$N = \sum_{v=1}^{n} A^v \quad \wedge \quad N \text{ hat vollen Rang} \quad \Rightarrow \quad MC_A = n \tag{30}$$

Die Vorbedingung ist für Netze mit paarweise verschiedenen Frequenzen immer erfüllt, da n verschiedene Frequenzen eine n-dimensionale Basis bilden [7] und N somit vollen Rang

[1]

$$\begin{aligned} \mu(f) &= \langle f(i) \rangle_i \\ \sigma^2(f) &= \langle (f(i) - \mu(f))^2 \rangle_i \\ cov(f,h) &= \langle f(i)\, h(i) \rangle_i \end{aligned}$$

besitzt. Da zufällige Matrizen sehr selten identische Frequenzen besitzen, kann man praktisch volle $MC = n$ für Jägernetze voraussetzen.

Durch die Konstruktion der Dirac-Kernel mithilfe der DFT aus Kapitel 3.2 kann man im speziellen Fall eines homogenen Netzes den Beweis von Jäger alternativ führen:

Es sei angenommen, dass in einem homogenen Netz der optimale Kernel p_s für eine Vergangenheit s mit Gewichten

$$w_v = q_s \, e^{-i \, \omega_v \, s} \tag{31}$$

konstruiert wird, wobei q_s einen noch zu wählenden aber in dieser Betrachtung beliebig wählbaren Streckungsfaktor darstellt. Der Kernel ist somit

$$p_s(t) = q_s \, n \sum_{k=0}^{\infty} \alpha^{mod(s,n)+n\,k} \, \delta(t - mod(s,n) - n\,k) \tag{32}$$

wobei $mod(a, b)$ den ganzzahligen Rest beim Teilen von a durch b darstellt.
Speist man das Netz mit weissem Rauschen $u(t) = \xi$ mit $\mu_\xi = 0$ und $\sigma_\xi^2 = 1$, dann ergibt sich ein Ausgang von

$$o_s(t) = (u * p_s)(t) = \sum_{t'=0}^{\infty} u(t - t') \, p_s(t') \tag{33}$$

Die Verteilung der stochastischen Variable $o_s(t)$ ergibt sich durch die Aufsummierung von unterschiedlich stark gewichteten stochastischen Variablen $\xi_{t'}$ $(t' \in [0, \infty])$, unter der Annahme, das das System bereits unendliche lange läuft und somit nicht mehr durch Initialisierungs-Effekte beeinflusst wird. Die Gewichtung zu einem Zeitpunkt t ist für jedes $\xi_{t'}$ jeweils der Faktor $p_s(t')$. Da die resultierende Gaussverteilung aus Additionen von Gaussverteilungen mit Streckungsfaktoren $k \in \mathbb{C}$ durch $\mu_{sum} = \sum_j k_j \, \mu_j$ und $\sigma_{sum}^2 = \sum_j |k_j|^2 \, \sigma_j^2$ angegeben werden kann [10], ergibt sich

$$\sigma^2(o_s) = \sum_{t'=0}^{\infty} (p_s(t'))^2 \tag{34}$$

Aus (32), (33) und (34) folgt

$$\begin{aligned} mc(s) &= \frac{(cov(u(t-s), o_s(t)))^2}{\sigma^2(u) \, \sigma^2(o)} \\ &= \frac{(cov(\xi_{t-s}, \sum_{t'=0}^{\infty} \xi_{t-t'} \, p_s(t')))^2}{1 \sum_{t'=0}^{\infty} (p_s(t'))^2} \\ &= \frac{(p_s(s))^2}{\sum_{t'=0}^{\infty} (p_s(t'))^2} \qquad\qquad (\text{da} \quad cov(\xi_t, \xi_{t'}) = \sigma_\xi^2 \, \delta(t - t')) \end{aligned}$$

$$
\begin{aligned}
&= \frac{(q\,n\,\sum_{k=0}^{\infty}\alpha^{mod(s,n)+n\,k}\,\delta(s-mod(s,n)-n\,k))^2}{\sum_{t=0}^{\infty}(q\,n\,\sum_{k=0}^{\infty}\alpha^{mod(s,n)+n\,k}\,\delta(t-mod(s,n)-n\,k))^2} \\
&= \frac{\alpha^s}{\sum_{k=0}^{\infty}\alpha^{mod(s,n)+n\,k}} \\
&= \frac{\alpha^{2\,n\,\lfloor s/n\rfloor}}{\sum_{k=0}^{\infty}\alpha^{2\,n\,k}} \qquad (\lfloor a/b\rfloor = (a-mod(a,b))/b \quad a,b\in\mathbb{N}) \qquad (35)\\
MC &= \sum_{s=0}^{\infty} mc(s) \\
&= \sum_{k=0}^{\infty}\sum_{m=0}^{n-1}\frac{\alpha^{2\,n\,\lfloor(n\,k+m)/n\rfloor}}{\sum_{k'=0}^{\infty}\alpha^{2\,n\,k'}} \\
&= \sum_{m=0}^{n-1}\frac{\sum_{k=0}^{\infty}\alpha^{2\,n\,k}}{\sum_{k'=0}^{\infty}\alpha^{2\,n\,k'}} \\
&= n \qquad \square \qquad (36)
\end{aligned}
$$

Es ist anzumerken, dass die MC nur lineare Korrelationen entdecken kann. Informationsspeicherung, die unter Umständen nichtlinear stattfindet, wird von ihr nicht erfasst. Dies ist vor allem bei den Kapiteln über Nicht-Linearitäten zu beachten, bei denen die Netze vergangene Inputs selbst dann noch speichern und verrechnen können, wenn man eine MC von 0 festgestellt hat, was im Linearen bedeutet, dass keine Information mehr vorhanden ist.

4.2 Mean square error für Vergangenheit s ($mse(s)$)

Die Aussagekraft der $mc(s)$ besteht nur unter der Annahme, dass das Netz mit weissem Rauschen gespeist wird. Mit einem autokorreliertem Eingangssignal nimmt der $mc(s)$ beim selben Netz fälschlicherweise zu. Speist man das Netz z.B. mit einer konstanten 1 und leitet diese ohne Veränderung zum Ausgang durch, dann hätte man eine unendliche Speicherfähigkeit. In manchen Fällen ist daher der $mse(s)$ aussagekräftiger als die $mc(s)$. Der mittlere Vergangenheitsfehler für eine Vergangenheit s ist definiert als

$$mse(s) = \langle(\hat{o}_s(t) - o_s(t))^2\rangle_t \qquad (37)$$

Es ist zu bemerken, dass die lineare Regression $mse(s)$ optimiert und nicht $mc(s)$. $mse(s) \to 0$ ist jedoch hinreichend für $mc(s) \to 1$, daher lässt sich meist ebenso gut mit dem $mc(s)$ argumentieren. Der $mse(s)$ ist nicht normierbar, dafür ist er sehr intuitiv und nicht beschränkt auf Aussagen über lineare Zusammenhänge.

4.3 Störquellen

In allen Unterkapiteln bis auf 'Störungen durch Nicht-Linearitäten' sind die Experimente mit linearer Transferfunktion, bzw. mit der Identität als Transferfunktion, durchgeführt worden.
Bei der Berechnung der MC muss man beachten, dass Ungenauigkeiten auftreten können, wenn Werte für α vorhanden sind, die nahe an 0 oder an 1 liegen, da dann die im Appendix im Beweis 8.4 invertierte Matrix Z singulär wird. Im stochastischen Experiment mit einem allgemeinen ESN, das mit weissem Rauschen gespeist wird, wird man somit kaum eine $MC = n$ feststellen können. Es sind daher fast alle folgenden Experimente nicht durch Input-Output-Paare aus weissem Rauschen erstellt worden, sondern die Regression wurde auf den gewünschten Kernel hin ausgeführt (siehe Kapitel 3.3 'Regression auf dem gewünschten Kernel'). Die $mc(s)$, bzw. MC, wurde aus der Netzwerkmatrix und den errechneten Ausgangsgewichten errechnet, indem mit linearer Regression ein optimaler Kernel erzeugt wurde, und die $mc(s)$ direkt aus dem Kernel an der Stelle s abgelesen wurde ($p_s(s)$, siehe Gleichung (70) im Appendix).

4.3.1 Störungen durch Geisterbilder

Ist das Netzwerk homogen so sind die optimalen Ausgangsgewichte jeweils $w_v = q\, e^{-i\,\omega_v\, s}$. Es entstehen neben dem δ-Puls bei s weitere Pulse bei $t = mod(s,n)+n\,k \quad (k \neq \lfloor s/n \rfloor, k \in \mathbb{N}^0)$. Diese 'Geisterbilder' haben die Amplitude $n\,q\,\alpha^t$. Man kann sehen, dass für ein α nahe 1 die Geisterbilder nur langsam abklingen. Ein in das Netz eingespeister Input ruft in diesem Fall Reaktionen hervor, die sich mit den Reaktionen der Inputs jeweils n Schritte vorher mischen. Diese Mischungen können vom Netz nicht mehr getrennt werden, was Störungen hervorruft, die umso grösser sind, je ausgeprägter die Geisterbilder im Kernel sind. Die MC wird durch Geisterbilder nicht beeinflusst, da wir bereits gezeigt haben, dass in homogenen Netzen $MC = n$ gilt. Die $mc(s)$, die in den Vergangenheiten von 0 bis $n-1$ zu 1 fehlt, findet sich im Bereich n bis ∞ wieder. Dieser Zusammenhang wird bereits durch Gleichung (35) dargestellt. Ab $s = n$ tritt die erste Phalanx von Geisterbildern auf, die einen abgeschwächten Block an $mc(s)$ hervorruft. Der Anteil von der gesamten MC in diesen Segmenten ist proportional zu den Anteilen der Geisterbild-Amplituden am summierten normierten Kernel. Dies kann man in Abbildung 15 bestätigt sehen.
Der $mse(s)$ vergrössert sich durch den periodischen Effekt. Je grösser die Ausprägungen der Geisterbilder und umso niedriger der $mc(s)$ desto mehr vermischen sich die Informationen zu den entsprechenden Vergangenheiten und desto grösser wird der $mse(s)$. Aus (37) und (32) folgt bei VR-Experimenten

$$mse(s) \;=\; \sum_{t=0}^{\infty} (\delta_s(t) - p_s(t))^2$$

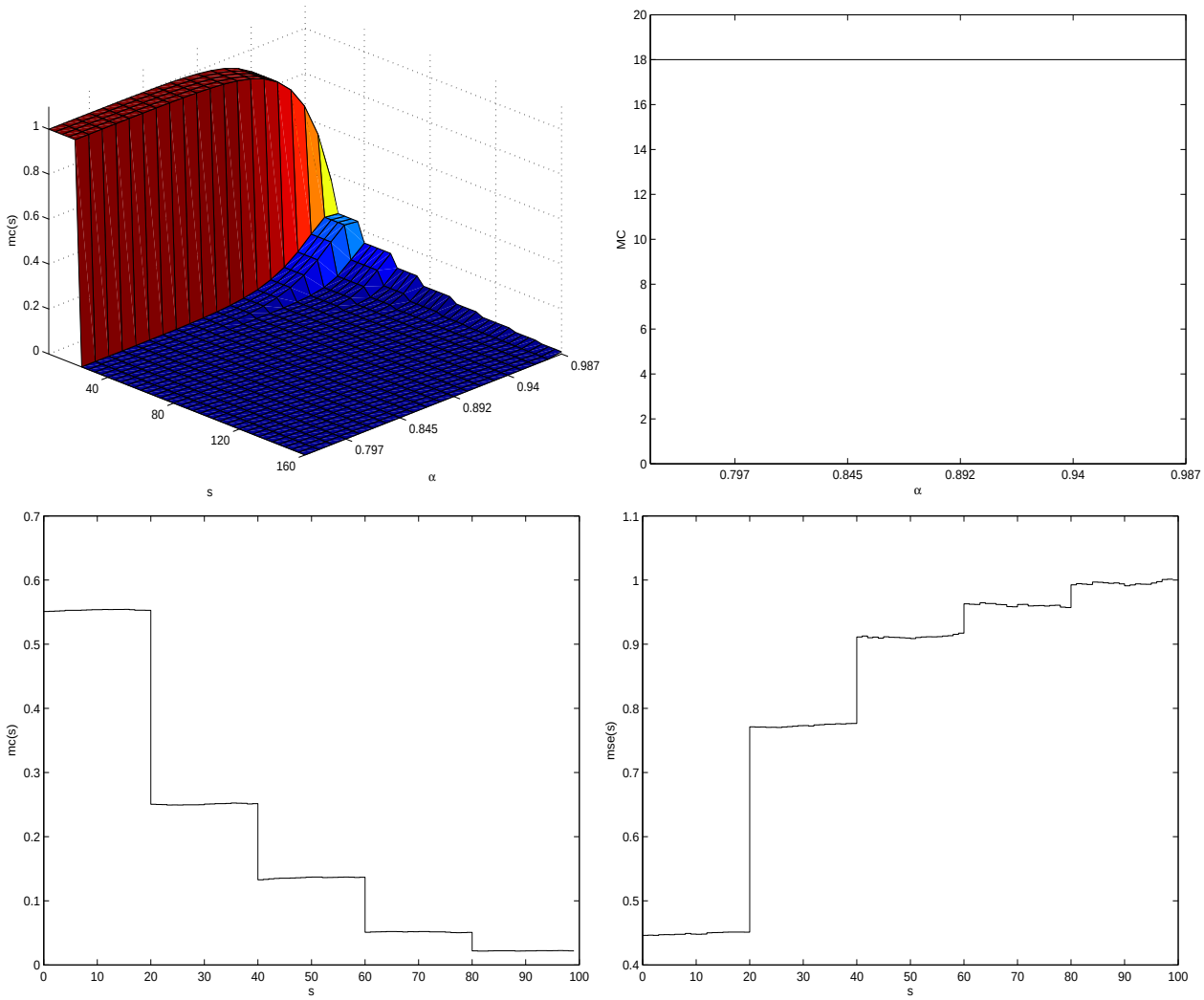

Abbildung 15: $mc(s)$ (l.o.) und MC (r.o.) für variierende α, $mc(s)$ (l.u.) und $mse(s)$ (r.u.) bei $\alpha = 0.98$ mit 20 homogenen Neuronen. Wir sehen in den oberen beiden Bildern, dass sich die Verteilung der MC auf die verschiedenen $mc(s)$ bei unterschiedlichem α ändert, die Aufsummierung über alls $mc(s)$ bei beliebiegem α jedoch immer konstant bleibt. In den unteren beiden Bildern kann man beobachten, wie die $mc(s)$ alle n Schritte treppenstufig abfällt und der $mse(s)$ entsprechend ansteigt, da bei jeder Stufe ein weiteres neues 'Geisterbild' vor dem gewünschten Peak bei s auftaucht.

$$
\begin{aligned}
&= (1 - q\,n\,\alpha^s)^2 + \sum_{k=0, k\neq\lfloor s/n\rfloor}^{\infty} (\alpha^{mod(s,n)+\,n\,k})^2 \\
&= 1 - 2\,q\,n\,\alpha^s + \sum_{k=0}^{\infty} (\alpha^{mod(s,n)+\,n\,k})^2 \qquad (38)
\end{aligned}
$$

Das optimale q_s für eine Vergangenheit s ist bestimmt durch

$$
\begin{aligned}
\frac{\delta mse(s)}{\delta q} &= -2\,n\,\alpha^s + \frac{2\,n^2\,\alpha^{2\,mod(s,n)}}{1-\alpha^{2\,n}} \\
\Rightarrow \quad q_s &= \frac{(1-\alpha^{2\,n})\,\alpha^s}{n\,\alpha^{2\,mod(s,n)}} \\
&= \frac{(1-\alpha^{2\,n})\,\alpha^{-s+2\,n\,\lfloor s/n\rfloor}}{n} \qquad (39)
\end{aligned}
$$

Diesen Zusammenhang kann man in Abbildung 16 beobachten, da alle Diracs die durch die Gleichung (32) und (39) vorhergesagte Amplitude haben.
Eine andere Möglichkeit mit einem fixen α den $mse(s)$ zu reduzieren ist die Anzahl n an Neuronen zu erhöhen.

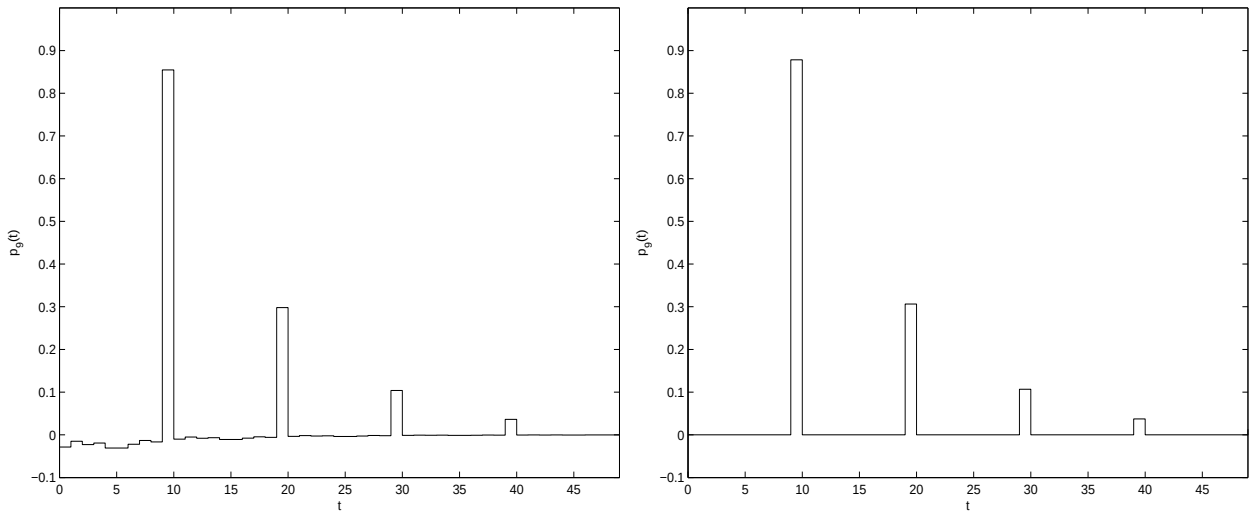

Abbildung 16: $p_9(t)$ bei einem homogenen Netz aus 10 Neuronen und $\alpha = 0.9$ mit $s = 9$; links: durch stochastisches Experimenent nach 2000 Trainingsschritten ermittelter Kernel; rechts: nach Gleichungen (32) und (39) ermittelter Kernel

4.3.2 Abweichung vom homogenen Netz durch unterschiedliche α_v

Abweichungen vom homogenen Netz durch unterschiedliche α_v rufen Fehler in der optimalen Konstruktion von Kerneln hervor. Hätte jedes Neuron x_v ein $\alpha_v = \alpha$ und ein Neuron

x_i stattdessen ein $\tilde{\alpha}_i = \alpha_i + \Delta\alpha$, dann hätte für jede Vergangenheit s, die rekonstruiert werden soll, dieses Neuron durch den veränderten exponentiellen Verfall einen Unterschied von $\Delta x(t) = \alpha^t - \tilde{\alpha}^t$. Man kann $x_v(t)$ zwar an der Stelle s durch das Multiplizieren des Ausgangsgewicht des Neurons mit $\frac{\alpha^s}{\alpha^s - \tilde{\alpha}^s}$ korrigieren, damit würden jedoch auch alle Zeitpunkte vor s mitgestreckt (im Fall für $\Delta\alpha < 0$, der umgekehrte Fall verhällt sich äquivalent) und der Bereich hinter s würde durch den zusätzlichen exponentiellen Zerfall einen Mangel an dieser Frequenz aufweisen. Somit wird die Verwendung dieses Neurons mit grösserem s immer riskanter, da seine Oszillation, die durch die Streckung bei s wieder verwendbar gemacht wurde im Bereich vor s zu exponentiell grossen Fehlern führt.
In Abbildung 17 kann man die $mc(s)$-Ergebnisse des Experimentes mit linearer Regression sehen, bei dem die α_v der verschiedenen Neurone uniform verteilt im Intervall $[0.1, 0.9]$ liegen und die Frequenzen homogen verteilt sind. Man sieht, dass ein Teil der MC hinter $n-1$ liegt und Teile vor n eine $mc(s) < 1$ besitzen. Überall dort, wo die $mc(s)$ nicht 1 ist, ist der $mse(s) > 0$. Trotdem bleibt die aufsummierte $MC = n$, wie man im Beweis 8.4 im Appendix sieht.
Ein weiteres Problem, dass durch eine ungünstige Wahl des Parameters α auftreten kann,

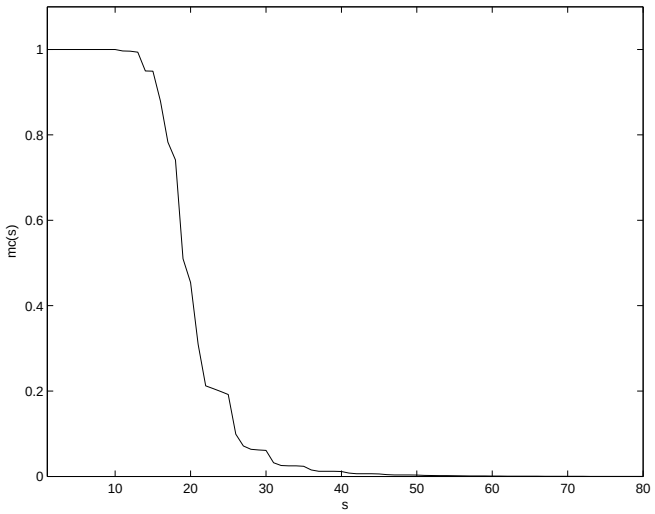

Abbildung 17: $mc(s)$ im VR-Experiment mit $\alpha_v \in [0.1, 0.9]$ mit 20 Neuronen mit homogenen Frequenzen. Im Unterschied zum homogenen Netz mit überall identischem α_v (Abbildung 15) fällt die $mc(s)$ nicht in Form von Treppenstufen ab, sondern kontinuierlich. $MC = 20$

ist Folgedes: Je kleiner α, desto grösser müssen die Ausgangsgewichte für grosse Vergangenheiten s sein. Rechenfehler, die z.B. durch ungenügende Rechengenauigkeit hervorgerufen werden, werden durch diese Ausgangsgewichte ebenfalls mitverstärkt. Für $s \in [0, n-1]$ wäre ein optimales α um 'Geisterbilder' zu eleminieren sicherlich $0 < \alpha \ll 1$ (da 0 durch die Bedingung der Rangvollständigkeit nicht erlaubt ist), jedoch würden in diesem Regime extrem verstärkte Rechenfehler den Fehler durch Geisterbilder um ein Vielfaches übertreffen. Rechenungenauigkeiten führen dazu, dass bei 40 Neuronen bei überall identischem α_v ab $\alpha = 0.5$ die MC stark einbricht (Abbildung 18). Ab einen $\alpha \approx 0.01^{1/n}$ sind die

Geisterbild-Fehler bereits derart gering, dass sie vernachlässigbar werden.

