Modelle zur Entkopplung von Lern- und Systemdynamik bei on-line Dichteschätzung

Diplomarbeit
von
Michael Schindler

Ludwig-Maximilians-Universität
Sektion Physik
Lehrstuhl für BioMolekulare Optik
Arbeitsgruppe Theoretische Biophysik
Prof. Dr. P. Tavan

München, den 23. Oktober 2001

Inhaltsverzeichnis

Einleitung

A memory is what is left when something happens and does not completely unhappen

Edward de Bono

E.1 Neuronale Netze

Vom Standpunkt der theoretischen Biophysik aus gesehen, dienen Ganglien und Gehirne biologischer Lebewesen als Werkzeuge zur Verarbeitung von Reizmustern, die einer dynamisch variierenden Umwelt entstammen. Die Reize werden von den Sinneszellen der Haut, der Retina, Kochlea etc. aufgenommen und an das Gehirn als zentrales Nervensystem weitergeleitet. Dort findet eine Vielzahl von Verarbeitungsschritten statt, bis eine Entscheidung gefunden und an die Motorneuronen und Muskeln weitergeleitet wird. Die Leitungsvorgänge zum Gehirn hin und von ihm weg sind einfach und für den Aspekt der Informationsverarbeitung irrelevant. Wer verstehen möchte, wie diese biologische Informationsverarbeitung funktioniert, muss nach den zentralen Vorgängen im Gehirn fragen.

Wie kann man nun diese internen Verarbeitungsschritte verstehen? Mit der Beantwortung dieser Frage beschäftigen sich Wissenschaftler vieler Gebiete, natürlich Biologen und Mediziner, aber auch Mathematiker, Statistiker, Physiker und Informatiker. Letztere versuchen die Funktionsweise der Gehirne auf mathematischem oder technischem Wege zu verstehen. Sie verwenden dazu vereinfachte Modelle, die ich im folgenden *Neuronale Netze* nennen werde, im Unterschied zu *Nervennetzen*, womit biologische Gehirne aller Art gemeint sind. Das Arbeitsgebiet zur Untersuchung der Neuronalen Netze heißt *Neuroinformatik*.

Auf welche Weise findet Modellbildung in der Neuroinformatik statt? Das Gehirn wird zunächst als ein aus vielen Nervenzellen zusammengesetztes System aufgefasst, von denen jede eine Aktivierung aufweisen und diese an andere Zellen weitergeben kann. Wie stark eine Aktivierung weitergegeben wird, hängt von der Effizienz der synaptischen Verbindungen ab. Auf diese Weise wird ein bestehendes Aktivierungsmuster in das nächste überführt. Der interne Verarbeitungsprozess im Gehirn ist eine *zeitliche Abfolge von Aktivierungsmustern*.

Abbildung 1 zeigt die grundlegenden Charakteristika des in der Neuroinformatik verwendeten Standardmodells. Es besteht aus Knoten und gerichteten Kanten. Die Kno-

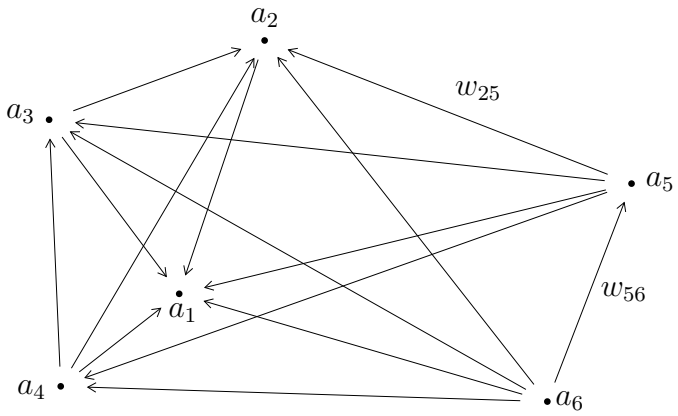


Abbildung 1: Das Standardmodell der Neuroinformatik, ein gerichteter Graph aus Knoten und gerichteten Kanten. Jeder Knoten besitzt eine Aktivität a und eventuell Parameter, die in die Neuberechnung der Aktivität nach Gleichung (1) eingehen. Jede Kante besitzt eine Übertragungsstärke w_{qr}, die ebenfalls in (1) eingeht.

ten entsprechen den Nervenzellen, die Kanten den synaptischen Verbindungen. Jeder Knoten r besitzt eine bestimmte Aktivität a_r, die er über seine Kanten mit der Stärke w_{qr} an andere Knoten weitergibt. Auf diese Weise wird die zeitliche Folge der Aktivierungsmuster berechnet.

Der *Zustand* eines Neuronalen Netzwerks aus N Knoten ist durch den Aktivitätsvektor $\mathbf{a} = (a_1, \ldots, a_N)^{\mathrm{T}}$ der Knoten zum aktuellen Zeitpunkt gegeben. Zum Modell gehört außerdem eine Vorschrift $\mathcal{A}$, die angibt, wie sich ein Zustand aus seinem Vorgänger berechnen lässt,

$$\mathcal{A}_{\boldsymbol{\theta}}\colon\ \mathbb{R}^N \to \mathbb{R}^N\colon\ \mathbf{a}(t) \mapsto \mathbf{a}(t{+}1), \tag{1}$$

wobei t die diskretisierte Zeitvariable bezeichnet. Die *Update*-Regel (1) hängt von der Wahl der *Netzwerkparameter* $\boldsymbol{\theta}$ ab, was durch den entsprechenden Index angedeutet ist. Zu den Parametern $\boldsymbol{\theta}$ der Abbildung $\mathcal{A}_{\boldsymbol{\theta}}$ gehören natürlich die Verbindungsstärken w_{rq} der Kanten, aber, je nach Modellbildung, noch weitere.

Ein Neuronales Netz ist also ein mathematisches Modell aus drei Elementen, dem zugrundeliegenden gerichteten Graphen, den Aktivitäten $\mathbf{a}$ und der parameterabhängigen Abbildung $\mathcal{A}_{\boldsymbol{\theta}}$,

$$\mathrm{NN} := \big(\mathcal{G}, \mathbf{a}, \mathcal{A}_{\boldsymbol{\theta}}\big). \tag{2}$$

McCulloch & Pitts-Netzwerke

Das erste Modell für ein solches Netzwerk war dasjenige von McCulloch & Pitts (1943). Ein Knoten r ist dabei ein *logisches Schwellwertelement* mit zwei möglichen Zuständen (Ruhe- oder Aktivierungszustand, $a_r = 0$ oder 1). Es hat eine Reihe von Eingangsleitungen (*afferente* Verbindungen) und eine Ausgangsleitung (*efferentes Axon*), die verschiedene nachgeschaltete Knoten kontaktieren kann. In jedem Rechenschritt ergibt sich sein Zustand aus dem Vergleich der anliegenden Aktivitäten mit einem Schwellenwert s_r,

$$a_r(t+1) = \Theta\Big(\sum_q w_{rq} a_q(t) - s_r\Big) := \begin{cases} 1 & \text{falls } \sum_q w_{rq} a_q(t) > s_r, \\ 0 & \text{sonst.} \end{cases} \tag{3}$$

Der Schwellenwert s_r ist mithin einer der zusätzlich zu den w_{qr} zu spezifizierenden Parameter aus der Menge $\boldsymbol{\theta}$.

Feed-forward-Netzwerke

Alle in der vorliegenden Arbeit verwendeten Netzwerke sind *feed-forward*-Schichtennetze. Allgemein sind sie dadurch definiert, dass sich ihre Knoten so in Untermengen einteilen lassen, dass die Aktivitätsberechnung in einer Menge lediglich die Aktivitäten einer weiteren Menge benötigt. Die Mengen sind auf diese Weise *hintereinandergeschaltet*. Abbildung 2 zeigt den Graphen eines einfachen *feed-forward* Netzes aus zwei Schichten, welches nach Rosenblatt (1958) als *Perzeptron* bezeichnet wird. Die Aktivitäten $\mathbf{a}$ der unteren Schicht hängen nur von denen der oberen Schicht, $\mathbf{x}$, ab. Die Aktivitäten $\mathbf{x}$ errechnen sich aus Eingaben, die nicht als Teil des Netzwerkes angesehen werden. Sie sind *Reize*, die dem Netz von außen präsentiert werden.

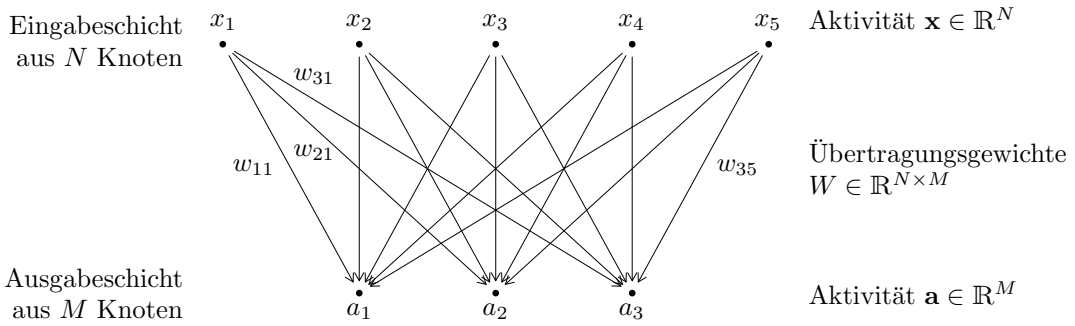


Abbildung 2: Das Perzeptron: ein *feed-forward*-Netz aus zwei Knotenschichten ohne laterale Verschaltung. Es gibt nur Kanten von allen oberen zu allen unteren Knoten. Eingezeichnet sind auch die Aktivitäten $\mathbf{x}$ und $\mathbf{a}$ der Knoten sowie die Übertragungsgewichte W der Kanten.

E.2 Lernende Neuronale Netze

Das Netzwerk aus Abb. 1 berechnet seine Aktivitäten immer und immer wieder, bis ein stationäres Aktivitätsmuster oder ein Grenzzyklus gefunden wurde. *Feed-forward* Netze propagieren jeden Reiz auf der Eingabeschicht einmal durch alle nachgeschalteten Schichten. Sie beide und jedes andere Neuronale Netz, das der Definition (2) entspricht, sind darauf angewiesen, die *richtigen* Parameter $\boldsymbol{\theta}$ zu besitzen, die dem Problem angepasst sind, zu dessen Lösung das Netz eingesetzt werden soll. Es stellt sich nun die Frage, wie die Verschaltungen und die Parameter derart bestimmt werden können, dass ein Netz eine gegebene Aufgabe der Informationsverarbeitung löst.

Das Auffinden der passenden Parameter $\boldsymbol{\theta}$ bezeichnet man als *Lernen*. Eine Vorschrift, nach der die Parameter sukzessive verändert werden, bis sich eine sinnvolle, d. h. problemangepasste Verschaltung herausbildet, nennt man *Lernregel*. Ein lernendes, oder *adaptives* Neuronales Netzwerk besteht also zusätzlich aus dem Satz seiner Parameter $\boldsymbol{\theta}$ und einer Lernregel $\mathcal{P}$, die parametrisch vom Zustand des Netzwerkes abhängt.