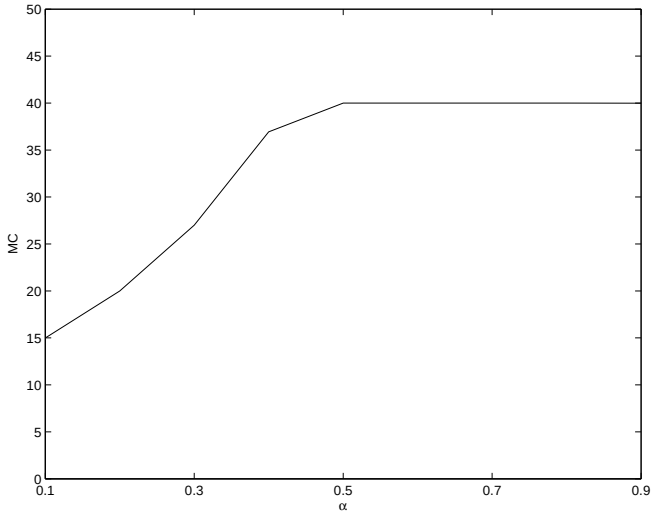

Abbildung 18: MC im VR-Experiment bei variierendem α mit 40 homogenen Neuronen mit einer Rechengenauigkeit von 32 Bit. Da ab $\alpha = 0.5$ in den Vergangenheiten nahe $s = 40$ nur noch 0.5^{40} vom Eingangsimpuls vorhanden bleibt, treten Rechenfehler auf, die durch die Verstärkung durch die Ausgangsgewichte von 0.5^{-40} hochskaliert werden. Je weiter α absinkt, desto stärker ist dieser Effekt und desto mehr sinkt die MC.

4.3.3 Abweichung vom homogenen Netz durch beliebige ω_v

In einem homogenen Netz ist die VR für 0 bis n - 1 mit vernünftig gewähltem α beinahe exakt. Sind die Frequenzen ω_v nicht harmonisch gewählt, dann entstehen Fehler in den erstellten δ-Kernels. Trotz dieser Störung bleibt die $MC = n$ (siehe Beweis 8.4 im Appendix). Man kann in diesen Fällen die Konstruktion des optimalen Kernels nicht mehr analytisch vornehmen, sondern bedarf der linearen Regression. In den Kernels von verschiedenen Vergangenheiten (Abbildung 19) sieht man, dass keine Geisterbilder mehr vorhanden sind, da die Frequenzen sich im Gegensatz zum homogenen Fall nie wiederholt in Phase befinden. Zum gewünschten Peak bei s gesellt sich statt der ehemaligen Wiederholungen der Diracs ein auf der Zeitachse verschmiertes rauschartiges Schwingen, das ähnlich den Geisterbildern durch den exponentielles Verfall langsam abklingt. Dieses Rauschen ist umso stärker und der eigentliche Peak ist umso kleiner, je grösser s ist.
Führt man die VR-Experimente mit einem Netz mit uniform gezogenen Frequenzen aus dem Intervall $[0, 2\pi]$ und identischen $\alpha_v = \alpha$ mit linearer Regression aus und wiederholt dieses Experiment mit verschiedenen α (Abbildung 20), dann sieht man, dass mit $\alpha \to 1$ die $mc(s)$-Kurve immer flacher wird und sich mit immer flacher werdender Schleppe bis ins Unendliche hinzieht. Der Graph fällt somit nicht wie bei homogenen Netzen treppenstufig exponentiell ab, sondern der Abfall verläuft kontinuierlich. Bei $\alpha \to 0$ nähert sich

der Graph dem einer Heavyside-Funktion an, die von 0 bis $n-1$ auf 1 steht und kurz vor n rapide abfällt. In allen Fällen der Experimente ist die $MC = n$.
Da durch ein $\alpha \to 0$ der $mc(s)$-Graph sich immer mehr dem eines homogenen Netzes annähert, sind in diesen Fällen die $mse(s)$ für die Vergangenheiten von 0 bis $n-1$ geringer, da die $mc(s)$ in diesem Gebiet sich 1 nähert. Bei kleinerem α fällt aber auch hier Rauschen stark ins Gewicht, da dies in diesem Fall durch grosse Ausgangsgewichte mitverstärkt wird (siehe Kapitel 4.3.2).

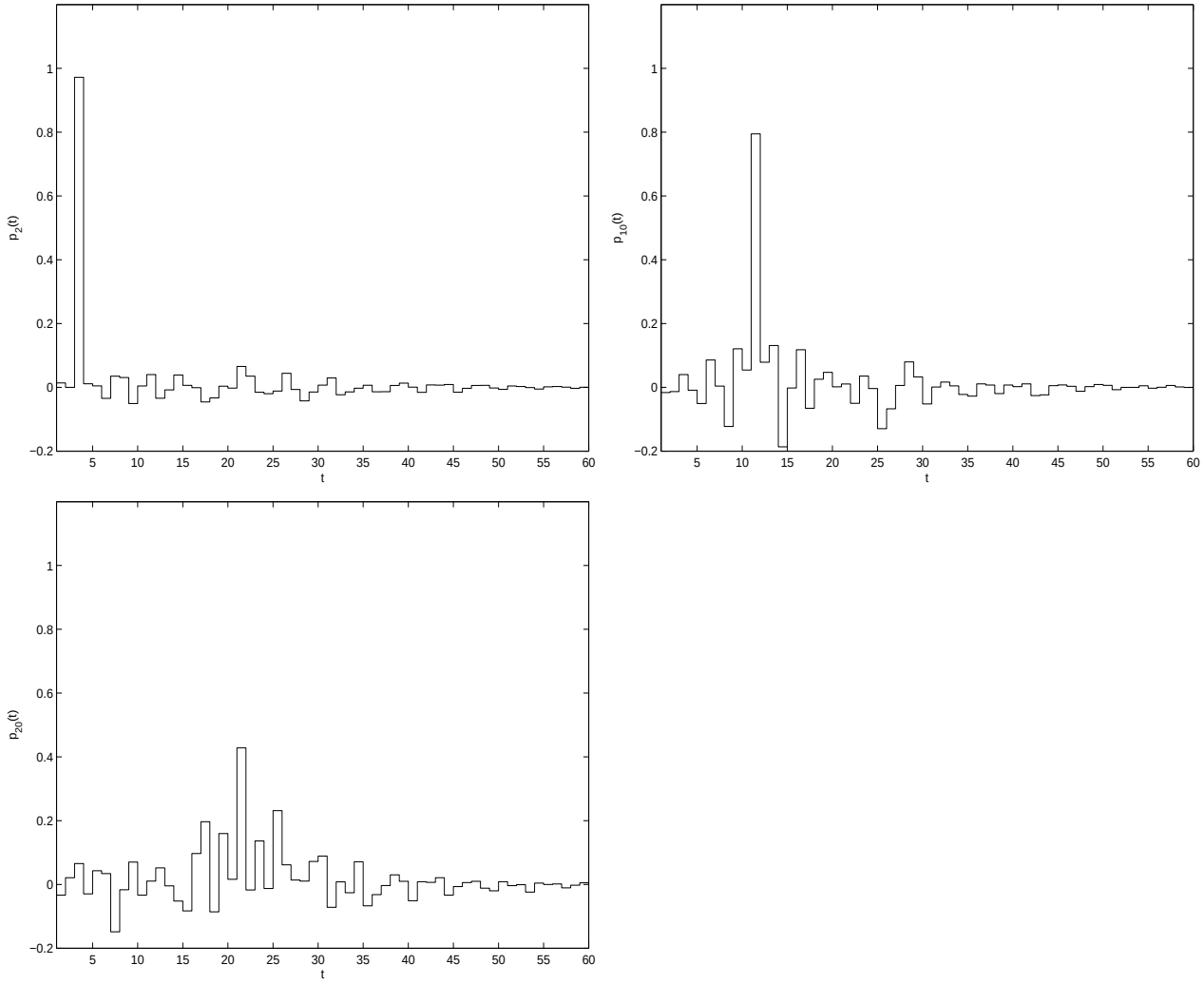

Abbildung 19: Kernel für Vergangenheiten $s = 2$ (l.o), 10 (r.o.) und 20 (l.u.) bei einem Netz aus 20 Neuronen mit $\alpha_v = 0.9$ und zufällig gezogenen ω_v. Wir sehen, dass keine 'Geisterbilder' mehr vorhanden sind wie in Abbildung 16. Dafür liegt ein Rauschen um den Peak bei s. Das Rauschen ist um so stärker und der Peak bei s ist umso kleiner, je grösser s ist.

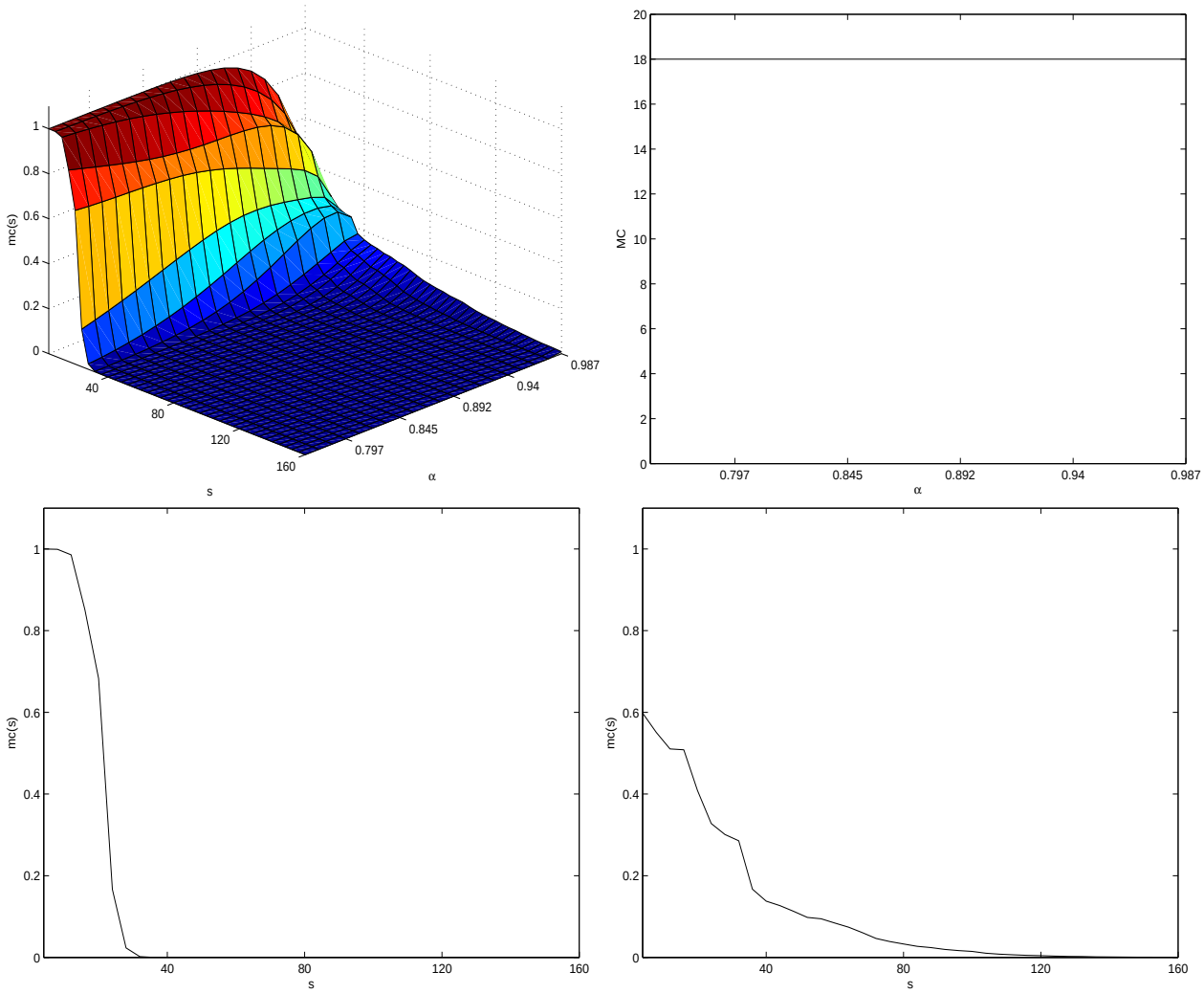

Abbildung 20: $mc(s)$ (l.o) und MC (r.o) mit variierendem α, sowie mc(s) mit $\alpha = 0.75$ (l.u.) und mc(s) mit $\alpha = 0.975$ (r.u.) in einem Netz aus 18 Neuronen mit zufällig gezogenen Frequenzen und identischen $\alpha_v = \alpha$. Vergleicht man die Bilder mit denen aus Abbildung 15, dann sieht man, dass die $mc(s)$ nicht mehr treppenstufig abfällt, sondern kontinuierlich. Trotzem bleibt der aufsummierte $MC = n$. In den unteren beiden Bildern sieht man, dass die $mc(s)$-Kurve länger nahe 1 bleibt und nahe n stark abfällt wenn $\alpha \ll 1$.

4.3.4 Rauschen auf den Erregungszuständen der Echo-Neuronen

Wenn man zu jedem Zeitschritt auf allen Neuronen jeweils ein gaussches Rauschen anlegt, dann führt dies ebenfalls zu einem Rauschen am Ausgang und somit zu einem Fehler in der VR. Dieser Fehler führt im Gegensatz zu den übrigen Fehlerquellen nicht nur zu einem Anstieg des $mse(s)$ sondern auch zu einem Abfall der MC. Dabei stellt sich die Frage, wie stark sich ein kontinuierliches Rauschen mit fixer Varianz auf die MC auswirkt.
Das Rauschen auf den Erregungszuständen der Echo-Neuronen kann durch einen Vektor $\mathbf{z}(t)$ dargestellt werden. $z_i(t)$ hat die Eigenschaft, dass es für jeden Zeitschritt für jedes Neuron unabhängig voneinander normalverteilt gezogen wird, d.h. für jede der Zufallsvariablen $z_i(t) = \xi$ gilt $\mu_\xi = 0$ und $\sigma_\xi^2 = f$. Haben wir eine komplexe Verbindungmatrix A vorliegen, dann soll $Re(f) = Im(f)$ gelten.
Die Iterationsvorschrift der Echo-Schicht (5) wird durch das zusätzliche Rauschen ergänzt.

$\mathbf{z}(t)$ Rauschvektor zum Zeitpunkt t

$$\mathbf{x}(t) = A\,\mathbf{x}(t-1) + B\,\mathbf{u}(t) + \mathbf{z}(t) \tag{40}$$

Durch die Linearitäts-Eigenschaft des Systems können die Auswirkungen des Inputs und des Rauschens unabhängig voneinander untersucht werden. Dies kann bewerkstelligt werden, indem man $\mathbf{u}(t) = 0$ setzt und somit nur noch die Reaktionen des Rauschens verbleiben.

$$\mathbf{x}(t) = A\,\mathbf{x}(t-1) + \mathbf{z}(t) \tag{41}$$

Wir betrachten wieder ein diagonalisiertes System, wie in Kapitel 2.1 vorgestellt. Das Rauschen auf jedem einzelnen Echo-Neuron ist die Faltung von z_v mit dem jeweiligen Einzelkernel g_v (siehe Gleichung (11)).

$$x_v(t) = (z_v * g_v)(t) \tag{42}$$

Die Verteilung der stochastischen Variable $x_v(t)$ ergibt sich durch das Aufsummierung von unterschiedlich stark gewichteten stochastischen Variablen $z_v(t')$ mit $t' \in [-\infty, t]$, unter der Annahme, das das System bereits unendliche lange läuft und somit nicht mehr durch Initialisierungs-Effekte beeinflusst wird. Die Gewichtung zu einem Zeitpunkt t ist für jedes $z_v(t')$ jeweils der Faktor $g_v(t-t')$. Es ergibt sich

μ_{x_v} Mittelwert der Erregung von Neuron v induziert durch konstantes Rauschen
$\sigma^2_{x_v}$ Varianz der Erregung von Neuron v induziert durch konstantes Rauschen

$$\mu_{x_v} = \sum_{t'=-\infty}^{0} g_v(0-t')\,\mu_z = 0 \tag{43}$$

$$\begin{aligned}\sigma^2_{x_v} &= \sum_{t'=-\infty}^{0} |g_v(0-t')|^2\,\sigma_z^2 \\ &= f\sum_{t'=0}^{\infty}\alpha_v^{2\,t'} = \frac{f}{1-\alpha_v^2} \qquad (44)\end{aligned}$$

Die Rauscheffekte jedes einzelnen Echo-Neurons werden über die Ausgangsgewichte **w** im Ausgangs-Neuron aufsummiert. Es ergibt sich am Output-Neuron für eine Vergangenheit von s ein Gesamt-Rauschen.