$$\begin{aligned} \mathrm{ANN} := \big(\mathcal{G}, \mathbf{a}, \boldsymbol{\theta}, \mathcal{A}_{\boldsymbol{\theta}}, \mathcal{P}_{\mathbf{a}}\big), \quad \text{mit} \\ \mathbf{a} \in \mathbb{R}^N, \quad \boldsymbol{\theta} \in \mathbb{R}^d, \\ \mathcal{A}_{\boldsymbol{\theta}} \colon\ \mathbb{R}^N \to \mathbb{R}^N \colon\ \mathbf{a}(t) \mapsto \mathbf{a}(t{+}1), \\ \mathcal{P}_{\mathbf{a}} \colon\ \mathbb{R}^d \to \mathbb{R}^d \colon\ \boldsymbol{\theta}(t) \mapsto \boldsymbol{\theta}(t{+}1). \end{aligned} \tag{4}$$

Das Perzeptron

Der erste Vorschlag für ein adaptives Neuronales Netz stammt von Rosenblatt (1958). Sein *Perzeptron* besteht, wie in Abb. 2 dargestellt, aus zwei Knotenschichten und einer Schicht von Kanten. Die Vorschrift für die Bestimmung der Aktivitäten a_r ist ähnlich der von McCulloch & Pitts,

$$\mathcal{A}_W \colon\ \mathbf{x} \mapsto a_r(\mathbf{x}) = \sum_{i=1}^{N} w_{ri} x_i, \quad r \in \{1, \ldots, M\}. \tag{5}$$

Als Antwort auf Reize $\mathbf{x}_\alpha \in \mathbb{R}^N$ aus einem Datensatz $\mathcal{X} = \{\mathbf{x}_\alpha \,|\, \alpha = 1, \ldots, T\}$ soll das Perzeptron mit seinen Aktivitäten $a_r(\mathbf{x}_\alpha)$ einen zugehörigen Satz von *Sollwerten* $y_{r\alpha}$ *approximieren*. Es muss dazu seine Parameter so anpassen, dass die Vektoren $\mathbf{a}(\mathbf{x}_\alpha) \in \mathbb{R}^M$ und $\mathbf{y}_\alpha \in \mathbb{R}^M$ einander immer näher kommen. Bei jedem Reiz $\mathbf{x}$ wird ihr Abstand bestimmt und die Parameter so angepasst, dass ihr Abstand kleiner wäre, würde erneut derselbe Reiz präsentiert. Aus dieser Überlegung ergibt sich die sogenannte *Deltaregel*

$$\mathcal{P}_{\{\mathbf{a},\mathbf{x}\}} \colon\ \mathbf{w}_r(t{+}1) = \mathbf{w}_r(t) + \varepsilon\,(y_r - a_r)\,\mathbf{x}, \quad r \in \{1, \ldots, M\}, \tag{6}$$

in der $\varepsilon \in [0, 1]$ der *Lernparameter* ist. Eine formale Begründung für die *Deltaregel* führt eine Fehlerfunktion E ein, die unter Variation der Parameter $\boldsymbol{\theta}$ minimiert werden soll. E ist das euklidische Abstandsquadrat zwischen der Soll- und der tatsächlichen Ausgabe,

$$E(W, \mathbf{x}) := \frac{1}{2} \sum_{r=1}^{M} \big(y_r - \mathbf{w}_r^{\mathrm{T}}\,\mathbf{x}\big)^2. \tag{7}$$

Die Lernregel (6) führt dann einen Gradientenabstieg auf E durch,

$$\mathbf{w}_r(t{+}1) = \mathbf{w}_r(t) - \varepsilon\,\nabla_{\mathbf{w}_r} E, \tag{8}$$

$$-\nabla_{\mathbf{w}_r} E = (y_r - a_r)\,\mathbf{x}. \tag{9}$$

Das Lernziel W^* ist erreicht, wenn die Lernregel (6) stationär geworden ist, wenn also die Gradienten $\nabla_{\mathbf{w}_r} E(W^*, \mathbf{x})$ für alle Reize $\mathbf{x}$ bei festem W^* verschwinden. Abbildung 3 versucht, anhand eines einfachen Beispiels zu zeigen, dass es ein solches W^*, für das

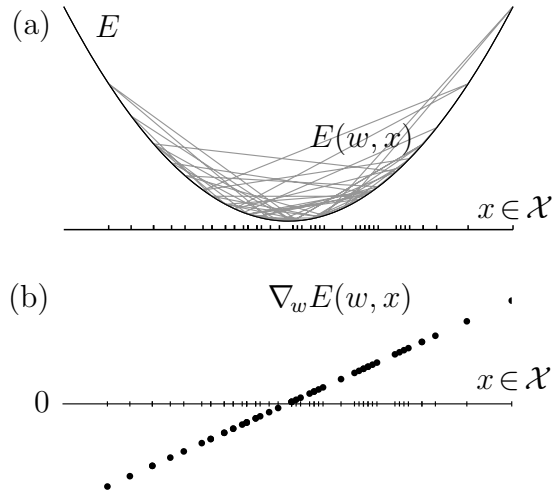


Abbildung 3: Ein Fehlerfunktional E und Fehlerwerte für sequentiell präsentierte Reize x. In (a) sind die verschiedenen nacheinander auftretenden Punkte $(x, E(w,x))$ mit einer grauen Linie verbunden. In (b) sieht man den Satz der dazugehörenden Gradienten $\nabla_w E(w,x)$. Die Gradienten sind von Null verschieden, haben im gegebenen Beispiel aber etwa den Mittelwert Null.

alle Gradienten gleichzeitig verschwinden, in der Regel nicht gibt. Es kann aber ein W^* geben, für das der Mittelwert der Gradienten verschwindet,

$$\mathbf{0} = \langle \nabla_{\mathbf{w}_r} E(W^*, \mathbf{x}) \rangle_{\mathcal{X}} \quad \forall r. \tag{10}$$

Man sieht also, dass als Lernziel des Perzeptrons nur eine statistische Größe in Frage kommt. Statt $E(W, \mathbf{x})$ aus Gleichung (7) muss nun der Mittelwert von $E(W, \mathbf{x})$ auf dem Datensatz $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ minimiert werden,

$$E(W) := \frac{1}{2T} \sum_{t=1}^{T} \sum_{r=1}^{M} \big(y_r - \mathbf{w}_r^{\mathrm{T}}\, \mathbf{x}(t)\big)^2. \tag{11}$$

Daraus ergibt sich sofort die *gemittelte* Lernregel

$$\mathbf{w}_r(t{+}1) = \mathbf{w}_r(t) + \varepsilon \, \left\langle \big(y_r - \mathbf{w}_r^{\mathrm{T}}\, \mathbf{x}\big)\, \mathbf{x} \right\rangle_{\mathcal{X}}, \tag{12}$$

deren stationärer Zustand durch (10) gegeben ist.

Es bieten sich nun zwei Möglichkeiten für die Formulierung des Perzeptron-Algorithmus und seiner Lernregel an. Beide sind in Abbildung 4 veranschaulicht.

- Die erste Variante wurde schon in Gleichung (6) eingeführt. Jeder Punkt $\mathbf{x}$ wird einzeln verarbeitet, es wird eine kleine Änderung der Parameter gemacht, und zwar als *sequentieller stochastischer* Gradientenabstieg auf der Fehlerfunktion $E(W)$ aus (11),

  $$\mathbf{w}_r(t{+}1) = \mathbf{w}_r(t) + \frac{\varepsilon}{T} \nabla_{\mathbf{w}_r} E(W(t), \mathbf{x}(t{+}1)). \tag{13}$$

  Diese Version heißt *sequentiell*, da sie die Reize $\mathbf{x}$ nacheinander verarbeitet. Sie heißt *stochastisch*, da die Punkte $\mathbf{x}$ zufällig aus dem Datensatz $\mathcal{X}$ gezogen werden. Daher besteht die Bewegung der Parameter $\mathbf{w}_r$ aus vielen kleinen Schritten, die eine mehr oder minder zufällige Richtung haben, und deren Weite durch den Lernparameter ε/T bestimmt ist. Die mittlere Richtung der Parameterbewegung ist aber parallel zum Gradienten von $E(W)$.

– In der folgenden Gleichung (14) wird nicht nach jedem $\mathbf{x}$ angepasst, sondern es werden zunächst alle zum Datensatz gehörenden Gradienten bei festem Parametersatz gemittelt,

$$\mathbf{w}_r(t+1) = \mathbf{w}_r(t) + \varepsilon \, \langle \nabla_{\mathbf{w}_r} E(W(t), \mathbf{x}) \rangle_{\mathcal{X}} \, . \tag{14}$$

Die T kleinen Schritte aus (13) werden hier also zu einem großen Schritt zusammengefasst. Diese Version heißt *batch*-Algorithmus. Sie führt zu glatteren Gradientenabstiegskurven auf E (vgl. Abb. 4), die dadurch aber auch leichter in lokalen Extrema von E steckenbleiben.

In beiden Algorithmen wird eine *Mittelung* der Gradienten $\nabla_{\mathbf{w}} E$ durchgeführt. In der *batch*-Version explizit, in der *sequentiell-stochastischen* Version durch die um $1/T$ verkleinerte Schrittweite.

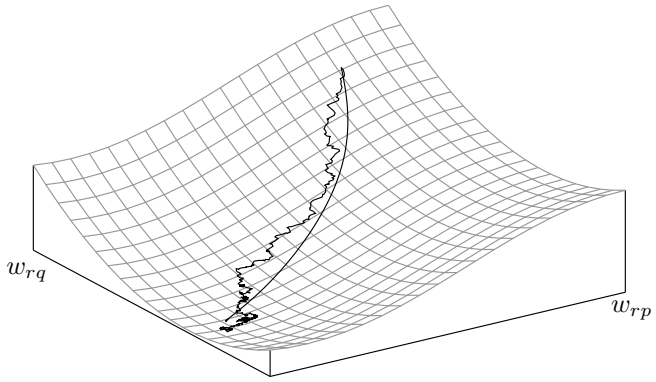


Abbildung 4: Gewöhnlicher und stochastischer Gradientenabstieg. Beide Trajektorien konvergieren in das Minimum des Fehlerfunktionals E, die eine glatt, die andere bekommt nicht nur die Schankungen der Eingabereize $\mathbf{x}$ mit, sondern sieht deswegen auch Gradienten $\nabla_{\mathbf{w}} E(W, \mathbf{x})$ in der Umgebung der glatten Trajektorie.

Mittelung bei adaptiven Neuronalen Netzwerken

Die Eigenschaft, dass die gesehenen Reize, also die Aktivitäten auf der Eingabeschicht, nur als Mittelwerte in die Adaptation der Lernparameter eingehen, ist wesentlich für das zuverlässige Konvergieren des Lernalgorithmus. Sie ist nicht nur beim vorgestellten Perzeptron zu finden, sondern bei allen bekannten Lernregeln für Neuronale Netze. An (13) und (14) wird deutlich, dass sich die Argumentation leicht auf alle Lernregeln erweitern lässt, die einen Gradientenabstieg durchführen. Sie gilt ebenfalls für diverse Formen des Hebb'schen Lernens, das im Rest der Arbeit ausschließlich verwendet wird. Die Mittelungseigenschaft findet sich somit insbesondere beim Hopfield-Modell (Hopkins, 1982), dem Backpropagation-Algorithmus (Rumelhart et al. 1986), dem Kohonen-Netz (Kohonen, 1982), diversen Assoziativspeichern (siehe Ritter et al. 1992, auch für kurze Beschreibungen der anderen Verfahren) und verwandten Algorithmen.

E.3 Wie lernen Neuronale Netze in veränderlichen Umwelten?