$\mu_{o,s}$	Mittelwert der Erregung des Output-Neurons für eine Vergangenheit s induziert durch Rauschen
$\sigma^2_{o,s}$	Varianz der Erregung des Output-Neurons für eine Vergangenheit s induziert durch Rauschen

$$\mu_{o,s} = \sum_{v=1}^{n} w_v\,\mu_{x,v} = 0 \qquad (45)$$

$$\sigma^2_{o,s} = \sum_{v=1}^{n} |w_v|^2\,\sigma^2_{x_v} \qquad (46)$$

In einem homogenen Netz gilt mit (31)

$$\sigma^2_{o,s} = \sum_{v=1}^{n}\frac{f\,q_s^2}{1-\alpha^2} = \frac{f\,q_s^2\,n}{1-\alpha^2} \qquad (47)$$

Das zusätzliche Rauschen wirkt sich bei der Berechnung der $mc(s)$ nicht auf die Kovarianz aus, sondern verändert nur die Varianz des Ausgangs, die im Term in Gleichung (28) im Nenner steht. Mit (47) eingebracht in (28) gelangt man zu

$$\begin{aligned}mc(s) &= \frac{(p_s(s))^2}{(\sum_{t=0}^{\infty} p_s(t)^2) + \sigma^2_{o,s}} \\ &= \frac{q_s^2\,n^2\,\alpha^{2\,s}}{q_s^2\,n^2\,\alpha^{2\,mod(s,n)}\,\sum_{k=0}^{\infty}\alpha^{2\,n\,k} + \frac{f\,q_s^2\,n}{1-\alpha^2}} \\ &= \frac{\alpha^{2\,s}}{\frac{\alpha^{2\,mod(s,n)}}{1-\alpha^{2\,n}} + \frac{f}{n\,(1-\alpha^2)}} \qquad (48)\end{aligned}$$

In Abbildung 21 sieht man den MC-Graphen einiges homogenen Netzes mit Rauschen.

4.3.5 Vergleich mit Jägernetzen

Im Model der Jägernetze hat man die bisher beschriebenen Störquellen kombiniert. Da die Eigenwerte einer gauss'sch zufällig belegten Matrix uniform im komplexen Kreis mit

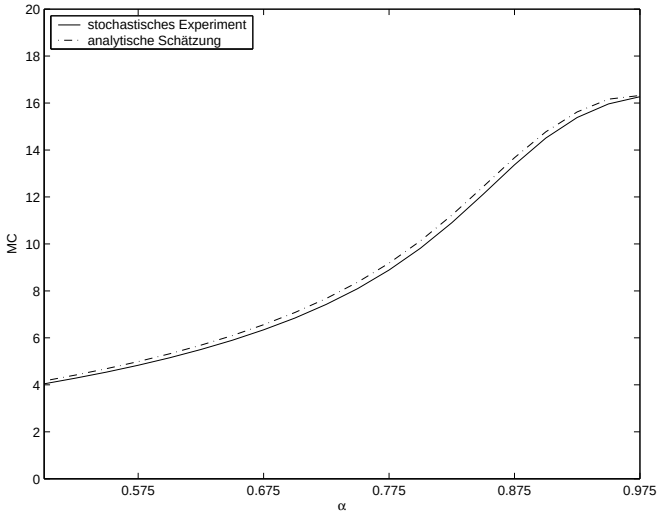

Abbildung 21: MC für verschiedene Verfallsfaktoren α mit Fehlervarianzen $f = \sqrt{0.05}$ auf jedem Neuron bei einem homogenen Netz aus 18 Neuronen. Die durchgezogene Linie wurde erstellt durch Speisen des Netzes mit weissem Rauschen, nach 5000 Trainingsschritten Durchführen der Regression für die Ausgangsgewichte und anschliessendem Test mit weissem Rauschen. Die gestrichelte Linie wurde erstellt durch das analytische Errechnen der MC durch Gleichung (48). Man sieht, dass bei geringerem α die MC sinkt, im Gegensatz zu Abbildung 15, wo sie stets konstant bleibt. Die Ursache liegt darin, dass bei geringerem α die Ausgangsgewichte grösser sind und dadurch das Rauschen auf den einzelnen Einheiten ebenfalls mitverstärkt wird, wodurch die MC sinkt.

Radius $\sqrt{n}$ verteilt sind, steigt die Dichte der α mit dem Radius quadratisch an und die Dichte der Frequenzen ist für jeden Frequenzwinkelabschnitt konstant (siehe Appendix 8.5 'Girko's circular law'). Bei diesen Netzen sind die α_v alle unterschiedlich, daher treten die im Kapitel 4.3.2 genannten Störungen durch unterschiedliche α_v auf. Weiterhin sieht man die Effekte der inhomogen verteilten Frequenzen, was man daran sieht, dass der $mc(s)$ eine lange Schleppe für $s > n$ aufweist.

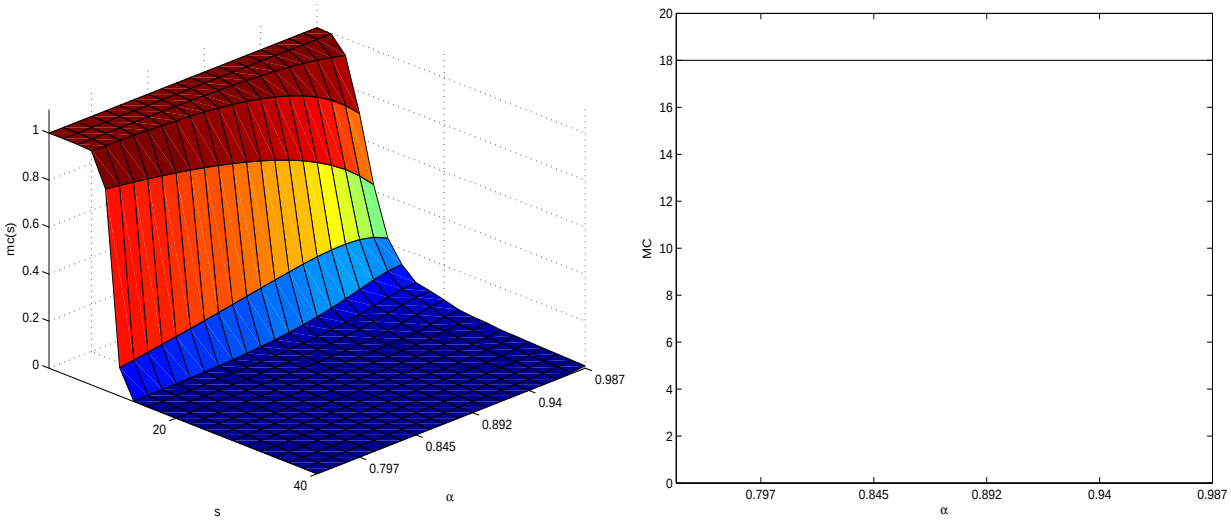

Abbildung 22: $mc(s)$ und MC für variierendes α in einem Jägernetz mit 18 Neuronen. Wir sehen ähnlich zu Abbildung 20, dass die $mc(s)$ kontinuierlich abfällt. Genauso wie in Abbildung 15 und 20 ist auch hier der MC unabhängig vom α konstant gleich n.

Im Appendix befindet sich der Beweis für $MC = n$ für allgemeine ESNs, d.h. für ESNs mit unterschiedlichen, beliebigen α_v und ω_v.

4.3.6 Störungen durch Nicht-Linearitäten

Eine Behandlung von Störungen durch nichtlineare Effekte möchte ich erst nach Klärung der Eigenschaften der Nicht-Linearitaten behandeln. Daher erscheint dieses Unterkapitel erst im Kapitel 5 'Untersuchungen von ESNs im Nicht-Linearen'.

5 Untersuchung von ESNs im Nicht-Linearen

5.1 Transfer-Funktionen

Bevor ich mich mit den Netzen selbst befasse möchte ich einige Vorraussetzungen an die Transfer-Funktion klären. Damit ESNs im Nicht-Linearen funktionieren, muss die 'fading memory'-Eigenschaft gewährleistet bleiben. Man muss sicherstellen, dass die Norm des Zustandsvektors nach einem Zeitschritt kleiner wird oder gleich bleibt. Daher darf man als Transfer-Funktion nur solche Funktionen in Betracht ziehen, für die gilt

$$|f(x)| < |x| \quad , \tag{49}$$

bzw. für die diese Ungleichung im Wertebereich des Netzes gilt.

5.1.1 $\tanh(x)$ als Transfer-Funktion

Der $\tanh(x)$ wird sehr gerne als Transfer-Funktion verwendet, da sein Wertebereich im Intervall zwischen 0 und 1 liegt, und er überall differenzierbar ist. Im Komplexen ist $\tanh(x)$ jedoch nicht überall kontrahierend, weswegen er im Komplexen nicht verwendet werden darf. Die Funktion $f(x) = \tanh(Re(x)) + i \tanh(Im(x))$ erfüllt dagegen die gewünschten Eigenschaften. Um Probleme mit der Transfer-Funktion $\tanh(x)$ zu vermeiden kann man ein komplexes Netz mit dem im Kapitel 8.3 'Ersatzschaltbilder' beschriebenen Verfahren in ein reelles Netz umwandeln, in dem die divergierenden Teile der Transfer-Funktion nicht mehr verwendet werden. Dabei muss man beachten, dass das sich ergebende Bandmatrix-Netz nach derartigen Umformungen im Nicht-Linearen nicht äquivalent zum diagonalen Netz ist. Da die Wahl der nichtlinearen Funktion aber eine relativ willkürliche Entscheidung darstellt, da man einfach nur mit irgendeiner Möglichkeit den zu separierenden Funktionsraum krümmen möchte, sind auch die durch die Verdrehung ins Reelle entstehenden Fehler zu tolerieren (In meinen Experimenten habe ich keine Qualitätsunterschiede zwischen komplexen Matrizen und ihren reellen Korrelaten feststellen können).

5.1.2 Alternative Transfer-Funktion durch Taylorentwicklung von $\tanh(x)$

$$\begin{aligned} \gamma(x) &= \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \\ \gamma'(x) &= 1 - \gamma(x)^2 \\ \gamma''(x) &= -2 * \gamma(x) * (1 - \gamma(x)^2) \\ \gamma'''(x) &= -2 + 8\,\gamma(x)^2 - 6\,\gamma(x)^4 \\ \Rightarrow \quad \gamma^*(x) &= x - \frac{x^3}{3} \end{aligned} \tag{50}$$

Für die Taylorapproximierung des $\tanh(x)$ gilt dasselbe, wie für die Funktion selbst auch: Im Komplexen muss man die Ersatzfunktion separat auf Real- wie Imaginärteil anwenden, um 'fading memory' weiterhin zu gewährleisten. Durch Ungleichung (51) ist sichergestellt, dass die alternative Transfer-Funktion die 'fading memory'-Eigenschaft nicht verletzt, solange sich die Erregungs-Werte im Netz in einem gewissen Rahmen bewegen. Die Funktion ist in diesem Bereich kontrahierend und wenn Matrix A immer noch einen Spektralradius $\rho_A < 1$ besitzt, dann wird jeder Eingangs-Impuls, der derart klein skaliert ist, dass der kontrahierende Bereich der Transfer-Funktion nicht verlassen wird, mit der Zeit stetig abfallen.

$$\begin{aligned} |x - \frac{x^3}{3}| &< |x| \\ \Rightarrow \quad x^2 &< 6 \end{aligned} \tag{51}$$

5.1.3 Entladungsfunktion als Transfer-Funktion

Eine andere denkbare Transfer-Funktion ist

$$\gamma(x) = \begin{cases} x & \text{für } x < b \\ 0 & \text{für } x >= b \end{cases} \tag{52}$$

Diese Entladungs-Transfer-Funktion führt dazu, das die Neuronen ab einer gewissen Schwelle b sich auf 0 entladen. Sie verhalten sich somit ein wenig wie Leaky-Integrate-and-Fire-Neuronen, die beim Feuern keinen Impuls aussenden. Auch diese Transfer-Funktion wirkt sich auf die Experimente mit nichtlinearen Aufgaben, wie z.B. Mustererkennung, vorteilhaft aus (Abbildung 23).

5.2 Topologien von Netzen

Wenden wir uns nun den nichtlinearen Netzen selbst zu. Diese sind im Allgemeinen schwierig analytisch zu untersuchen, da ihre Operationen selten algebraisch aufgelösst werden können. Ich suche daher nach einer Methode, die zufällig belegten, spärlich vernetzten, nichtlinearen ESNs (Jägernetze) zu abstrahieren. Ein Ansatz hierzu ist das Netz wie im linearen Ansatz zu diagonalisieren. Im Gegensatz zum linearen Ansatz ist das diagonalisierte Netz nicht äquivalent zum ursprünglichen Netz. Die Belegung der Verbindungsmatrix hat im nichtlinearen Fall sehr wohl Einfluss auf die MC und nicht nur alleinig das Eigenwertspektrum. Um zu testen, wie nachteilhaft derartige Verallgemeinerungen sind, führe ich mit unterschiedlichen Topologien Experimente durch.
Ich möchte nun verschiedene Netztopologien definieren, mit denen ich MC-Experimente durchführe:

- a-Netz: ein homogenes Netz, das durch Verdrehungen in eine reelle Bandmatrix umgewandelt wird (siehe Appendix 8.3 'Ersatzschaltbilder').

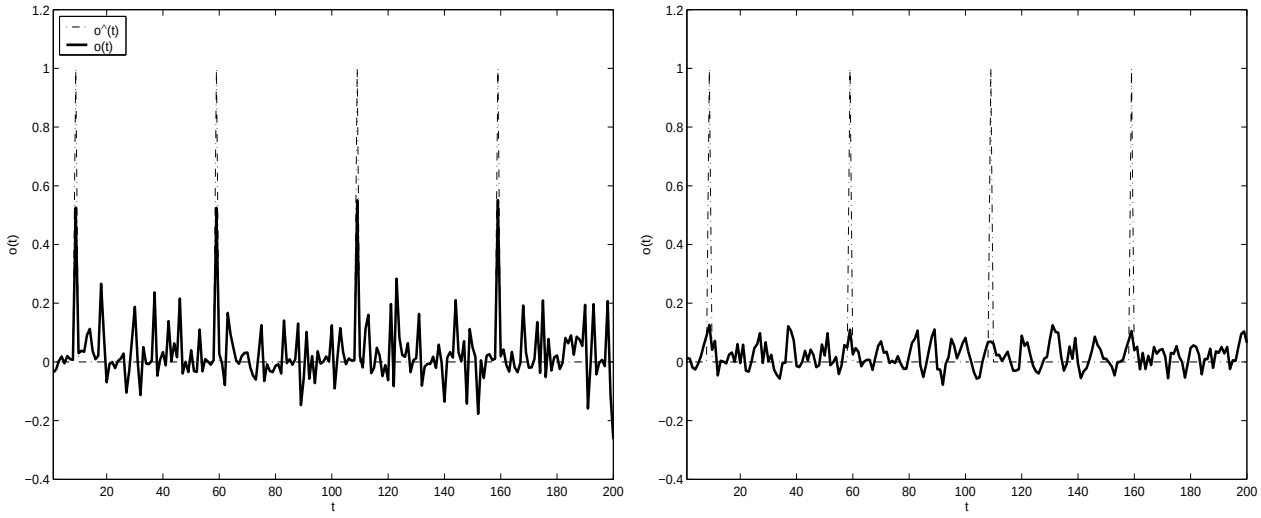

Abbildung 23: Mustererkennungs-Ergebnisse des gleichen Netzes einmal mit Entladungs-Transfer-Funktion (links) und einmal ohne Transfer-Funktion (rechts). Man sieht, dass die Transfer-Funktion grosse Vorteile bei der Mustererkennung bietet.

- b-Netz: das a-Netz, nur dass die Matrix A noch zusätzlich mit einer Drehungsmatrix V mit $|V| = 1$ in die Matrix $\hat{A} = V^T A V$ transformiert wurde, so dass alle Einträge der Matrix belegt sind, aber das Eigenwertspektrum erhalten bleibt.

- c-Netz: ein Netz mit einer Diagonalmatrix A, deren Eigenwerte alle denselben Betrag besitzen, deren Winkel jedoch zufällig aus dem Intervall $[0, 2\pi[$ gleichverteilt gezogen wurden. Diese Matrix wird ebenfalls in eine reelle Bandmatrix umgewandelt.

- d-Netz: ein Netz in dessen Matrix A in jedes Feld eine normalverteilte Zufallszahl mit $\mu = 0$ und $\sigma^2 = 1$ eingetragen wurde und das dann auf einen Spektralradius von 1 normiert wurde (Jägernetz).

- e-Netz: ein Netz, dass aus 3 homogenen ESN-Paketen aufgebaut ist, die jeweils aus einem Drittel der Neuronen der anderen Netze bestehen. Diese einzelnen kleinen Netze werden mit den Eingangs-Signalen $x(t)$, $x(t)\,x(t-1)$ und $x(t)\,x(t-1)\,x(t-2)$ gespeist.

Der Sinn der verschiedenen Topologien wird klarer, wenn ich einige Eigenschaften der verschiedenen Topologien erläutere:
Das d-Netz ist ein Jägernetz und dient zum Vergleich der anfangs willkürlichen gemachten Konstruktionsvorschrift mit den abstrahierteren Varianten.
a- und b-Netz haben dasselbe Eigenwertspektrum und verhalten sich somit im Linearen identisch, das eine ist jedoch noch zusätzlich durch eine Basistransformation verdreht, so

dass alle Einträge seiner Verbindungsmatrix ungleich 0 sind, was auch beim d-Netz der Fall ist. Man kann damit beobachten, wie viel Einfluss eine dichtere Verbindungsstruktur auf nichtlineare Verrechnungen besitzt.
Alle Netze ausser dem d-Netz haben die Eigenschaft, dass die Beträge ihres Eigenwertspektrum alle gleich sind ($\alpha_v = \alpha, \quad v \in [1, n]$). Im c-Netz sind jedoch die Frequenzen in einer Art verteilt, wie es auch im d-Netz der Fall ist (siehe Appendix 8.5 'Girko's circular law'). Man kann in den Unterschieden zwischen d-Netz und den übrigen Netzen sehen, welchen Einfluss Nicht-Linearitäten auf Netze mit unterschiedlichen α_v haben.
Das e-Netz ist dazu gedacht sehen zu können, wie viel der Fähigkeiten nichtlinearer Netze auch im Linearen erreicht werden können, wenn man nichtlineare korrelierte Eingänge einspeist.
Die Unterschiede im Verhalten von Netz a bis e lassen Rückschlüsse zu, welchen Einfluss Nicht-Linearitäten auf ESNs haben und welche Einflüsse anderen Quellen entstammen.

Die Auswirkungen von Nicht-Linearitäten hängen direkt von der Wahl der Transfer-Funktion ab. Ich möchte mich bei den folgenden Experimenten auf den $\tanh(x)$ als Transfer-Funktion beschränken.
Anders als in den linearen Experimenten ist nun die Skalierung des Eingangs von Bedeutung. In Gleichung (50) sieht man, dass sich der $\tanh(x)$ für $x \to 0$ linear verhält. Daher ist zu vermuten, dass die Ergebnisse in diesem Regime sich denen der linearen Experimente annähern.

5.3 Störung der MC durch Nicht-Linearitäten

In den folgenden Testreihen habe ich Netze aus 18 Neuronen auf ihre MC-Eigenschaft mit variierendem α (d.h. die Verbindungsmatrizen der im Kapitel 5.2 definierten Netze werden nochmals mit diesem Faktor multipliziert) und variierender Inputskalierung getestet. Im Linearen sollten alle Netze einen $MC = 18$ haben. Da die Experimente jedoch nur mit einer endlichen Anzahl an Trainingsschritten und mit einer endlichen Anzahl an Testschritten ausgeführt wurden, weichen die Ergebnisse leicht von den erwarteten theoretischen Werten ab. Daher zeige ich in den Bildern ebenfalls die Ergebnisse der linearen Experimente, um vergleichen zu können, wie stark sich die Nicht-Linearitäten auswirken.

In Abbildung 24 sieht man den MC der nichtlinearen Netze im Hinblick auf verschiedene α verglichen mit demselben Experiment mit linearer Transferfunktion. Getestet wurden 40 verschiedene α gleichmässig verteilt von 0.5 bis 0.975. Man sieht, dass die Nicht-Linearität die MC für niedrige α reduziert. a-, b- und c-Netz unterscheidet sich im Nicht-Linearen kaum voneinander. Das d-Netz ist stärker beeinträchtigt, da es mehrere niedrige α_v besitzt, die durch die Nicht-Linearität stärker beeinträchtigt werden, als die überall identischen, grösseren α_v in den übrigen Netzen.

In Abbildung 25 teste ich die MC der nichtlinearen Netze im Hinblick auf verschiedene Skalierungen des Eingangs, wieder im Vergleich mit demselben Experiment mit linearer Transferfunktion. Getestet wurden 40 verschiedene Skalierungen von $0.8^{-9} \approx 7.45$ bis $0.8^{30} \approx 0.001$. Die Netze zeigen alle ein ähnliches Verhalten. Je grösser die Input-Skalierung, desto kleiner die MC. Ab einer Skalierung von circa 0.09 tritt eine Sättigung ein. a-, b- und c-Netz sehen wieder sehr ähnlich aus, das d-Netz erreicht seine Sättigung erst um einige Grössenordnungen später.