Einige der oben genannten Algorithmen gibt es sowohl in *batch*- als auch in sequentieller Formulierung. Es stellt sich nun die Frage, ob beide Lernszenarien die gleichen Konvergenzeigenschaften haben, und wenn ja, unter welchen Bedingungen. Die meisten Algorithmen werden in ihrer *batch*-Version untersucht, da sich dann ihr Konvergenzverhalten wesentlich leichter formulieren und beweisen lässt (siehe z. B. Benveniste et al. 1990; Ritter et al. 1992 und Heskes et al. 1993). Die so gefundenen Aussagen lassen sich nur dann auf sequentielles Lernen übertragen, wenn die Trajektorie des stochastischen Gradientenabstiegs eine leicht verrauschte Version der glatten Kurve ist. Für die beiden Trajektorien in Abb. 4 würde das bedeuten, dass sich die sequentielle immer in einer Umgebung der deterministischen Abstiegskurve befinden müsste. Dies kann jedoch nicht immer garantiert werden. Ein prototypisches Beispiel, bei dem es nicht der Fall ist, findet sich schon beim allereinfachsten ANN, das nun zur Erläuterung der auftretenden Probleme diskutiert werden soll und das in Abbildung 5 skizziert ist.

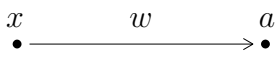


Abbildung 5: Das einfachste zwei-Schichten-ANN. Der Eingabeknoten hat die Aktivität x, der Ausgabeknoten a. Das Signal wird mit der Stärke w propagiert.

Das einfachste adaptive Neuronale Netzwerk

Das ANN aus Abb. 5 hat nur einen Eingabeknoten, eine Kante und einen Ausgabeknoten. Es folge der Lernregel

$$\mathcal{P}_x\colon\ w(t{+}1) = w(t) + \varepsilon(x_t - w(t)). \tag{15}$$

Die Form der Verarbeitungsabbildung $\mathcal{A}_w$ ist an dieser Stelle unwichtig, da sie in die angenommene Lernregel nicht eingeht. Man sieht sofort, dass der stochastisch stationäre Zustand w^* der Lernregel (15) bei

$$\langle x - w^* \rangle_{\mathcal{X}} = 0, \quad \text{oder} \quad w^* = \langle x \rangle_{\mathcal{X}} \tag{16}$$

erreicht ist. Der optimale Wert von w ist demnach der globale Mittelwert des Datensatzes $\mathcal{X}$. Dieses adaptive Neuronale Netz führt also eine Mittelwertsbildung durch.

Wie verhält sich die *batch*-Version des Algorithmus? Für sie gilt der Mittelwert von (15), was sich zusammen mit (16) zu der Lernregel

$$\mathcal{P}_x\colon\ w(t{+}1) = w(t) + \varepsilon(w^* - w(t)) \tag{17}$$

reduziert. Man erkennt sofort die exponentielle Konvergenz auf der Zeitskala $1/\varepsilon$,

$$\begin{aligned} w(t) - w^* &= \big(w(0) - w^*\big)\,(1-\varepsilon)^t, \quad \text{und für kleine } \varepsilon \\ &\approx \big(w(0) - w^*\big)\,\exp(-\varepsilon t). \end{aligned} \tag{18}$$

Gleitende Mittelung von randomisierten Daten

Kann diese Konvergenzaussage auch für den sequentiellen Fall gefunden werden? Abbildung 6 zeigt zwei Lernbeispiele mit der Regel (15). In Abb. 6a wurde bei jedem Lernschritt ein zufälliger Punkt aus dem Datensatz $\mathcal{X}$ gezogen, der in der folgenden Abb. 6b in seiner ursprünglichen Abfolge dargestellt ist. Dadurch wird der Datenstrom *randomisiert*, und aufeinanderfolgende Punkte sind statistisch voneinander unabhängig. Die sequentielle Mittelung, die (15) auf dem randomisierten Datenstrom durchführt, wird auch *gleitende* Mittelung genannt. Dieser Namensgebung liegt die Vorstellung zugrunde, dass ein Mittelungsfenster über die Datensequenz gleitet, und so den jeweils aktuellen Wert von w bestimmt. Wenn man (15) immer wieder in sich selbst einsetzt,

$$\begin{aligned} w(t) \equiv w_t &= (1-\varepsilon)\ w_{t-1} + \varepsilon x_{t-1} \\ &= (1-\varepsilon)^2 w_{t-2} + \varepsilon(1-\varepsilon)\ x_{t-2} + \varepsilon x_{t-1} \\ &= (1-\varepsilon)^3 w_{t-3} + \varepsilon(1-\varepsilon)^2 x_{t-3} + \varepsilon(1-\varepsilon) x_{t-2} + \varepsilon x_{t-1} \\ &\ \ \vdots \\ &= (1-\varepsilon)^t w_0 \quad + \varepsilon \sum_{i=1}^{t} (1-\varepsilon)^{t-i} x_{i-1}, \end{aligned} \tag{19}$$

dann sieht man an der Gewichtung $(1-\varepsilon)^{t-i}$, dass kurz zurückliegende Punkte stärker zur Summe beitragen als lang zurückliegende. Die Gewichtung im Mittelungsfenster beginnt also beim aktuellen Zeitpunkt t mit ε und fällt nach vergangenen Zeiten hin exponentiell ab. Die Breite $\tau \approx 1/\varepsilon$ des Mittelungsfensters ist die Dauer des *Gedächtnisses* des ANN.

Gleitende Mittelung von zeitlich korrelierten Daten

In Abb. 6b wurde mit dem gleichen Algorithmus die zugrunde liegende sogenannte *on-line* Datensequenz gemittelt. Sie wurde von einem stationären Markovprozess generiert, der zwischen zwei verrauschten Zuständen hin- und herschaltet. Aufeinanderfolgende Punkte haben also eine durch den Markovprozess definierte zeitliche Korrelation. Diese äußert sich dadurch, dass das System eine gewisse Anzahl von Zeitschritten in ein- und demselben Zustand verharrt. Die erwartete Verweildauer ist für ein stationäres Markovsystem konstant und stellt seine *Systemzeitskala* dar.

Die Trajektorie w'' der on-line Mittelung in Abb. 6b wurde mit derselben Lernregel berechnet wie w in Abb. 6a. Durch die zeitlichen Korrelationen wird sie immer wieder zu dem aktuellen Systemzustand „gezogen." Ein quasistatischer Zustand wird so lange gelernt, bis das gleitende Mittelungsfenster kaum noch Daten der vergangenen Systemzustände enthält. Es ist zu kurz, um außer dem aktuellen Zustand auch noch seinen Vorgänger zu überdecken, also gleiten vergangene Zustände aus seinem Gedächtnis. Dadurch, dass das ANN den aktuellen Zustand *gelernt* hat, musste es den letzten *vergessen*. Es befindet sich im Widerstreit zwischen *Lernen* und *Vergessen*. Das Mittelungsfenster, also das Gedächtnis des ANN, hat eine bestimmte typische Länge (hier $1/\varepsilon$). Sie ist die Zeitskala des Lerners. Das Markovsystem dagegen hat gewisse

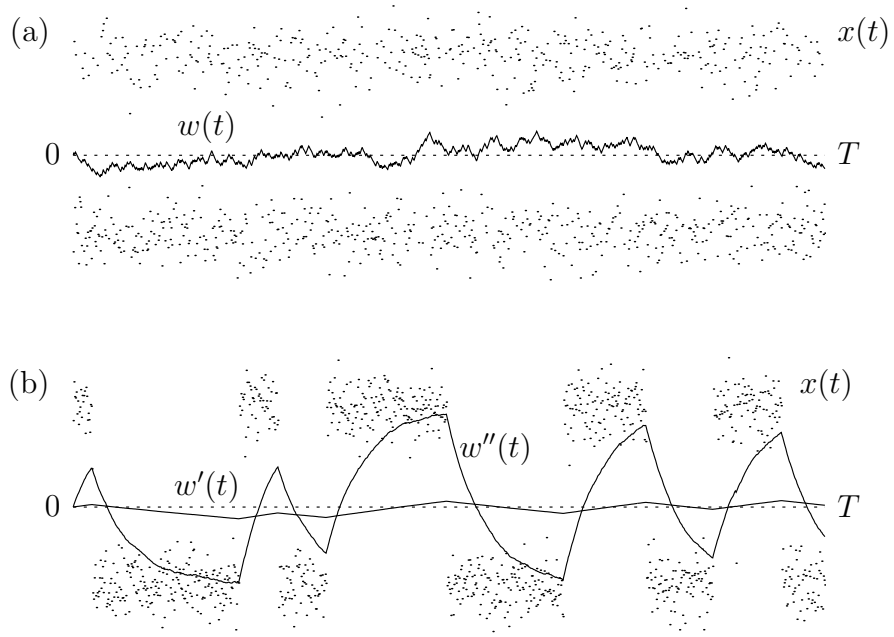


Abbildung 6: Gleitende Mittelwerte nach der Lernregel (15), aufgetragen als Funktion der Zeit. In (a) werden *randomisierte* Daten gemittelt. Man sieht deutlich die Abweichungen vom globalen Mittelwert $\langle x \rangle_{\mathcal{X}}$ (gepunktet eingezeichnet), der Parameter w entfernt sich jedoch nie weit von ihm. In (b) sind zwei gleitende Mittelwerte einer *on-line* Datensequenz eingezeichnet, die sich in der Wahl von ε unterscheiden. Für sehr kleines ε sieht man eine einigermaßen glatte Kurve $w'(t)$, die sich wie in (a) nicht weit von $\langle x \rangle_{\mathcal{X}}$ entfernt. Dazu ist jedoch notwendig, dass ε wesentlich kleiner als in (a) gewählt wird. Für das gleiche ε wie in (a) ergibt sich im on-line-Fall die stark gezackte Kurve $w''(t)$, die sich immer wieder weit vom globalen Mittelwert $\langle x \rangle_{\mathcal{X}}$ entfernt.

Verweildauern in seinen Zuständen. Das ANN vergisst vergangene Zustände erst dann, wenn neue Systemzustände so lange verweilen, dass sein Gedächtnis die alten nicht mehr behalten kann. Hier wird also die Dauer des Gedächtnisses mit der Zeitskala des Systems verglichen. Der Effekt des Lernens auf Kosten von vorher Gelerntem tritt also auf, wenn Lern- und Systemzeitskala von derselben Größenordnung sind. Ich möchte ihn als *Kopplung der Lerndynamik an die Systemdynamik* bezeichnen.

Die Trajektorie w' in Abb. 6b hat einen viel kleineren Lernparameter ε (hier 1/20 des ursprünglichen Wertes). Dadurch ist ihr Mittelungsfenster länger und kann die verschiedenen Systemzustände noch gut mitteln. Die Kopplung der Lerndynamik an die Systemdynamik kann hier nicht stattfinden, da die Lernzeitskala wesentlich größer als die des Systems ist. Die Trajektorie w' weist deshalb wieder ähnlich kleine Abweichungen vom globalen Mittelwert $\langle x \rangle_{\mathcal{X}}$ auf wie die des randomisierten Lernens.