Man konnte bisher sehen, wie sich bei den verschiedenen Netzen unterschiedlichen Spektralradien und Eingangsskalierungen auf ihre MC auswirken. In den folgenden Experimenten ist ein Teil der Aufgabe die Vergangenheit zu rekonstruieren und ein weiterer Teil diese Vergangenheit nichtlinear zu verrechnen. Wenn die MC durch einen zu grossen Eingang bereits sehr abgesunken ist, kann man natürlich keine grossen Fähigkeiten in der Verrechnung der schlecht rekonstruierten Vergangenheits-Werte erwarten.

5.4 Nichtlineare Aufgaben

Um nichtlineare Aufgaben lösen zu können, benötigt man nichtlineare Funktionen. Wie stark die Nicht-Linearitäten in den verschiedenen Netzwerk-Topologien ausgenutzt werden können soll nun untersucht werden. In a-Netzen könnte man beanstanden, dass in ihnen die verschiedenen Einheiten untereinander keine Information austauschen können. Sie verlieren ohne interne Vernetzung einen Teil der Verrechnungsstärke, jedoch bringt eine Nicht-Linearität auch in diagonalen Netzen grosse Vorteile. Dies kann veranschaulicht werden, wenn man sich die Verarbeitung eines Signals über zwei Zeitschritte mit der Taylorapproximierung der Transferfunktion $\tanh(x)$ und ein Neuron mit rekurrentem Gewicht d_v betrachtet.

$$\begin{aligned} x_v(u(t), u(t-1)) &= \gamma^*(\gamma^*(u(t-1)) + u(t)) \\ &= u(t) - \frac{1}{3}\,u(t)^3 + \\ &\quad d_v\,(u(t-1) - \frac{1}{3}u(t)^3 - u(t-1)\,u(t)^2 + \frac{1}{3}u(t-1)^3\,u(t)^2) + \\ &\quad d_v^2\,(u(t-1)^2\,u(t) + \frac{2}{3}u(t-1)^4\,u(t) - \frac{1}{9}u(t-1)^6\,u(t)) + \\ &\quad d_v^3\,(\frac{1}{3}u(t-1)^5 - \frac{1}{9}u(t-1)^7 + \frac{1}{81}u(t-1)^9) \end{aligned} \tag{53}$$

Man kann sehen, dass in den Erregungszuständen des Neurons nicht mehr nur die linearen Anteile der vergangenen Inputs $u(t)$ und $u(t-1)$ vorhanden sind, sondern auch Korrelationen dieser mit höheren Potenzen. Durch diese reichhaltige Vielfalt an Korrelationen hoher Potenzen von $u(t)$ und $u(t-1)$ kann man mit mehreren Neuronen mit unterschiedlichem d_v und geeigneten Ausgangsgewichten eine grosse Menge an nichtlinearen Funktionen approximieren.

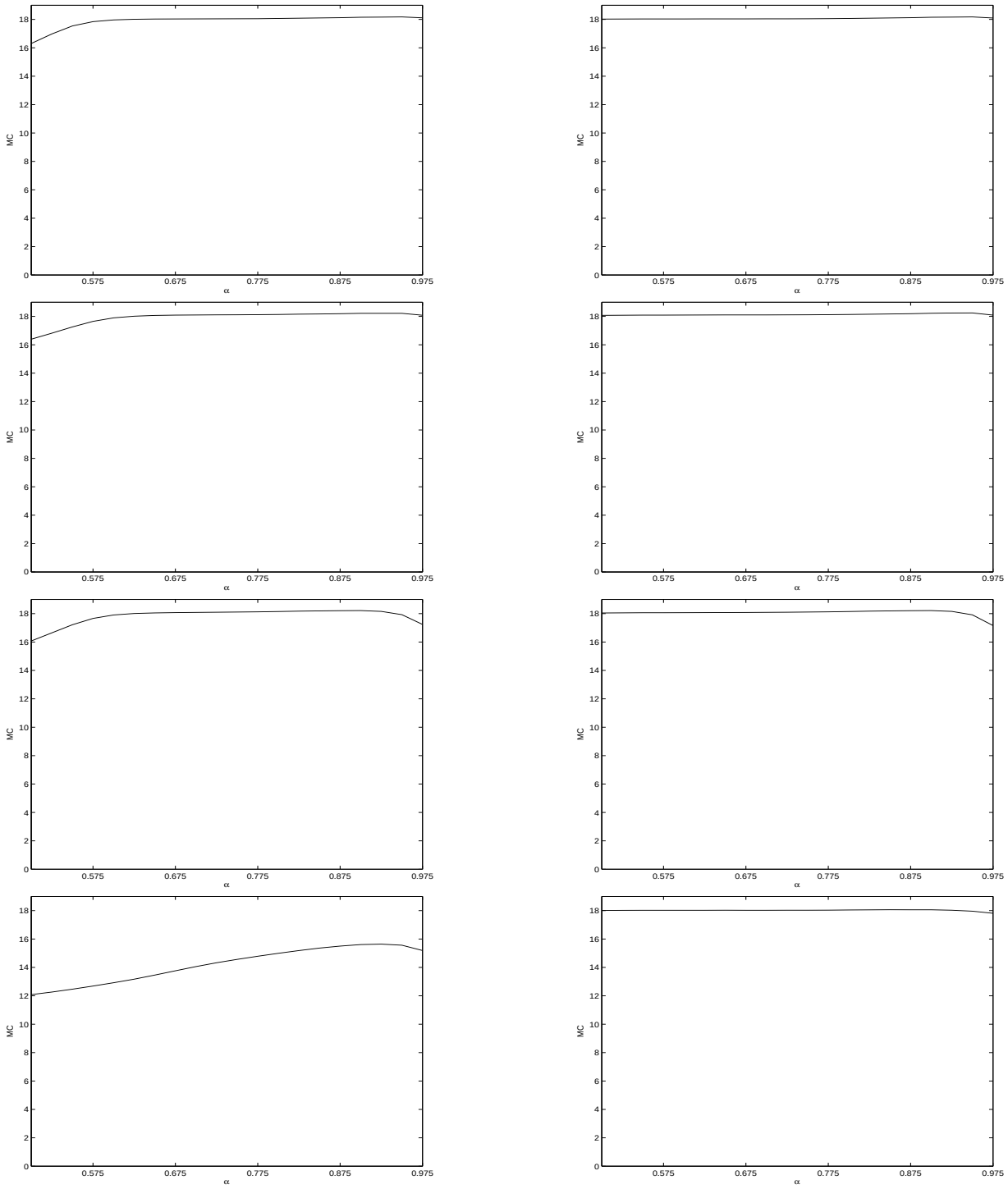

Abbildung 24: MC für verschiedene Verfallsfaktoren α mit Input-Skalierung 0.01 bei 18 Neuronen. links nichtlinear, rechts linear; von oben nach unten: a-Netz, b-Netz, c-Netz, d-Netz. Man sieht, dass im Nicht-Linearen nur bei einem niedrigem α die MC reduziert wird. Das d-Netz erfährt auch bei hohen Werten für α eine Reduzierung siner MC. Dies liegt vermutlich daran, dass es viele Eigenwerte besitzt, die sehr klein sind.

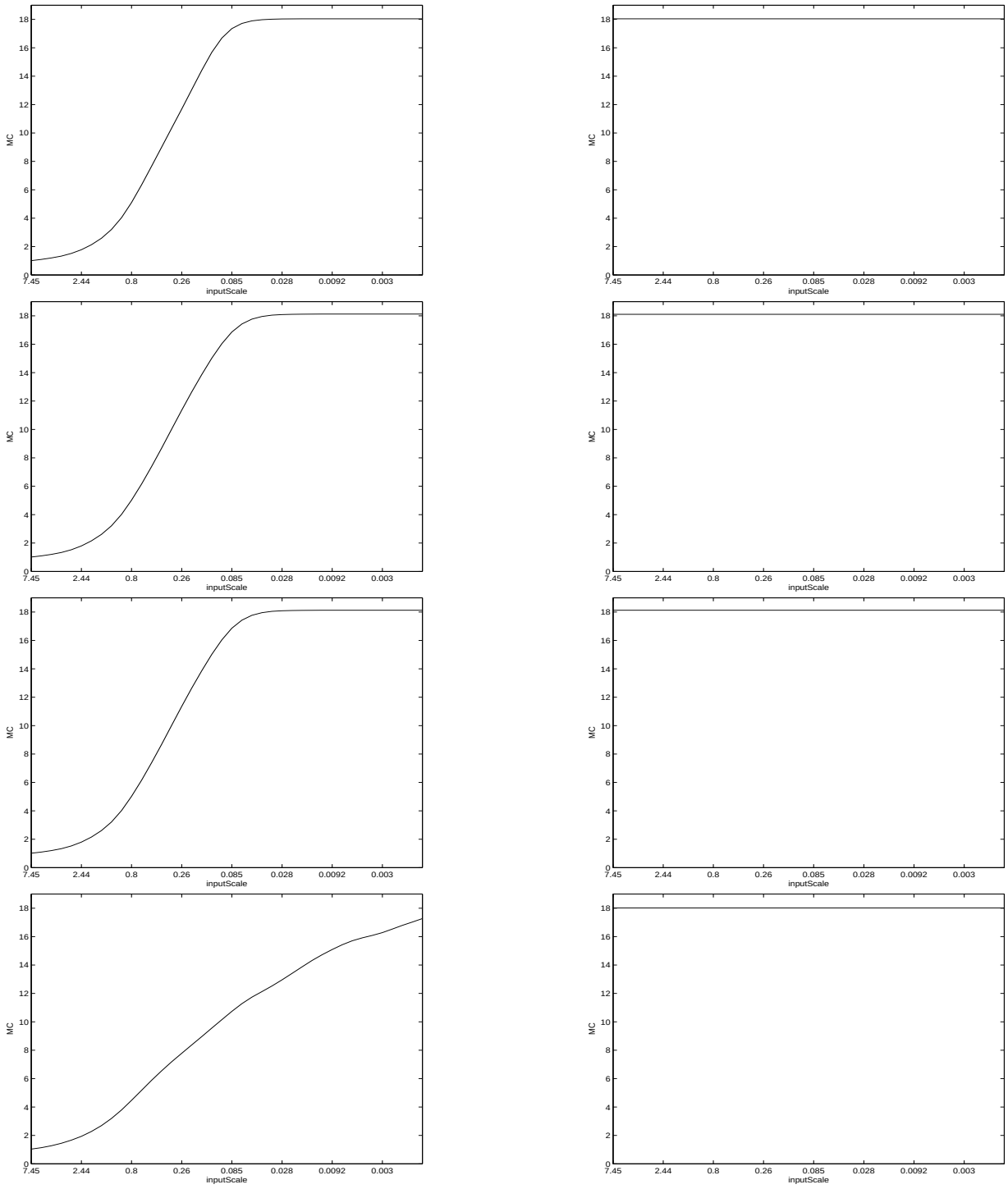

Abbildung 25: MC für verschiedene Input-Skalierungen mit $\alpha = 0.8$ bei 18 Neuronen. links nichtlinear, rechts linear; von oben nach unten: a-Netz, b-Netz, c-Netz, d-Netz. a- bis c-Netz ähneln sich stark. Die MC ist bei grossen Eingangs-Skalierungen im Bereich $[10^{-1}, 10^{1}]$ stark reduziert, nimmt jedoch bei kleinerer Skalierung wieder zu. Ab einer Skalierung von 0.09 haben a- bis c-Netz wieder volständige MC von $n = 18$. Beim d-Netz findet diese Sättingung erst viel später bei einer Input-Skalierung von kleiner 10^{-3} statt. Dieser Effekt stammt vermutlich von zu vielen Eigenwerten, die ein geringes α_v besitzen.

Ein Beispiel hierfür ist, dass man mit einem nichtlinearen Netz die XOR-Funktion berechnen kann, die ein gutes Beispiel für nichtlineare Verrechnungsstärken darstellt, da sie linear nicht separierbar ist. Wählt man 5 Neuronen mit den rekurrenten Gewichten $\mathbf{d} = [1, -1, i, -i, 0]$ (bzw. im reellen Ersatzschaltbild in Abbildung 26) $a_1 = 1$, $a_2 = -1$, $a_3 = 1$, $a_4 = -1$), den Skalierungsfaktor $\rho = 0.01$ (Spektralradius) damit alle Eingänge jenseits von $u(t-1)$ stark gedämpft werden und Eingangs-Gewichte von überall 1, dann kann man die Ausgangsgewichte derart wählen, dass die XOR-Berechnung fast exakt ist. In der gezeigten Verschaltung ergab sich durch lineare Regression die Wahl von $\mathbf{w} = 10^3\,[0.5663, 0.5588, 0.5663, 0.5588, -2.2501]$ und dadurch ein $mse = 2.5e-9$.

Ein ebenso mächtiges Netz, das linear arbeitet und das beschriebene Verfahren der zusätz-

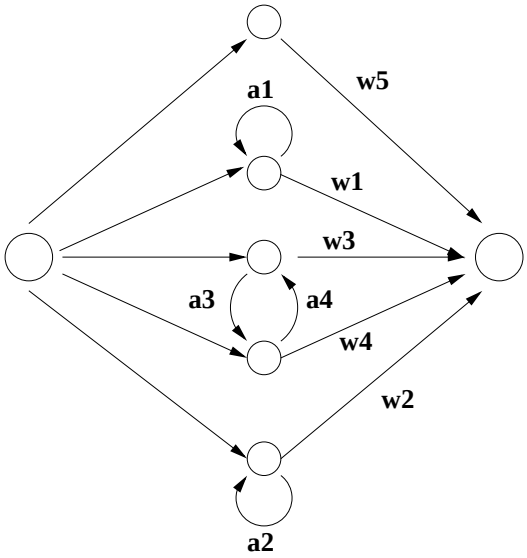

Abbildung 26: Konstruktion eines nicht-linearen Netzes, das den XOR der beiden Vergangenheiten berechnet

lichen korrelierten Eingänge verwendet, ist ein e-Netz, welches man in Abbildung 27 sehen kann. Dieses hat sogar einen exakten mse von 0.

Das Angebot an eingespeisten Korrelationen muss in dieser Topologie jedoch wohl gewählt

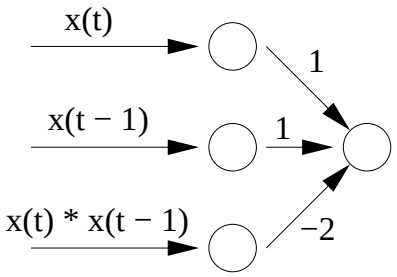

Abbildung 27: Konstruktion eines linearen Netzes, das den XOR der beiden Vergangenheiten berechnet

sein. Im Mustererkennungs-Experiment musste man den quadrierten Eingang mit einspeisen, in Fall des XORs muss es die Korrelation der letzten beiden Eingänge sein. Diese explizite Wahl muss man in allgemeinen ESNs nicht treffen. Diese suchen sich aus dem vorliegenden Pool aus mannigfaltigen Korrelationen diejenigen heraus, die sich in der linearen Regression am günstigsten erweisen.

In Gleichung (50) sah man die Taylor-Approximationen 3. Grades der Transfer-Funktion $\tanh(x)$ entwickelt um 0. Man sieht, dass sich die Funktion in der Nähe 0 linear verhält und in der etwas entfernteren Nähe um 0 kubisch mit linearem Anteil.
Bei der Variierung der Parameter des VR-Experiments kann man beobachten, dass mit kleiner skalierten Eingängen die Ergebnisqualitäten zunehmen. Dies erklärt sich dadurch, dass, wenn sich die Funktionswerte der Transfer-Funktion im Bereich um 0 bewegen, sich die Funktion linear verhält und das Netz die Eigenschaft gewinnt exakt seine Vergangenheit zu rekonstruieren, wie es die linearen Netze in der Lage sind (siehe Kapitel 3.2). Jedoch verliert es dann zunehmend seine Fähigkeit Korrelationen zwischen den verschiedenen Vergangenheiten zu errechnen, wie man es z.B. in der Mustererkennung benötigt, da die Beiträge höherer Potenzen der Erregungs-Funktion verschwinden.
Nichtlineare Netze verlieren also ihre Verrechnungsstärke aber gewinnen an MC, wenn man ihre Eingänge kleiner skaliert. Nicht-Linearitäten fördern die Fähigkeit zur nichtlinearen Separierung von Eingangssignalen, gleichzeitig verwischen sie das Bestehen des Eingangssignale über grössere Zeiträume in die Vergangenheit. Wenn man die Eingänge kleiner skaliert scheinen die Fähigkeit von ESNs zur nichtlinearen Separierung der Eingangssignale nicht im selben Masse abzunehmen, wie das Netz an MC gewinnt.

Weiterhin darf man bei den Untersuchungen folgenden Gesichtspunkt nicht unbeachtet lassen: lineare Netze sind durchaus in der Lage nichtlineare Funktionen mit einer gewissen Fehlergüte zu approximieren. In Abbildung 28 sieht man die Ergebnisse eines Netzes, dass die Multiplikation zweier Zahlen im Intervall [0, 1] erlernen sollte. Im linken Bild könnte man sich durch die augenscheinlich gute Fehlerrate verleiten lassen zu denken, dass das Netz nichtlineare Verrechnungen durchführt. Im rechten Bild, in dem man dasselbe trainierte Netz nur auf binären Eingangssignalen ausgetestet hat, sieht man jedoch, dass die vermeintliche Multiplikation nur auf Additionen der Eingänge basiert. Um nicht-lineare Verrechnungs-Eigenschaften zu beurteilen muss man sich dieses Effektes bewusst sein.

Ein weiterer Aspekt, der bei der Betrachtung der Ergebnisse zu beachten sei, ist, dass ein besseres Ergebnis in einem der Experimente gegenüber einem anderen Netz im selben Experiment nur indirekt eine Aussage über die allgemeinen Verrechnungseigenschaften des Netzes liefert. Die Qualitäten unterschiedlicher Topologien kann man nur problematisch vergleichen. Der Erfolg in der Berechnung einer bestimmten Funktion hängt immer sehr vom Aufbau des verwendeten Netzes ab. So können z.B. Netze mit rein linearen Eingängen ihre komplette Echo-Schicht für die Rekonstruktion der Vergangenheit heranziehen. Net-

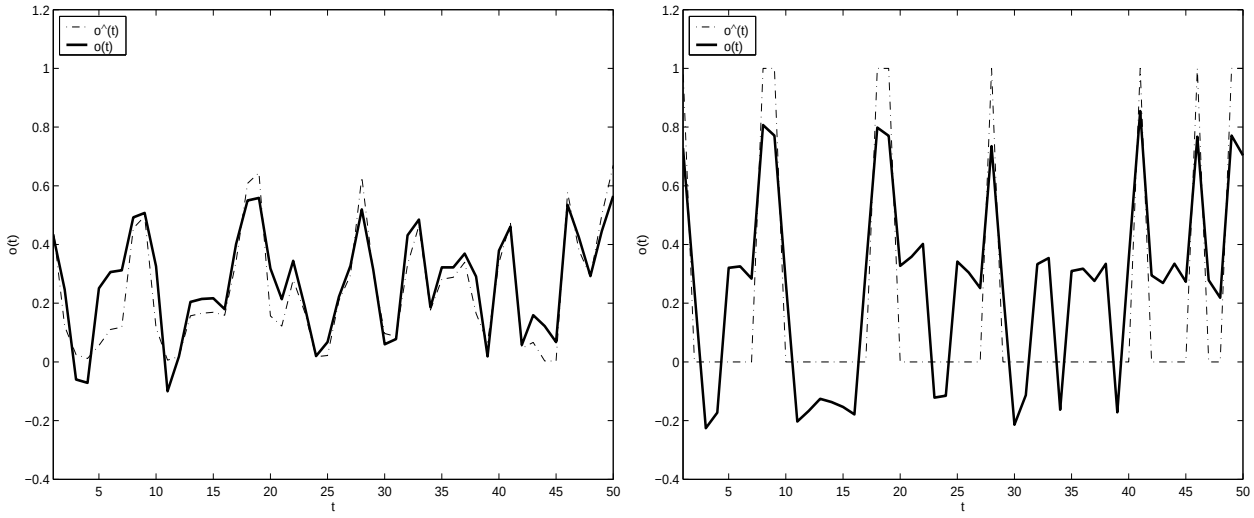

Abbildung 28: Ergebnisse eines linearen Netzes, dass die nicht-linear Funktion $f(a, b) = a\,b$ erlernen sollte, links Eingänge im Intervall [0, 1], rechts nur binäre Eingänge (0, 1)

ze, die die Menge an Echo-Neuronen in mehrere Pools aufteilen, die die Vergangenheiten von unterschiedlichen Korrelationen speichern, können diese in ihren Berechnungen ausschöpfen, jedoch nur auf einen entsprechenden Teil der Vergangenheit dabei zurückgreifen. Wenn man z.B. die beiden unterschiedlichen Netz-Topologien 'nichtlineares Netz' und 'lineares Netz mit nichtlinearen Eingängen' dem Experiment unterzieht, dass ein 20 Zeiteinheiten langes Muster erkannt werden soll, die Erkennung dessen jedoch erst 10 Zeiteinheiten später gemeldet werden soll, dann schneiden die Netze mit mehren Pools, die mit korrelierten Eingängen gespeisst werden, verständlicherweise besser ab, da diese die Vergangenheit der Korrelationen bis zu 20 Zeitschritte exakt speichern können und somit mit dieser Aufgabe keinerlei Probleme bekommen. Bei nichtlinearen Netzen sieht man bereits in der Analyse ihrer Qualitäten in VR-Experimenten, dass ihre MC stark eingeschränkt ist. Sie liefern daher auch in Mustererkennungs-Aufgaben mit verzögerter Reaktion schlechte Ergebnisse, da das Muster zwar im Prinzip relativ gut erkannt werden könnte, jedoch nicht in dieser Vergangenheits-Tiefe. Durch Verwenden von übermässig vielen Echo-Neuronen können auch nichtlineare Netze dazu befähigt werden die oben beschriebene Aufgabe zu erfüllen, da sich, bei ausreichender Grösse des zufällig angelegten Echo-Pools immer Verschaltungen ergeben, die sowohl eine ausreichende MC besitzen, als auch genügend korrelierte Verrechnungen ausführen können, damit die Erkennung des Musters gewährleistet ist.
Genau genommen sagt ein guter Fehlerwert eines Netzes in einem Experiment vorerst nur, dass das Netz im Vergleich mit einem anderen Netz, das bei dieser Aufgabe schlechter abschneidet, besser dazu geeignet ist genau diese Aufgabe zu erfüllen. Ein Generalisieren von diesem Ergebnis auf andere nichtlineare Aufgaben kann man nur unter Vorbehalt machen.