Das Problem des on-line Lernens

Jedes Lebewesen befindet sich in einer physikalischen Umwelt, in welcher die Zeit stetig voranschreitet. Alle Beobachtungen, die es von seiner Umgebung macht, müssen notwendigerweise sequentiell sein. Die Gegenstände der Beobachtung weisen eine gewisse zeitliche Konstanz auf. Ein einzelner Ton in einem Klavierkonzert von Bach hat eine

Dauer. Die akustischen Wellen, die im Ohr des Zuhörers ankommen, verändern sich eine Weile lang nicht, sie sind derart zeitlich korreliert, dass sie zusammen als Ton gehört werden. Zwischen den einzelnen Tönen des Stückes gibt es natürlich ebenfalls Korrelationen, die durch die Partitur vorgegeben sind. Das ganze Stück ist also voll von zeitlichen Korrelationen. Insgesamt sind alle Beobachtungen, die ein Lebewesen in seiner Umwelt macht, zeitlich korreliert. On-line erfahrene – d. h. sequentielle zeitkorrelierte – Reize bilden also die ursprüngliche und natürliche Umwelt von biologischen lernenden Systemen. Neuronale Netze, die ja als Modelle für biologische Reizverarbeitung gedacht sind, sollten in der gleichen Weise on-line lernen können.

Ein Mensch, der sich ein Klavierkonzert von Bach anhört, mag in der Lage sein, sich die Tonfolgen zu merken. Je länger er hinhört, umso mehr lernt er. Auch wenn es sich um die Brandenburgischen Konzerte handelt, die schier endlos sind, muss er trotzdem nichts von dem vergessen, was ihm vorher bekannt war. Er vermag zu lernen, *ohne* dabei alles andere zu vergessen.

Das einfachste ANN kann das nicht, wie im letzten Abschnitt vorgeführt wurde. Es befindet sich im dauernden Widerstreit zwischen Lernen und Vergessen. Je mehr Neues es lernt, umso mehr Altes muss es vergessen, und zeitliche Korrelationen im Datenstrom können dazu führen, dass es alles bis auf den aktuellen Systemzustand vergisst. Es würde also ein „Experte für die Brandenburgischen Konzerte," könnte sich aber sonst an nichts erinnern. Der Gradientenabstieg in Abb. 4 befindet sich in einem ähnlichen Dilemma. Geometrisch gesprochen, können zeitliche Korrelationen im Datenstrom dazu führen, dass sich die sequentielle Trajektorie weit von der optimalen glatten *batch*-Kurve entfernt. Wenn sie sich so weit entfernt, dass sie schließlich in ein anderes Minimum der Fehlerfunktion $E(W)$ läuft, muss man das Lernverfahren als gescheitert betrachten. Besonders bei nichtlinearen Lernverfahren, bei denen sich $E(W)$ stark mit dem Parameter W verändert, kann eine solche Situation leicht auftreten. Das Lernverfahren wird somit instabil. Dies ist das *Problem des on-line Lernens*, dem jedes bekannte adaptive Neuronale Netzwerk bei sequentieller Parameteranpassung ausgesetzt ist, denn es ist eine direkte Folge von gleitender Mittelwertsbildung. Für dieses Problem hat die Neuroinformatik bislang keine Lösung gefunden.

Aufgabe der vorliegenden Arbeit ist die Behebung dieses Problems, soweit sie in einer Diplomarbeit durchführbar ist. Zuerst wird das on-line Lernproblem ausführlich formuliert, anschließend wird ein Ansatz zu seiner Lösung vorgeschlagen.

Alle Untersuchungen werden dabei an einem sehr vielseitigen adaptiven Neuronalen Netz und dem dazugehörigen Lernverfahren, dem *multivar*-Algorithmus, durchgeführt. Dieser Algorithmus wurde von Kloppenburg & Tavan (1997) entwickelt und von Albrecht et al. (2000) ausführlich diskutiert. Das zugehörige ANN besitzt eine durchgängige mathematisch/statistische Interpretation, die es als selbstorganisierten *Maximum-likelihood*-Approximator für die Verteilungsdichte des präsentierten Datensatzes ausweist. Die Dichteschätzung beruht auf lokaler räumlicher Mittelwertsbildung. Der Algorithmus kann, wie wir sehen werden, ebenfalls als *Clusteringalgorithmus* aufgefasst werden, wenn man die Dichteschätzung auf unterschiedlichen räumlichen Skalen durchführt. Dabei bekommt man einen hierarchischen Aufspaltungsprozess von Netz-

werkparametern. An diesen Aufspaltungen lässt sich das Problem der on-line Mittelung sehr gut demonstrieren.

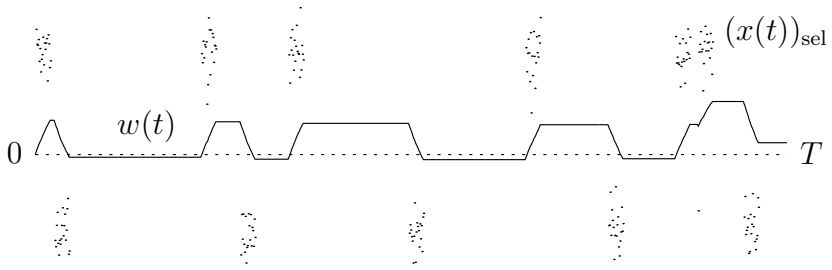


Abbildung 7: Der Lernprozess aus Abbildung 6b, angewandt auf abgekürzte Datenfolgen. Gezeigt sind die selektierten Daten $(\mathbf{x}(t))_{\mathrm{sel}}$, die gelernt werden, und der Parameter $w(t)$ als Funktion der Zeit.