5.4.1 Muster Erkennung

Die Input-Output-Paare der Muster Erkennung werden erstellt, wie bereits im Kapitel 1.3 'Erste Experimente' beschrieben. Es wird ein Netz aus 96 ESN-Neuronen verwendet, das mit den Eingangs-Skalierungen $0.8^{-9} \approx 7.45$ bis $0.8^{40} \approx 0.0001$ getestet wird. α ist 0.85.

Ich definiere vorerst, wie ich in den Experimenten ein Fehlermass bestimme, um die verschiedenen Netze mit unterschiedlichen Skalierungen zu vergleichen:
Wenn ich ein Muster-Erkennungs-Experiment durchführe, erhalte ich Peaks an den Stellen, an denen das Muster erkannt werden soll. Diese Peaks sollten im Bestfall die Amplitude 1 haben, liegen jedoch in fast allen Experimenten im Intervall [0.1, 0.8]. An den Stellen, an denen das Muster nicht erkannt werden soll, sollte das Netz im Bestfall 0 ausgeben, man erhält jedoch üblicherweise ein 'Rauschen' um 0. Ich nehme nun an, dass die Reaktionen auf die beiden Klassen 'Muster' - 'Nicht-Muster' gaussverteilt sind und errechne für beide Klassen einen Mittelwert und eine Varianz. Wenn ich nun bei einem Ergebnis des Netzes entscheiden möchte, ob das Ergebnis als 'Muster' oder 'Nicht-Muster' gewertet wird, dann kann ich dies mit harten Schwellen durchführen, die ich auf die Schnittpunkte der beiden Gaussglocken lege. Ab diesen Punkten wird die Wahrscheinlichkeit, dass die zweite Musterklasse vorliegt, grösser als die, dass die erste Musterklasse vorliegt. Der Fehler (der Flächeninhalt der durch die Schwellen von den beiden Gaussglocken abgeschnitten wird) ist in diesem Fall am geringsten. Dieser Fehler ist mein Fehlermass für das Muster-Erkennungs-Experiment. Wenn sich die beiden Glocken nicht schneiden definiere ich den Fehler als die Fläche der kleineren Glocke. Abbildung 29 illustriert das Verfahren.

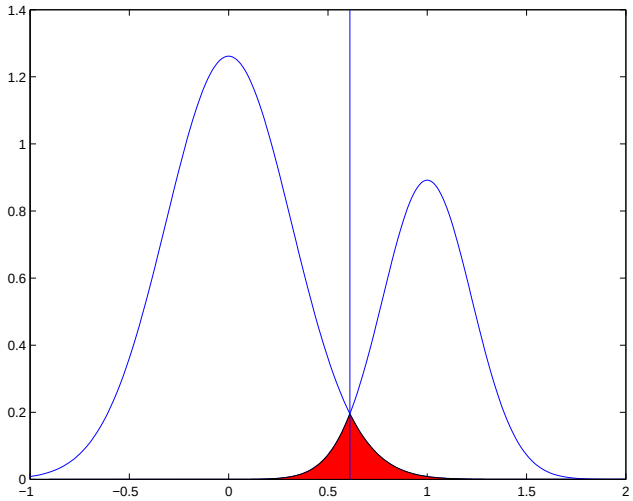

Abbildung 29: Fehlerfläche beim Schneiden der beiden geschätzten Gaussglocken

T_i Zeitpunkt-Menge, an denen Klasse i erkannt werden soll

$$
\begin{aligned}
T_{Nicht-Muster} = T_1 \quad & \forall t \in T_1 : \hat{o}(t) = 0 \\
T_{Muster} = T_2 \quad & \forall t \in T_2 : \hat{o}(t) = 1 \\
P(\omega_i) \quad & \text{Häufigkeit des Auftretens von Klasse i} \\
\mu_i = \frac{\sum_{t \in T_i} o(t)}{|T_i|} \quad & \text{Mittelwert der Ergebnisse von Klasse i} \\
\sigma_i^2 = \frac{\sum_{t \in T_i} (o(t) - \mu_i)^2}{|T_i|} \quad & \text{Varianz der Ergebnisse von Klasse i} \\
c \quad & \text{Schnittpunkt der beiden Gaussglocken} \\
e \quad & \mathit{Fehler}
\end{aligned}
$$

$$
c = \sigma_1 \mu_2 - \sigma_2 \mu_1 + \frac{\sqrt{2 (\sigma_2 - \sigma_1) log(\frac{P(\omega_2)}{P(\omega_1)} \sqrt{\frac{\sigma_1}{\sigma_2}}) - \mu_1 \mu_2}}{\sigma_2 - \sigma_1} \tag{54}
$$

$$
e = (1 - erf(\frac{c - \mu_1}{\sqrt{2 \sigma_1}})) \frac{P(\omega_1)}{2} + (1 + erf(\frac{c - \mu_2}{\sqrt{2 \sigma_2}})) \frac{P(\omega_2)}{2} \tag{55}
$$

Mit diesem Fehlermass habe ich zunächst 4 verschiedene Netze (vom Typ a, b, d und e aus Kapitel 5.2) mit linearen Transfer-Funktionen mit einem Muster der Länge 10 getestet. Da die Netze im Linearen mit allen Eingangsskalierungen gleich arbeiten (abgesehen von numerischen Instabilitäten) kann man die Qualitäten durch Tabelle 56 vollständig beschreiben.

$Netz$	(a)	(b)	(d)	(e)
$Fehler$	0.003	0.004	0.004	0.0004

(56)

Man sieht, dass sich die ersten drei Netze relativ ähnlich verhalten. Das e-Netz kann durch seine zusätzlichen korrelierten Eingänge das Muster mit linearer Transfer-Funktion besser erkennen als die anderen drei Netze.

Nun untersuche ich die Netze mit nichtlinearer Transfer-Funktion mit Variation der Eingangsskalierung. In Abbildung 30 sieht man die Ergebnisse der vier Netze mit einem Muster der Länge 10.

Ab einer Eingangsskalierung von circa 4 steigt der Diskriminierungsfehler stark an, so dass die Feinstruktur bei kleinen Fehlern nicht mehr zu erkennen ist. In Abbildung 31 werden daher dieselben Ergebnisse in einem anderen Skalierungs-Regime gezeigt. Es sei auf die unterschiedlichen Achsenskalierungen hingewiesen.

Man kann sehen, dass bei einer nichtlinearen Transferfunktion ab einer Eingangsskalierung von grösser als 4 die Netze völlig versagen, da in diesem Regime, wie wir in den VR-Experimenten gesehen haben, keine MC mehr vorhanden ist. Sobald die Eingangsskalierung sehr klein werden, werden die Diskriminierungs-Ergebnisse auch wieder langsam schlechter. Am besten schneidet das b-Netz ab, danach mit Faktor 10 schlechter das d-Netz. Die anderen beiden Netze sind um einige Zehnerpotenzen schlechter. Die Kopplung

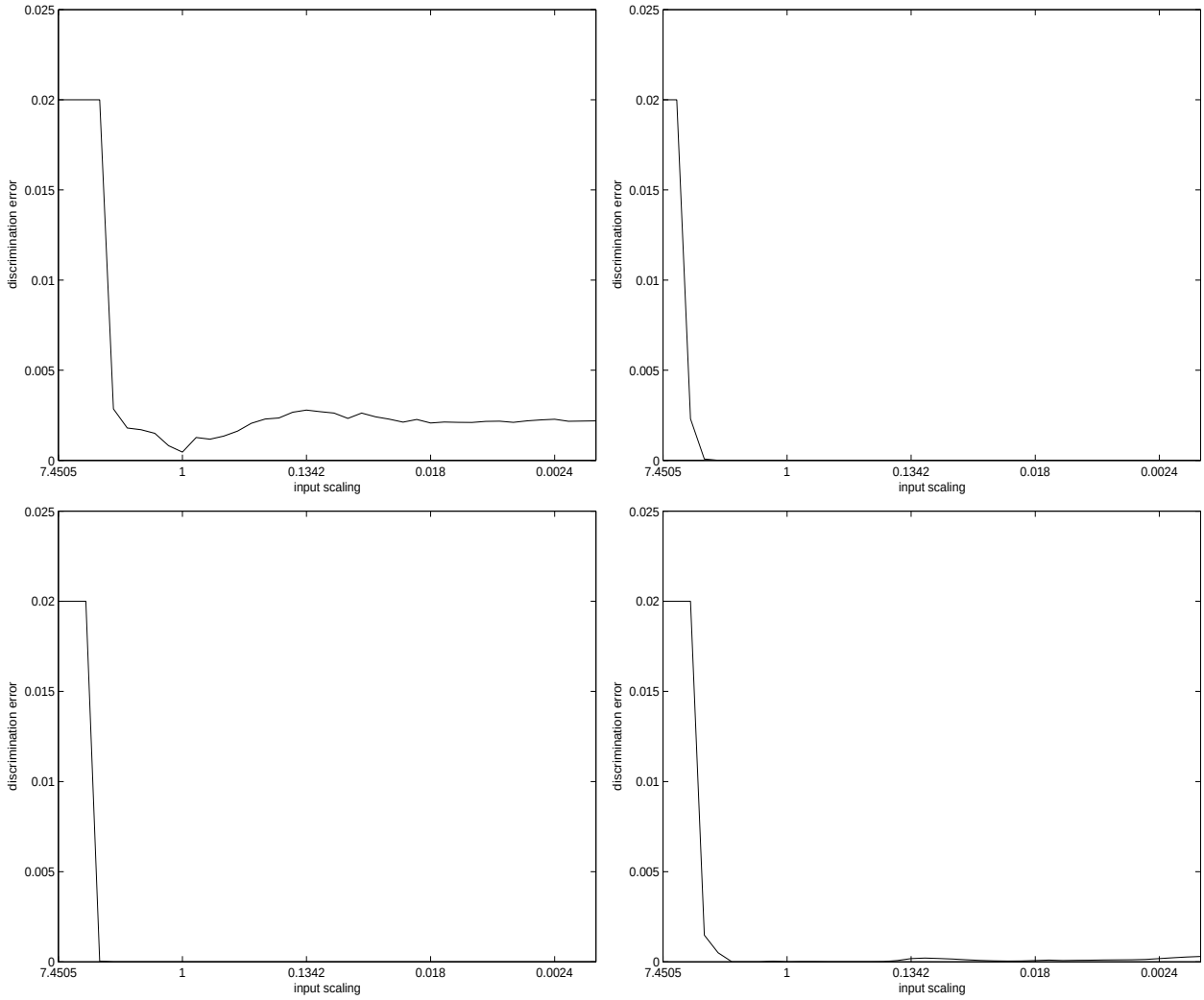

Abbildung 30: Diskriminierungsfehler des nicht-linearen a-Netzes (l.o.), b-Netzes (r.o.), d-Netzes (l.u.) und e-Netzes (r.u.) im Muster-Erkennungs-Experiment. Man sieht, dass alle Netze ab einer Eingangs-Sklaierung von circa 4 sehr schlechte Diskriminierungsergebnisse liefern. Das a-Netz hat auch bei niedriger Eingangsskalierung nicht so gute Ergebnisse wie die übrigen drei Netze.

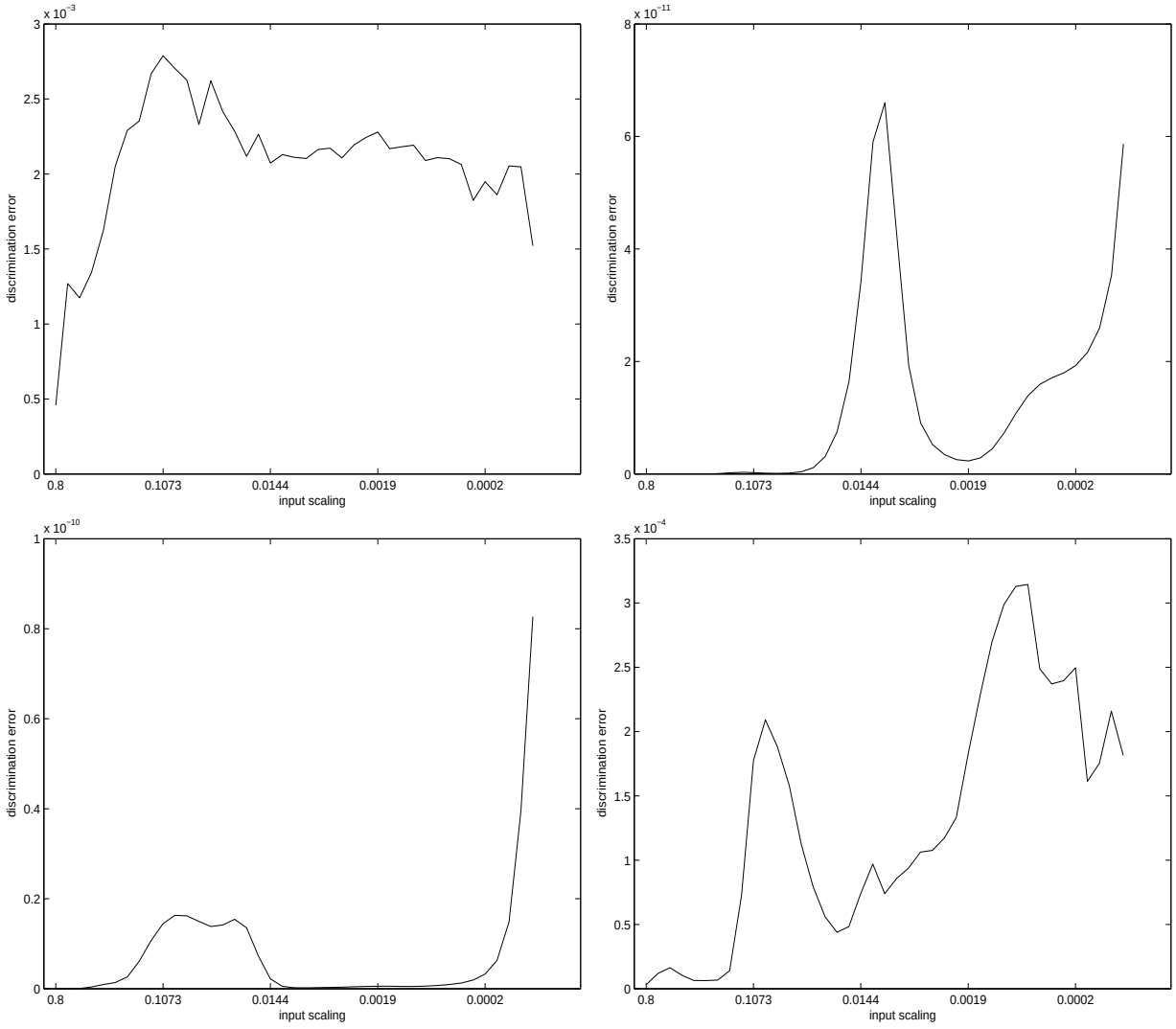

Abbildung 31: Diskriminierungsfehler des nicht-linearen a-Netzes (l.o.), b-Netzes (r.o.), d-Netzes (l.u.) und e-Netzes (r.u.) im Muster-Erkennungs-Experiment. Man sieht, dass a-, b- und d-Netz bei sehr kleinen Eingangsskalierungen wieder schlechter werden. Dies folgt daraus, dass die Transfer-Funktion sich bei sehr kleinen Eingangs-Skalierungen der Identität annähert.

der Neuronen untereinander scheint somit bei nichtlinearer Transferfunktion durchaus Vorteile zu bieten. Beim a-Netz hat die Nicht-Linearität für eine Verbesserung um den Faktor 10 gesorgt, beim e-Netz bringt sie fast keinerlei Vorteil.

5.4.2 Einsen zählen

Boolesche Funktionen wie im XOR-Experiment erscheinen mir als angebracht die nichtlinearen Verrechnungsstärken von ESNs zu testen, da die Funktionen in hohem Grade nichtlinear sind und sie sich in ihrem Wertebereich auch nicht gut linear approximieren lassen. Die Menge an zu testenden Funktionen lässt sich auf kompliziertere Funktionen ausweiten, wenn man als Input-Variablen noch weitere Vergangenheiten des Inputs zulässt. Ich habe mich dabei auf Experimente konzentriert, in denen die vorkommenden Einsen von einem binären Input-Strom gezählt werden sollen. Der XOR ist in diesem Hinblick der Test auf eine Eins in den letzten beiden Inputs. Man kann dies erweitern auf die Fragestellung ob in den letzten m Inputs n Einsen auftraten. Dabei muss man beachten, dass man durch den summierten Fehler in diesen Experimenten nicht nur die Verrechnungsstärke der Netze bewertet, sondern auch die Fähigkeit sich an die vergangenen Inputs zu erinnern, um diese dann verrechnen zu können. Ich verwende als Fehlermass wieder dieselbe Vorgehensweise wie bei der VR, d.h. quadrierter Fehler zwischen Soll- und Ist-Output. In den Tabellen (57) sieht man die Ergebnisse der Netze im Linearen für verschiedene Mengen an Neuronen. Es sollte immer in einem Zeitfenster der Länge n (inklusive dem Zeitpunkt t selbst) auf genau m Einsen geprüft werden (m in n). Der Spektralradius ist jeweils $0.001^{1/n}$.

Mit 12 Neuronen:

	(a)	(b)	(d)	(e)
$1in2$	0.27	0.26	0.28	$2.5e-13$
$2in3$	0.16	0.19	0.15	0.04
$2in4$	0.23	0.22	0.23	0.18
$2in5$	0.28	0.21	0.28	0.25

Mit 30 Neuronen:

	(a)	(b)	(d)	(e)
$1in2$	0.27	0.26	0.28	$1.7e-25$
$2in3$	0.16	0.21	0.15	0.04
$2in4$	0.23	0.24	0.23	0.18
$2in5$	0.25	0.18	0.25	0.25

Mit 60 Neuronen:

	(a)	(b)	(d)	(e)
$1in2$	0.27	0.28	0.27	$2.2e-14$
$2in3$	0.16	0.19	0.15	0.04
$2in4$	0.23	0.26	0.24	0.17
$2in5$	0.26	0.19	0.25	0.24

(57)

Man sieht, dass die Netze im linearen Regime sehr schlecht abschneiden. Allein das e-Netz

schafft es durch seine korrelierten Eingänge eine bessere Qualität zu erzielen. Wenn die Aufgabe jedoch nichtlineare Berechnungen benötigt, die nicht mehr durch die Korrelierten Eingänge mitgeliefert werden ist es genauso schlecht wie die übrigen Netze. Weiterhin sieht man, dass eine Erhöhung der Neuronenzahl keinen Vorteil bringt, was durchaus einleuchtend ist, da eine egal wie grosse Anzahl an linearen Einheiten es nicht schafft nichtlineare Verrechnungseigenschaften zu erlangen.

Nun teste ich die Netze mit nichtlinearer Transferfunktion mit unterschiedlichen Input-Skalierungen. In den Abbildungen 32 bis 35 sieht man die Ergebnisse der Netze im nichtlinearen Einsen-Zähl-Experiment. Man beachte die unterschiedlichen Achsenskalierungen.

Man kann sehen, dass die Nicht-Linearitäten sich in dieser Art Experiment bei jeder Netztopologie als äusserst vorteilhaft erweisen. Für eine komplexere Aufgabe wie '2 Einsen in den letzten 5 Inputs erkennen' ist eine gewisse Mindestmenge an Neuronen notwendig, um die Aufgabe in zufrieden stellender Weise zu lösen. Wenn mehr Neuronen vorhanden sind, als für eine Aufgabe notwendig sind, dann machen die überzähligen Einheiten das System stabiler gegenüber niedriger skaliertem Input. In sehr kleinen Input-Skalierungen nimmt der Fehler zu, da sich dann die Transfer-Funktion der Identität nähert. Je mehr Neuronen beteiligt sind, desto kleiner kann der Input skaliert werden, bis wieder eine Verschlechterung der Lösungsgüte eintritt. Ein überraschendes Ergebnis ist, dass sich in den meisten Fällen eine Eingangsskalierung von ca. 2 als am optimalsten herausstellt. In diesem Regime scheint die MC, die benötigt wird, um die letzten 3 (bzw. 2, 4 oder 5) Inputs zu rekonstruieren, gerade noch ausreichend zu sein, um die Aufgabe zu erfüllen. Die nichtlineare Krümmung des hochdimensionalen Lösungsraumes ist dagegen in diesem Fall am besten geeignet, um die nichtlineare Aufgabe zu erfüllen.

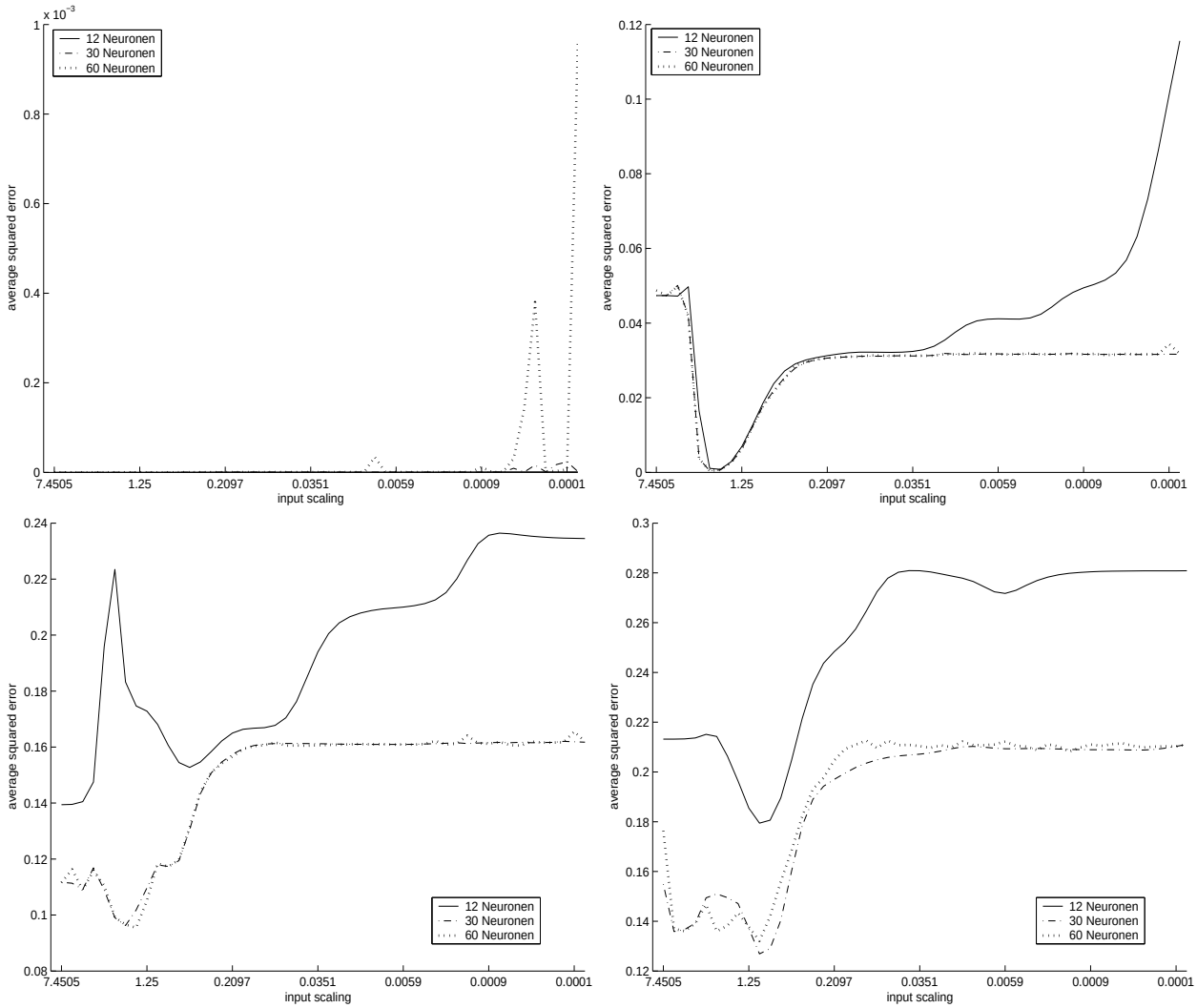

Abbildung 32: mse des a-Netzes bei verschiedenen Neuronen-Anzahlen im Nicht-Linearen bei der Detektion von genau m Eins in den letzten n Zeitschritten (l.o.: m = 1, n = 2; r.o.: m = 2, n = 3; l.u.: m = 2, n = 4; r.u.: m = 2, n = 5. Man sieht, dass das Netz bei dieser Aufgabe mit Eingangsskalierung um 2 am besten funktioniert. In diesem Regime ist die MC noch gerade ausreichend, um die zur Verrechnung nötigen Werte zu rekonstruieren und die nichtlineare Krümmung des hochdimensionalen Lösungsraumes ist maximal. Im Bild l.o. haben die unterschiedlichen Neuronenanzahlen nahezu keine Auswirkungen. Beim Bild r.o. haben die unterschielichen Neuronenzahlen mit hohen Eingangsskalierungen keinen Unterschied, mit niedriger Skalierung wird das Netz mit nur 12 Neuronen schlechter. Beim den übrigen beiden Bildern hat die geringe Anzahl von 12 Neuronen schlechtere Ergebnisse als mit mehr Neuronen. Zwischen 30 und 60 Neuronen kann nur ein geringer Unterschied festgestellt werden.

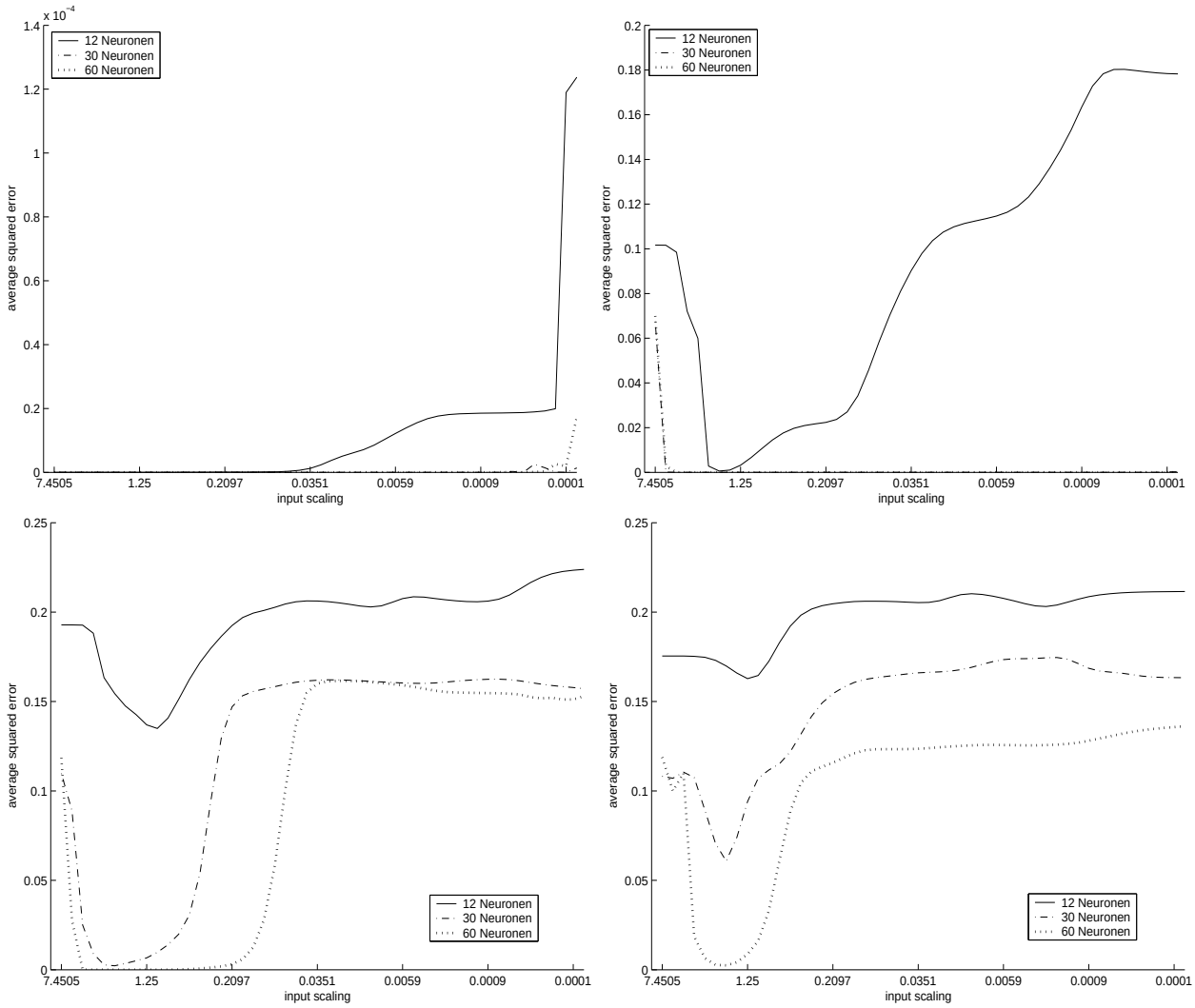

Abbildung 33: mse des b-Netzes bei verschiedenen Neuronen-Anzahlen im Nicht-Linearen bei der Detektion von genau m Eins in den letzten n Zeitschritten (l.o.: m = 1, n = 2; r.o.: m = 2, n = 3; l.u.: m = 2, n = 4; r.u.: m = 2, n = 5. Im Bild l.o. sieht man, dass alle drei Neuronenanzahlen mit grossen Eingangsskalierung einen sehr geringen Fehler haben und mit niedrigeren Eingangsskalierungen der Fehler zunimmt. Im Bild r.o. sieht man, das das Netz mit 30 und 60 Neuronen die Aufgabe bei fast allen Eingangsskalierungen sehr gut lösen kann, mit 12 Neuronen dagegen auf eine optimale Eingangsskalierung von cirac 2 angewiesen ist, um zu funktionieren. Im Bild l.u. sieht man, dass nur die Netze mit 30 und 60 Neuronen fähig sind, die Aufgabe zufriedenstellend zu lösen. Es sind dabei für diese beiden Netze hohe Eingangsskalierungen nötig, diese dürfen jedoch nicht so gross sein, dass zu viel an MC verloren geht. Das Netz mit 60 Neuronen kann die Aufgabe mit niedrigeren Eingangsskalierungen lösen als das Netz mit 30 Neuronen. Im Bild r.u. sieht man, dass die schwierige Aufgabe genau 2 Einsen in den letzen 5 Zeitschritten nur vom Netz mit 60 Neuronen mit niedrigem Fehler gelöst werden kann. hierfür ist auch eine Eingangsskalierung von circa 2 nötig.

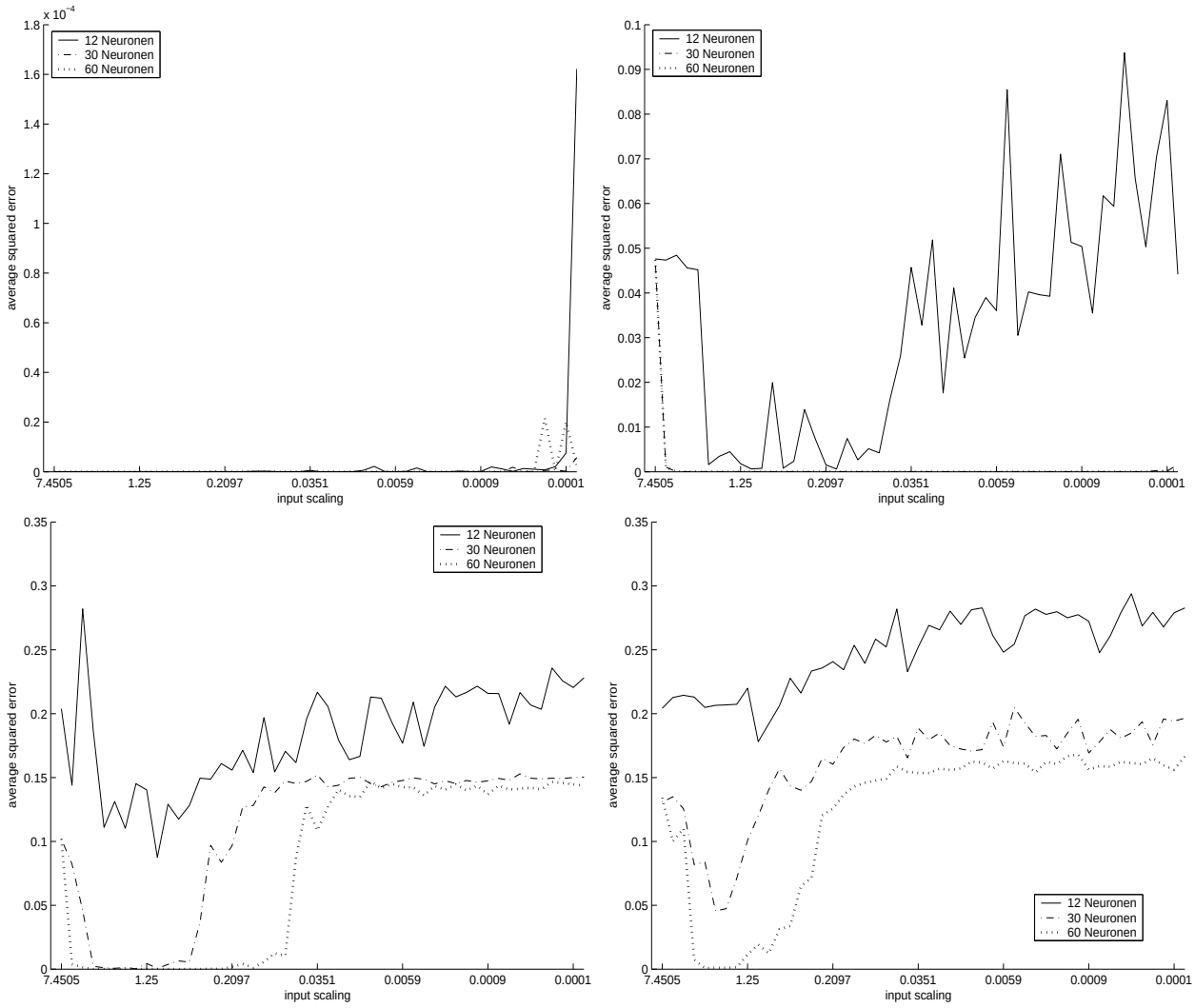

Abbildung 34: mse des d-Netzes bei verschiedenen Neuronen-Anzahlen im Nicht-Linearen bei der Detektion von genau m Eins in den letzten n Zeitschritten (l.o.: m = 1, n = 2; r.o.: m = 2, n = 3; l.u.: m = 2, n = 4; r.u.: m = 2, n = 5. Die Ergebnisse ähneln stark denen aus Abbildung 33, nur dass die Kurven-Verläufe unregelmässiger aussehen. Nur in den Bildern l.o. und r.o. sind die Ergebnisse besser als beim b-Netz.

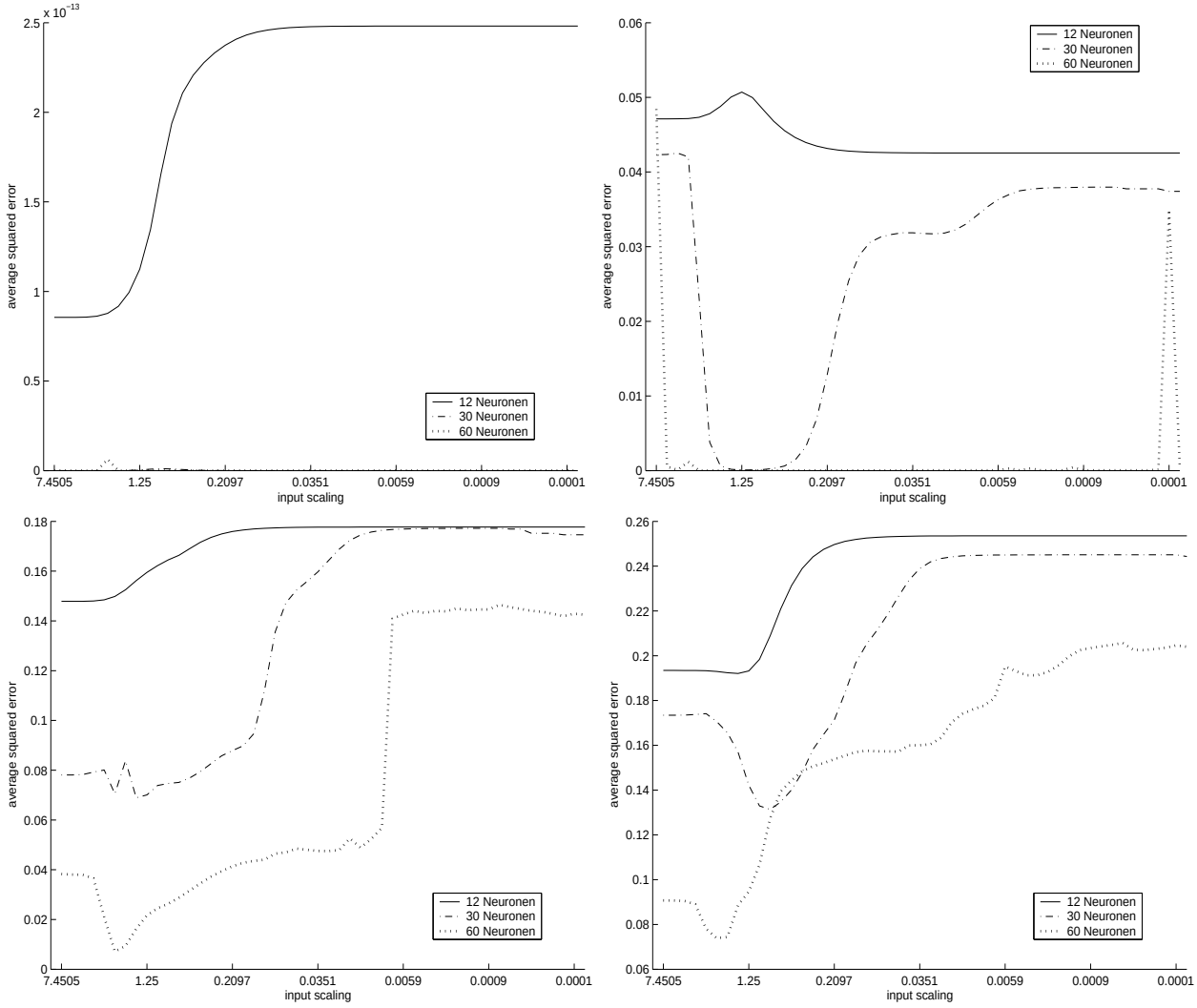

Abbildung 35: *mse* des e-Netzes bei verschiedenen Neuronen-Anzahlen im Nicht-Linearen bei der Detektion von genau m Eins in den letzten n Zeitschritten (l.o.: m = 1, n = 2; r.o.: m = 2, n = 3; l.u.: m = 2, n = 4; r.u.: m = 2, n = 5. Die Ergebnisse dieses Netzes unterscheiden sich stark von denen der anderen Netze. Die Aufgabe genau eine Eins in den letzen zwei Zeitschritten zu detektieren (l.o.) wird mit extrem niedrigem Fehler gelöst. Der Grund hierfür ist, dass die Lösung der Aufgabe bereits mit einem der Eingänge eingespeist wird. Aber auch die übrigen drei Aufgaben werden mit besseren Ergebnisen gelöst als die anderen drei Netze vorweisen können. Dieses Ergebnis sollte jedoch nicht überschätzt werden, da die Aufgabe Korrelationen im Input zu entdecken für dieses Netz einfacher zu lösen ist, da es mit korrelierten Eingängen gespeist wird.

6 Anwendungs-Beispiel

Wie man in den bisherigen Erläuterungen sehen konnte, kann man mit ESNs sowohl vergangene Inputs speichern als auch diese verrechnen. Ich möchte nun in einem Anwendungsbeispiel diese Fähigkeiten praktisch einsetzen. Mein Experiment-Aufbau wurde schon in ähnlicher Form in [6], [2] ausprobiert, allerdings mit sehr viel komplexeren Netzen aufgebaut aus biologisch orientierten Leaky-Integrate-and-Fire-Neuronen.

Ich möchte ein Netz konstruieren, dass die Geschwindigkeit von Punkten, die sich über ein rezeptives Feld bewegen, registrieren kann.
Ich verwende als Eingangs-Signal-Schicht eine Matrix aus 15 mal 15 Pixeln, bei der jedes Pixel mit einem Paket aus 9 Neuronen mit Verbindungsgewichtungen von 1 verbunden ist. Die Pakete sind in sich vollständig mit zufälligen (gaussverteilten) Gewichten verbunden und auf einen Spektralradius von 0.7 skaliert. Jeder Pixel kann sich somit eine kurze Vergangenheit von ca. 5 Zeitschritten merken. Alle Neuronen sind mit $\tanh(x)$ als nichtlinearer Transfer-Funktion ausgestattet.
Über das Feld wird nun ein Punkt bewegt, der sich in jedem Durchlauf mit einer der zufällig gewählten Geschwindigkeiten 1, 2 oder 3 Pixel pro Zeiteinheit bewegt. Der Start der Bahn des Punktes liegt zufällig verteilt auf dem Rand des 15 mal 15 Feldes und endet auf einem zufälligen Feld auf der gegenüberliegenden Seite des Startpunktes. Zu jedem Zeitschritt erzeugt der Punkt auf den Neuronen auf denen er sich befindet sowie gauss'sch abfallend auf den Neuronen in seiner lokalen Umgebung, eine Erregung. Nachdem der Punkt das Feld verlassen hat wird ein neuer Durchlauf nach demselben Muster gestartet. An den $9 \cdot 15 \cdot 15 = 2025$ Neuronen umfassenden Echo-Pool sind 3 Output-Neurone angebracht, die zu jedem Zeitschritt detektieren sollen, ob sich der Punkt mit ihrer Geschwindigkeit bewegt. Hat der Punkt z.B. die Geschwindigkeit 1 Feld-Länge pro Zeiteinheit, dann soll Output 1 gleich 1 und die anderen beiden Ausgänge gleich 0 sein. Wenn man sich die aufgezeichnete Vergangenheit eines jeden Pixels als Vektor vorstellt, der oben die jüngsten Ereignisse und unten die am längsten vergangenen Ereignisse speichert, dann hinterlässt der Punkt beim Durchlaufen des Feldes ein Abbild im Vergangenheitsvektor der Neuronen, über die er hinwegzieht. Zu jedem Zeitpunkt ist dieses Abbild im Vektor umso tiefer, je länger der Zeitpunkt der Traversierung zurückliegt. Zwei in einer fixen Entfernung liegende Pixel, die von dem Punkt überstreift werden, können nun durch den Vergleich ihrer Vergangenheiten die Geschwindigkeit des Punktes herausfinden. Sobald das Abbild in den Vektoren der beiden Pixel vollständig vorhanden ist, stellt die Verschiebung des Punkt-Abbildes in den beiden Vektoren die Geschwindigkeit des Punktes dar. Je langsamer der Punkt ist, desto tiefer ist das Abbild im ersten Vektor nach unten gewandert, bis sich der Punkt im zweiten Vektor zeigt. Jeder Output-Pool muss somit ein pattern-matching durchführen, indem er eine hohe Korrelation in zwei unterschiedlichen Verschiebungen der Vergangenheitsvektoren der Neuronen detektieren muss.
Für das Training werden 1000 Durchläufe durchgeführt. Nach dem Training sind 30 Testläufe vorgenommen worden. In den Bildern in Abbildung 36 sieht man die Ergebnisse der Tests

des Experimentes. Man kann sehen, dass das Netz im allgemeinen ein gutes Ergebnis liefert. Das Netz brauch nach dem Auftauchen des Punktes im rezeptiven Feld einen kurzen Zeitraum, bis es genügend Information aufgenommen hat, so dass eine verlässliche Berechnung des Ergebnisses erfolgen kann. Der Grund ist, dass die Bewegung verständlicherweise erst dann detektiert werden kann, wenn sich der Punkt ein paar Felder bewegt hat und damit Strukturinformation in den Vergangenheitsspeicherungen der ESN-Pools vorhanden ist, die dann über verschiedene Orte verglichen werden kann.

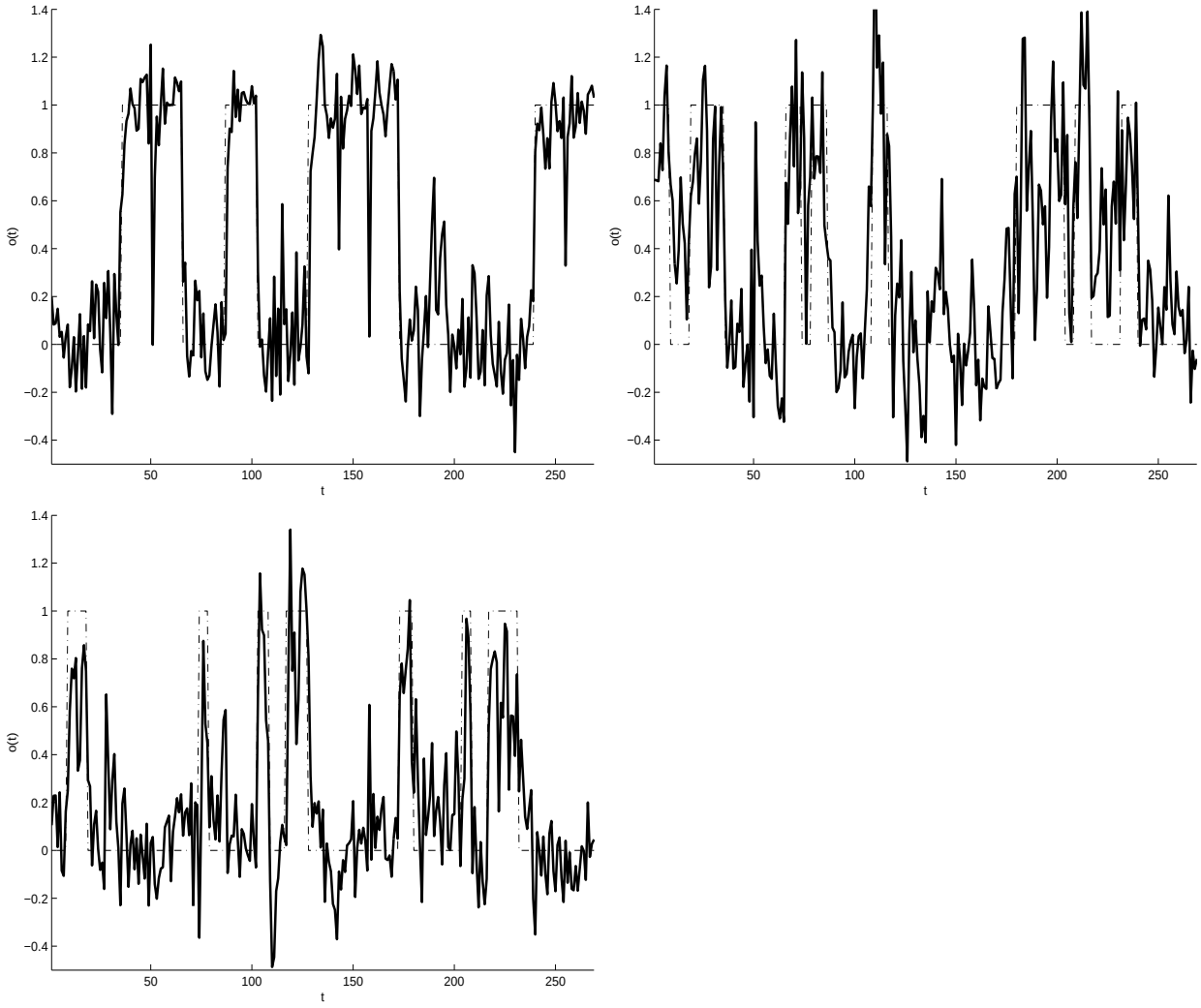

Abbildung 36: Ein Punkt wird wiederholt mit unterschiedlichen Geschwindigkeiten über ein rezeptives Feld bewegt. Die drei Ausgangs-Neuronen sollen erkennen, ob der Punkt sich mit der ihnen zugeordneten Geschwindigkeit bewegt. In den Bildern sieht man beim Test nach dem Training die Aktivitäten an den Ausgängen, die jeweils eine der drei Geschwindigkeiten 1 (l.o), 2 (r.o) und 3 (l.u) Feld-Längen pro Zeitschritt detektieren sollen. Circa alle 10 Zeitschritte läuft ein weiterer Punkt durch das Feld, was man daran sieht, dass die verschiedenen Geschwindigkeiten immmer circa 10 Zeitschritte dicke Blöcke mit Soll-Aktivität 1 haben. Nach einem Durchlauf fällt die Erkennungsqualität für eine spezifische Geschwindigkeit erst wieder fast auf 0 ab, da der neue Durchlauf an einer neuen unbekannten Stelle auf dem rezeptiven Feld beginnt. Die neue Geschwindigkeit wird jedoch bereits nach 2 bis 3 Zeitschritten erkannt. Besonders gut kann man diesen Effekt bei Geschwindigkeit 1 im Bild links-oben beobachten.

7 Zusammenfassung und Ausblick

Im zweiten und dritten Kapitel habe ich gezeigt, dass die anfangs unverständlichen Transformationen, die durch die Netzwerkmatrix und die Auslesegewichte durchgeführt wurden sich durch anschauliche Transformationen (Kernels) beschreiben lassen, indem man die Netzwerkmatrix diagonalisiert und das äquivalente, vereinfachte System als Summation von gedämpften Oszillationen beschreibt. Dadurch konnte der Vorgang der Vergangenheits-Rekonstruktion vollständig erklärt werden und mit dieser Hilfe auch die Netzwerkfähigkeit der Mustererkennung im Linearen. In Kapitel 4 habe ich vorerst für den speziellen Fall der homogenen Netzwerke gezeigt, dass die memory-Kapazität gleich der Netzwerkgrösse ist und dies im Appendix auch für allgemeine ESNs gezeigt. Weiterhin habe ich im 4. Kapitel erläutert, wie sich Abweichungen vom idealen Modell der homogenen Netze auf die Eigenschaften der Netze auswirken. Eine grundlegende Erkenntnis dabei war, dass alle ESNs zwar eine theoretische Vergangenheits-Speicherfähigkeit (memory capacity MC) von n besitzen, homogenen Netze mit einem vernünftig gewählten Spektralradius jedoch am robustesten für die Vergangenheits-Rekonstruktion waren.
Im 5. Kapitel habe ich experimentell gezeigt, dass mit nichtlinearer Transferfunktion die Berechnungsfähigkeiten von nichtlinearen Funktionen um ein Vielfaches grösser sind als mit linearer Transferfunktion und dass im Nicht-Linearen die Speicherfähigkeit der Netze mit klein skalierten Eingängen am grössten ist. Als Transferfunktion können dafür vielfältige nichtlineare Funktionen dienen. Im nichtlinearen Regime haben wir gesehen, dass die Netzwerkparameter starken Einfluss auf die Fähigkeiten des Netzes haben und daher je nach Experiment das Netzwerk speziell eingestellt werden muss (Eingangsskalierung, Spektralradius).
In Kapitel 6 habe ich gezeigt, dass sich die beschriebene Technik eignet, um Muster erkennen zu können, die als Verschiebung im Abbild der Vergangenheiten zweier lokal verschobener Einheiten definiert sind. Damit wurde eine Erkennung von spezifischen Geschwindigkeiten möglich.
Die aufgezeigten Ergebnisse sollten ein Stück dazu beitragen ESNs ihren mystischen Glanz des Unerklärbaren zu nehmen und ihnen ein analytisches Fundament zu geben. Als weitere zu lösende Fragestellungen bleibt offen, wie sich Rauschen auf den einzelnen Neuronen bei beliebigen ESN (und nicht nur bei homogenen Nezten) auswirkt und eine analytische Erfassung wie sich Nicht-Linearitäten auf die Netzwerkeigenschaften auswirken. Weiterhin könnte man das Experiment der Geschwindigkeitserkennung verfeinern um das Training schneller und effizienter stattfinden zu lassen.
Es liessen sich sicherlich auch noch andere Einsatzgebiete finden, in denen man eine Berechnung basierend auf einem Strom von seriell eintretenden Informationseinheiten vornehmen muss. Ist in diesem Fällen eine Berechnung mit konventionellen Methoden zu aufwendig, so könnten ESNs durch ihre massiv parallele Verarbeitung und ihren einfachen Lernalgorithmus eine sinnvolle Alternative darstellen.

8 Appendix

8.1 Verwendete Abkürzungen

n	$\mathbb{N}$	Anzahl der Echo-Neuronen
m	$\mathbb{N}$	Anzahl der Eingänge
t_{tr}	$\mathbb{N}$	Länge des Trainings in Zeitschritten
$\mathbf{x}(t)$	$\mathbb{C}^n$	Zustands-Vektor der Neuronen der Echo-Schicht zum Zeitpunkt t
X	$\mathbb{C}^{t_{tr} \times n}$	Matrix aller gemessenen Netzwerkzustände in der Echo-Schicht
γ	$\mathbb{C} \to \mathbb{C}$	Transfer-Funktion, gültig für jedes Neuron
A	$\mathbb{C}^{n \times n}$	Verbindungsmatrix der Echo-Neuronen
M	$\mathbb{C}^{n \times n}$	reelle Bandmatrix als Verbindungsmatrix der Echo-Neuronen
ρ_A	$\mathbb{R}$	Spektralradius von A
D	$\mathbb{C}^{n \times n}$	Diagonalmatrix von A nach der Basistransformation
d_v	$\mathbb{C}$	Gewicht der rekurrenten Kante von Neuron v
α	$\mathbb{R}$	exponentieller Verfallsfaktor eines einzelnen Neurons ($= \lvert d \rvert$)
s	$\mathbb{N}$	Verschiebung eines Delta-Pulses
ω	$\mathbb{R}$	Winkel-Frequenz eines einzelnen Neurons ($= \varphi(d)$)
$\mathbf{u}(t)$	$\mathbb{C}^m$	Eingangsvektor zum Zeitpunkt t
B	$\mathbb{C}^{m \times n}$	Verbindungsmatrix zwischen den Eingangs- und den Echo-Neuronen
$\mathbf{h}(t)$	$\mathbb{C}^n$	$= B\,u(t)$
Δt	$\mathbb{R}$	Zeitschrittweite
$\mathbf{w}$	$\mathbb{C}^n$	Auslesevektor
$q_v(t)$	$\mathbb{C}$	Anteil eines Neurons v am Gesamtausgang o(t)
$o(t)$	$\mathbb{C}$	Ausgangsfunktion
$\mathbf{o}$	$\mathbb{C}^{t_{max}}$	Vektor der Ist-Ausgänge
$\hat{\mathbf{o}}$	$\mathbb{C}^{t_{max}}$	Vektor der Soll-Ausgänge
g	$\mathbb{C} \to \mathbb{C}$	Kernelfunktion eines Neurons
p	$\mathbb{C} \to \mathbb{C}$	Gesamtkernelfunktion
$\mathbf{z}(t)$	$\mathbb{C}^n$	Rauschvektor
ξ	$\mathbb{C}$	einmalige Ziehung einer komplexen Zufallsvariable

8.2 Rückkopplung der Ausgänge

Eine Rückkopplung der Ausgänge in die Echo-Schicht kann als weiterer Eingang aufgefasst werden. In der Trainingsphase, in der die Ausgangsgewichte noch nicht gesetzt worden sind, muss dieser Ausgang mit dem gewünschten (korrekten) Ausgangssignal beschrieben werden, damit simuliert werden kann, dass dieser Ausgang mit gesetzten Ausgangsgewichten den korrekten Ausgang erzeugt. Dieser Vorgang wird von Jäger als 'teacherforcing' bezeichnet. Das Rückkoppeln führt jedoch das Problem mit sich, dass nach dem Lernen der Ausgängsgewichte der Output nicht hundertprozentig dem Wunschausgang entspricht. Diese Diskrepanz wird dem Netzwerk wieder eingespeist und der Abstand zwischen bekannten Eingangsmustern, sowie dazu erzeugten Ausgangsmustern, und ihren Soll-Äquivalenzen wird immer grösser. Im 'Little-Red-Riding-Hood'-Experiment von Jäger [3], versuchte Jäger dem Netz das Rezitieren von Rotkäppchen und dem bösen Wolf anzutrainieren. Er verwendete circa 30 Ein- und Ausgänge für die verschiedenen Buchstaben und Satzzeichen und schaltete zu jedem Zeitschritt den Eingang, der dem aktuellen Satzzeichen entsprach auf 1 und den Rest auf 0. Als gewünschten Ausgangvektor verlangte er den Ausgang der dem folgenden Satzzeichen entsprach auf 1 und den Rest auf 0.
Das trainierte Netz begann den Anfang des Märchens korrekt wiederzugeben. Nach den ersten Fehlentscheidung verfing es sich in weiteren Fehlprognosen bis es sich schliesslich in einem Kauderwelsch-Attraktor aus wiederholten Wortfetzen gefangen hatte. Ich würde das Verhalten des Netzes erklären, dass der 'fast richtige' Ausgang irgendwann derartige Fehler macht, dass die davor liegenden korrekten Zustände nicht mehr ausreichen, um dem Netz wieder auf einen korrekten Ausgang zu lenken. Das Netz verliert irgendwann vollständig seinen Soll-Ablauf. Die immer schlechter werdenden Prognosen des Netzes lassen sich vergleichen mit einem akkumulierten Rauschen, dass das Netz früher oder später zum Zusammenbruch führt. Ähnlichkeiten zu derartigen Effekten kann man auch bei Experimenten mit Reinforcement Learning [8] beobachten, bei denen die Prognosen in die weitere Zukunft immer schlechter werden, weil sie auf Iterationen basieren, die immer schlechtere Ausgangsdaten zur Verfügung stellen.
Weiterhin ist zu beachten, dass durch die weitere Rückkopplung, die das System erfährt, eventuell die fading-memory-Eigenschaft verletzt wird. Wenn die Matrix A vorher einen Spektralradius nahe 1 besass, dann könnte das System nach Rückkopplung der Ausgänge instabil werden.

8.3 Ersatzschaltbilder

In den Spalten der Matrix T aus Kapitel 2.1 stehen die normierten Eigenvektoren der Matrix A. Auf der Diagonalen der Matrix D, stehen die Eigenwerte der Matrix A. Dank der Rangvollständigkeit von A sind diese Eigenwerte alle verschieden und ungleich 0. Je nach Wahl der Matrixelemente sind manche dieser Eigenwerte rein reell, manche komplex. Komplexe Eigenwerte treten immer als Paar von zueinander komplex konjugierten Zahlen auf. Diese Eigenschaft lässt sich anhand einer Analyse der Eigenwerte einer 2×2-Matrix

anschaulich illustrieren.

$$
\begin{aligned}
det(\begin{pmatrix} a & b \\ c & d \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}) &= 0 \\
(a-\lambda)(d-\lambda) - cb &= 0 \\
\lambda^2 - \lambda\,(a+d) + ad - bc &= 0 \qquad (58)
\end{aligned}
$$

Dies ist eine quadratische Gleichung, die entweder 2 reellwertige Lösungen besitzt oder 2 komplex konjugierte. (Setzt man z.B. $a = d$ und $-b = c$, dann erhählt man $\lambda_{1,2} = a \pm i\,b$, setzt man wiederum z.B. $a = d$ und $b = c$, dann erhält man $\lambda_{1,2} = a \pm \sqrt{b^2}$).
Ordnet man die Zeilen und Spalten in D so an, dass die komplex konjugierten Eigenwerte diagonal nebeneinander stehen, so kann man die komplexe Matrix D umformen in $D = T^T M T$ mit $M^{n\times n}, m_{i,j} \in \mathbb{R}$

$$
D = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & a+ib & 0 & \cdot \\ \cdot & 0 & a-ib & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} = T\,M\,T^T =
$$

$$
\frac{1}{\sqrt{2}} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & i & \cdot \\ \cdot & 1 & -i & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & a & b & \cdot \\ \cdot & -b & a & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & 1 & \cdot \\ \cdot & -i & i & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \frac{1}{\sqrt{2}} \qquad (59)
$$

Jede beliebige Matrix mit vollem Rang lässt sich also in eine reellwertige Matrix umwandeln, die aus Einzel-, bzw. maximal Zweier-Blockmatrizen auf der Diagonalen besteht und den komplexen Nebenmatrizen. Die Matrizen T und T^T, die blockweise eine orthogonale Transformation durchführen, lassen sich in die Eingangs- und Ausgangsgewichte hineinmultiplizieren.

$$
\begin{aligned}
\mathbf{x}(t) &= D\,\mathbf{x}(t-1) + B\,u(t) \\
\mathbf{x}(t) &= T^T M\,T\,\mathbf{x}(t-1) + B\,u(t) \\
T\,\mathbf{x}(t) &= M\,T\,\mathbf{x}(t-1) + T\,B\,u(t) \\
\Rightarrow \quad \hat{B} &= T\,B = T\,(1,1)^T = (\sqrt{2},0)^T \qquad (60) \\
o(t) &= \mathbf{w}\,\mathbf{x}(t) = \mathbf{w}\,T^T\,T\,\mathbf{x}(t) = \hat{\mathbf{w}}\,\hat{\mathbf{x}} \\
\Rightarrow \quad \hat{\mathbf{w}} &= \mathbf{w}\,T^T \qquad (61)
\end{aligned}
$$

Die Veränderung ist eine Basistransformation der internen Zustände und verändert daher die linearen Eigenschaften des Gesamtsystems nicht.
Die Einzelblöcke auf der Diagonalen lassen sich im Neuronen-Schaltbild als einzelne Neurone interpretieren, die eine rekurrente Kante mit einem Gewicht entsprechen dem Matrixeintrag besitzen. Die Erregung eines dieser Neurons verhält sich im diskreten Zeitverlauf wie eine e-Funktion entsprechend Gleichung (11) mit reellem Exponenten. Dies führt zu einem Abklingen eines Impulses ohne Oszillation.

Die Zweier-Blockmatrizen entprechen einem Schaltbild, in dem zwei Neurone gegenseitig mit den Gewichten a und b wie in Abbildung 38 gezeigt verschaltet sind. Ihre jeweiligen Erregungszustände lassen sich durch durch Gleichung (11) mit komplexem Exponenten beschreiben. Dabei stellen in Abbildung 38 der Eingang u_1 den Realteil und u_2 den Imaginärteil eines eingehenden Impulses, bzw. der Ausgang x_1 den Realteil und x_2 den Imaginärteil einer Impulsantwort, dar.

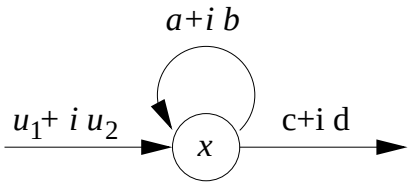

Abbildung 37: Schaltbild eines Neurons mit komplexem rekurrentem Gewicht

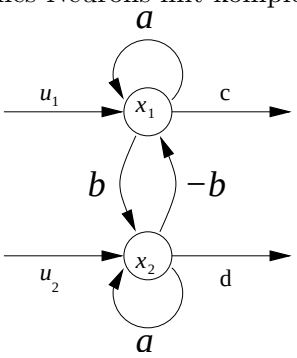

Abbildung 38: Ersatzschaltbild mit 2 Neuronen, die in ihrer Kombination dasselbe Verhalten zeigen wie Schaltbild 37

$$\begin{aligned} g(t) &= d^t = e^{t\,ln(d)} = e^{t\,(ln(\alpha)+i\,\omega)} = e^{t\,ln(\alpha)}\,e^{i\,t\,\omega} \\ &= \alpha^t\,(cos(t\,\omega) + i\,sin(t\,\omega)) \\ x_1(t) &= \alpha^t\,cos(t\,\omega) \qquad (62) \\ x_2(t) &= \alpha^t\,sin(t\,\omega) \qquad (63) \end{aligned}$$

Eine Phase kann durch die Kombination der beiden oszillierenden e-Funktionen erreicht werden. Die beiden sich gegenseitig dämpfenden Neurone schwingen mit derselben Frequenz aber mit einer Phasenverschiebung von $\frac{1}{2}\pi$. Bei der linearen Kombination der Erregungen dieser beiden Neurone ergibt sich eine resultierende Schwingung mit derselben Frequenz aber einer Phasenverschiebung, die je nach den Gewichtungen der Ausgänge w_1 und w_2 ausfällt.

$$\begin{aligned} o(t) &= w_1\,\alpha^t\,cos(t\,\omega) + w_2\,\alpha^t\,sin(t\,\omega) \\ &= \alpha^t\,(w_1\,sin(t\,\omega + \frac{1}{2}\pi) + w_2\,sin(t\,\omega)) \\ &= \alpha^t\,(\sqrt{{w_1}^2 + {w_2}^2}\,sin(t\,\omega + \frac{w_1}{w_2})) \qquad (64) \end{aligned}$$

8.4 allgemeiner Beweis für $MC = n$

Ein ESN mit n Echo-Neuronen, einem Eingang, der vollständig mit beliebigen Gewichten mit der Echo-Schicht vernetzt ist, einem Ausgang und einer beliebigen Verbindungsmatrix M (M muss vollen Rang haben für die Invertierbarkeit von Z in (69)) hat eine memory capacity $MC = n$.
M lässt dich zu D diagonalisieren.

$$\mathbf{x}^{\bar{t}} = (x_1^t, ..., x_n^t)^T \tag{65}$$

$$\begin{aligned} p(t) &= \mathbf{w}^T \mathbf{d}^{\bar{t}} \\ &= \sum_{v=1}^{n} w_v \, d_v^t \end{aligned} \tag{66}$$

$$\hat{p}_s(t) = \delta(t-s) \tag{67}$$

$$\begin{aligned} e(\mathbf{w}) &= \sum_{t=0}^{\infty} (p(t) - \delta(t-s))^2 \\ &= 1 - 2\,p(s) + \sum_{t=0}^{\infty} (p(t))^2 \\ &= 1 - 2 \sum_{v=1}^{n} w_v \, d_v^s + \sum_{t=0}^{\infty} (\sum_{v=1}^{n} w_v \, d_v^t)^2 \end{aligned} \tag{68}$$

$$\begin{aligned} \frac{\delta e(\mathbf{w})}{\delta w_i} &= -2\, d_i^s + \sum_{t=0}^{\infty} 2 \sum_{v=1}^{n} w_v \, d_v^t \, d_i^t \\ &= -2\, d_i^s + \sum_{v=1}^{n} \frac{2\, w_v}{1 - d_v \, d_i} \\ \Rightarrow \quad 0 &= Z\, \mathbf{w}_s^{opt,s} - \mathbf{d}^{\bar{s}} \qquad \text{mit } Z = \begin{pmatrix} \frac{1}{1-d_1\,d_1} & \cdots & \frac{1}{1-d_1\,d_n} \\ \cdots & \cdots & \cdots \\ \frac{1}{1-d_n\,d_1} & \cdots & \frac{1}{1-d_n\,d_n} \end{pmatrix} \\ \Rightarrow \quad \mathbf{w}^{opt,s} &= Z^{-1}\, \mathbf{d}^{\bar{s}} \end{aligned} \tag{69}$$

$$\begin{aligned} \sum_t (\sum_v w_v^{opt,s} \, d_v^t)^2 &= \sum_t \sum_v \sum_u w_v^{opt,s} \, w_u^{opt,s} \, d_v^t \, d_u^t \\ &= \sum_v w_v^{opt,s} \sum_u w_u^{opt,s} \frac{1}{1 - d_v \, d_u} \\ \text{mit (69)} \qquad &= \sum_v w_v^{opt,s} \sum_u \sum_c (Z^{-1})_{cu} \, d_c^s \frac{1}{1 - d_u \, d_v} \\ &= \sum_v w_v^{opt,s} \sum_c d_c^s \sum_u (Z^{-1})_{cu} \, Z_{uv} \end{aligned}$$

$$= \sum_v w_v^{opt,s}\, d_v^s \tag{70}$$

$$mc(s) = \frac{((\mathbf{w}^{opt,s})^T\, \mathbf{d}^{\bar{s}})^2}{\sum_{t=0}^{\infty} ((\mathbf{w}^{opt,s})^T\, \mathbf{d}^{\bar{t}})^2}$$

$$\text{mit (70)} \quad = (\mathbf{w}^{opt,s})^T\, \mathbf{d}^{\bar{s}}$$

$$= \sum_{v=1}^{n} w_v^{opt,s}\, d_v^s$$

$$= \sum_{v=1}^{n}\sum_{u=1}^{n} (Z^{-1})_{uv}\, d_u^s\, d_v^s \tag{71}$$

$$MC = \sum_{s=0}^{\infty} mc(s)$$

$$\text{mit (71)} \quad = \sum_{s=0}^{\infty}\sum_{v=1}^{n}\sum_{u=1}^{n} (Z^{-1})_{uv}\, d_u^s\, d_v^s$$

$$= \sum_{v=1}^{n}\sum_{u=1}^{n} (Z^{-1})_{uv}\, \frac{1}{1 - d_v\, d_u}$$

$$= \sum_{v=1}^{n}\sum_{u=1}^{n} (Z^{-1})_{uv}\, Z_{vu}$$

$$= \sum_{v=1}^{n} (Z^{-1}\, Z)_{vv}$$

$$= \sum_{v=1}^{n} 1 = n \qquad \square \tag{72}$$

8.5 Girko's circular law

Vyacheslav L. Girko bewies 1984 [9] folgenden Satz:

Sei $A \in \mathbb{C}^{n\times n}$ eine Matrix, deren Einträge $a_{i,j}$ normalverteilt mit Mittelwert 0 und Varianz 1 gezogen werden und λ die Eigenwerte von A. Dann ist, wenn $n \to \infty$, $\lambda/\sqrt{n}$ gleichverteilt im komplexen Einheitskreis.
In Abbildung 39 sieht man zwei Beispiele für n = 400 und n = 1000. Aus dem Satz folgt, dass die Winkel der Eigenwerte ebenfalls gleichverteilt sind. Die Beträge der Eigenwerte sind entsprechend dem Umfang des Kreises auf dem der Eigenwert liegt verteilt. Die Dichte der Eigenwert-Beträge nimmt somit von Eigenwerten von 0 bis 1 quadratisch zu. Diese Zusammenhänge sind in Abbildung 40 zu sehen.

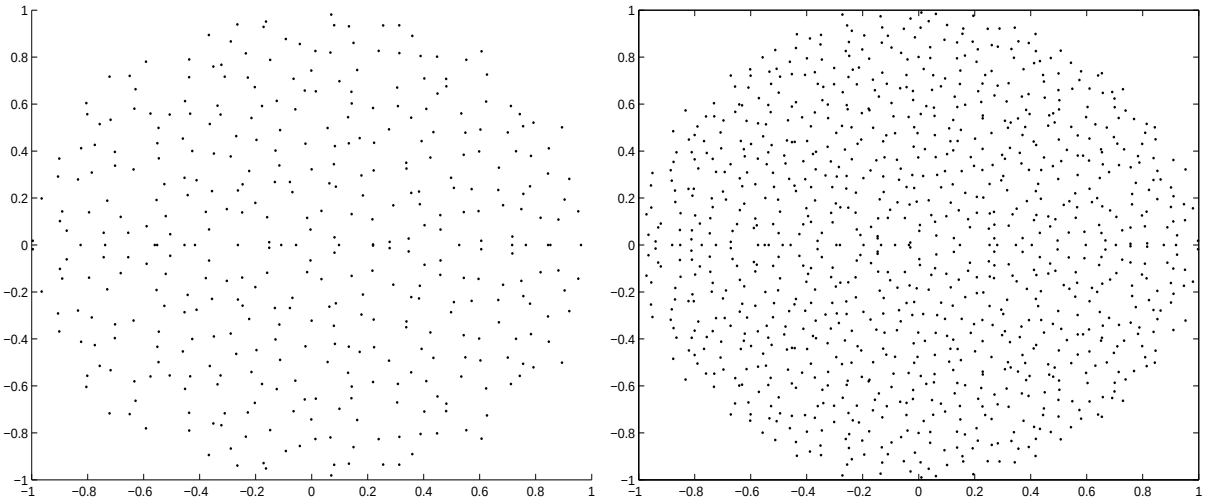

Abbildung 39: Eigenwertspektrum gestaucht durch $\sqrt{n}$ zweier gaussverteilt gezogener Matrizen mit n = 400 (links) und n = 1000 (rechts)

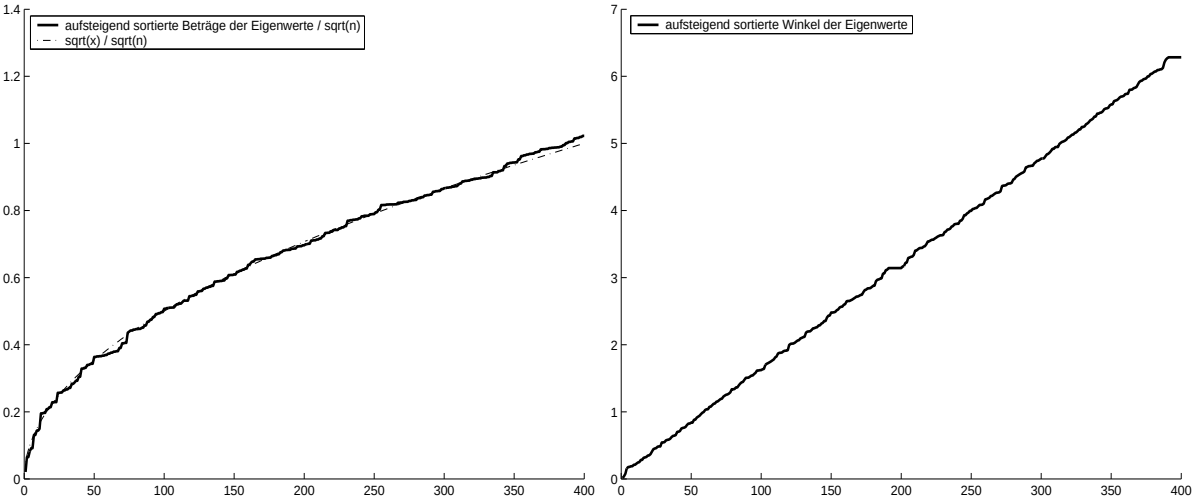

Abbildung 40: links: aufsteigend sortierte Beträge gestaucht durch $\sqrt{n}$ (entspricht der Dichte $\sim x^2$); rechts: Winkel ($\omega_v \in [0, 2\pi]$) der Eigenwerte einer gausschen Zufalls-Matrix mit n = 400

8.6 Software

Ich habe alle Simulationen mit dem numerischen Mathemtikprogramm MATLAB ausgeführt. Insgesamt habe ich mein Simulations-Framework auf knapp ein dutzend Funktionen beschränkt. Dabei verlief der Programmablauf für Experimente zur Berechnung der memory Kapazität nach diesem Muster:

```
function getMC(netType, neuronCount, noiseVariance)
    matrix = constructNet(netType, neuronCount);
    maxShift = bigNumber;
    for shift = 0 to maxShift
        if (doRegressionOnKernel)
            outWeights = regressionOnKernel(matrix, shift);
        else
            [input, wantedOutput] = getInputOutput(shift);
            outWeights = regressionOnStochasticExperiment(matrix, input, wantedOutput, noiseVariance);
        endIf
        if (getMcsFromKernel)
            kernel = getKernel(matrix, outWeight);
            mc[s] = kernel[shift];
        else
            [input, wantedOutput] = getInputOutput(shift);
            realOutput = getTestOutput(matrix, outWeights, input, noiseVariance);
            mc[s] = getMcs(wantedOutput, realOutput);
        endIf
    endFor
    MC = sum(mc);
    return MC;
end


% Errechnung der Ausgangsgewichte durch Regression auf den Zustandvektoren im Experiment mit stochastischen Input-Output-Paaren
function regressionOnStochasticExperiment(matrix, input, wantedOutput, noiseVariance)
    initLength = myInitLength;
    trainLength = myTrainLength;
    runLength = initLength + trainLength;
    currentState = (0, 0, ..., 0);
    for x = 1 to runLength
        currentState = transferFunction(matrix * currentState + input[x]);
        if (x > initLength)
            logs[x - initLength] = currentState;
        endIf
    endFor
    outWeights = linearRegression(logs, wantedOutput);
    return outWeight;
end


% Errechnung der Ausgangsgewichte durch Regression auf dem Kernel function regressionOnKernel(matrix, shift)
    kernelLength = bigNumber;
    for x = 1 to kernelLength
        logs[x] = pow(matrix, x - 1) * (1, 1, ..., 1);
    endFor
    wantedOutput = (0, 0, ..., 0, 1, 0, ..., 0); %die 1 ist an der Stelle 'shift'
    outWeights = linearRegression(logs, wantedOutput);
    return outWeight;
end
```

Dabei ist zu beachten, dass sowohl Rauschen als auch Nicht-Linearitäten nur im langwierigen Experiment mit stochastischem Input verwendet werden können. Im linearen Fall ohne Rauschen kann das Experiment viel schneller durch die Regression auf den Kernel und dem Auslesen des $mc(s)$ aus dem Kernel erfolgen.

Literatur

[1] W. Maass, T. Natschläger, H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation, in press, 2002.

[2] W. Maass, R. Legenstein, H. Markram. A new approach toward vision suggested by biologically realistic neural Microcircuit Models. BMCV 2002, Tübingen

[3] Herbert Jaeger, Short term memory in echo state networks. GMD Report 152, GMD - German National Research Institute for Computer Science, 2002.

[4] Herbert Jaeger, The echo state approach to analysing and training recurrent neural networks. GMD Report 148, GMD - German National Research Institute for Computer Science, 2001.

[5] N.C.Bertschinger, Kurzzeitspeicher ohne stabile Zustaende in rückgekoppelten neuronalen Netzen. Diplomarbeit, TH Aachen, 2002.

[6] W. Maass, T. Natschläger, Computational Models for Generic Cortical Microcircuits. TU Graz, 2003.

[7] S. Mallat, A wavelet tour of signal processing. Second Edition, Kapitel 2, 1998.

[8] R.S. Sutton, A.G. Barto, Reinforcement Learning: An introduction, 1998.

[9] Girko, V.L. Circular Law.Theory Probab. Appl. 29, 694-706, 1984.

[10] Athanasios Papoulis, Probability, Random Variables, and Stochastic Processes. Third Edition, 1991.