Datenpunkte ignorieren

Die Idee, die zur Lösung des on-line Problems führt, ist denkbar einfach. In Abb. 6b wird deutlich, dass das Problem der on-line Mittelung auf – im Vergleich zur Dauer des Gedächtnisses des ANN – zu lange Verweildauern des Systems in einem Zustand zurückzuführen ist. Die gelernten Daten sind eine Weile lang quasistationär. Man kann das verhindern, indem man den Lernprozess abbricht, sobald die Daten redundant werden. Dazu ist notwendig, einen *Filter für irrelevante Reizdaten* einzuführen. Dieser kann aber nicht starr sein, denn ob Daten redundant sind, also für den Lernprozess irrelevant, kann nur am jeweiligen Zustand des Lerners selbst festgestellt werden. Der Filter muss sich also selbst zusammen mit den Netzwerkparametern adaptiv an die präsentierten Daten anpassen. Wie ich in meiner Diplomarbeit zeigen möchte, kann ein solches Modell für einen *selbstreferentiellen Selektionsprozess* aus einem einfachen Verhaltensmuster der Meeresschnecke *Aplysia californica*[1] abgeleitet werden. Dieses einfache Lebewesen ignoriert nach einer Weile Reizungen am Kiemen, wenn sie häufig wiederholt werden (Habituation). Wenn sich die Reizumgebung jedoch schnell verändert, beispielsweise durch einen zusätzlichen Schlag auf den Kopf oder Schwanz der *Aplysia*, wird die Habituation sofort wieder aufgehoben (Sensitivierung).

Wenn wir dieses Verhaltensmuster auf das einfache Beispiel in Abb. 6b übertragen, so erhalten wir, ähnlich wie in Abbildung 7, eine Abkürzung der gelernten Datensequenzen. Durch ein noch zu spezifizierendes Kriterium soll der Lernprozess unterbrochen werden, wenn das Verhalten der Datensequenz stagniert. Falls es gelingt, ein solches Kriterium anzugeben, so erwarten wir das in Abb. 7 skizzierte Ergebnis. Wie gewünscht, entfernt sich der Parameter w auch bei kurzer Gedächtnisdauer nicht mehr weit von seinem Optimalwert. Wie ein solches Abbruchkriterium formuliert werden

[1] Das Nervensystem der *Aplysia* ist eines der am besten untersuchten im Tierreich. Für die weitgehende Aufklärung seiner Bestandteile und Funktionsweise wurde im Jahr 2000 der Nobelpreis an Eric Kandel verliehen.

kann, insbesondere, wie es ohne zusätzliche Modellannahmen auskommt, wird in Kapitel 4 besprochen.

E.4 Gliederung der Arbeit

Kapitel 1 beschäftigt sich mit denjenigen Eigenschaften des *multivar*-Algorithmus, die für die vorliegende Arbeit relevant sind. In Abschnitt 1.1 werden zunächst seine mathematischen Eigenschaften benannt, die ihn zum *Maximum-likelihood*-Dichteschätzer und Clusteringalgorithmus machen. In Abschnitt 1.2 wird dann die Interpretation des Algorithmus und der zugehörigen Netzwerkstruktur in biologischem Kontext vorgenommen. Die im Algorithmus verwendeten Begriffe, die vorher nur als mathematische Objekte definiert werden, sollen mit einem physiologienahen Hintergrund belegt werden, auch um ihnen die nötige Anschaulichkeit zu geben. Neben den Begriffen werden die Näherungen erläutert, die der neuronalen Modellbildung zugrundeliegen. Dadurch soll einigermaßen klar werden, welche Aspekte der biologischen Reizverarbeitung relativ detailliert simuliert und welche nur grob approximiert werden.

Die Charakterisierung des on-line Lernproblems anhand des Clusteringprozesses wird in Kapitel 2 vorgenommen. Sie wird zunächst an zwei Beispielen vorgeführt, und danach durch die analytische Behandlung von Spezialfällen genauer gefasst. Die Untersuchungen zeigen, dass sowohl die räumlichen als auch die zeitlichen Skalen der Mittelwertsbildung im *multivar*-Algorithmus zu diesem Effekt beitragen. Wie stark sich beide Skalen dabei gegenseitig beeinflussen, ist in Abschnitt 2.3 beschrieben.

Kapitel 4 führt das Prinzip des *selbstreferentiellen Lernens* ein, bei dem anhand des aktuellen Netzwerkzustands entschieden wird, ob und wie stark ein Datenpunkt in den Lernprozess eingehen soll. Das gewählte Prinzip ist, wie oben erwähnt, durch den Gewöhnungs- und Sensitivierungsprozess des Kiemenreflexes der Meeresschnecke *Aplysia californica* inspiriert, der in Kapitel 3 erklärt wird. Selbstreferentielles Lernen stellt einen möglichen Ansatz dar, die Probleme beim on-line Lernen zu vermeiden. Es wird hier anhand des ANTS (*Algorithm for Neural Timescale Separation*) vorgeführt, der eine Modifikation des *multivar*-Algorithmus ist. Seine Eigenschaften werden in Abschnitt 4.2 erklärt und in Appendix C bewiesen.

Kapitel 1

Grundlagen

Das in dieser Arbeit verwendete *multivar*-Netzwerk (MVNN) ist aus zwei Schichten von Knoten aufgebaut, ähnlich wie das Perzeptron aus Abb. 2, nur hat es mehr Netzwerkparameter und kompliziertere Lernregeln. Es dient der Verarbeitung hochdimensionaler Datensätze $\mathcal{X} \subset \mathbb{R}^d$ ($d \gg 1$). Als Modell der biophysikalisch orientierten Neuroinformatik, welche die Phänomene der biologischen Reizverarbeitung zu verstehen versucht, zielt die Verwendung einer Struktur aus zwei Schichten auf die effektive Modellierung eines Eingabe-/Ausgabe-Verhaltens ab.

Die Signale der biologischen Reizverarbeitung bestehen typischerweise aus vielen Teilsignalen, denn der sensorische Apparat, mit dem Lebewesen ihre Umwelt beobachten, ist aus vielen Sensoren aufgebaut, von denen jeder auf einen kleinen Teilaspekt seiner physikalischen Umgebung spezialisiert ist. Ein solches multisensorisches biologisches System ist beispielsweise die Retina, die aus etwa 10^8 lichtempfindlichen Stäbchen und Zapfen besteht. Ihre Erregungssignale werden von der Retina an den primären visuellen Kortex weitergeleitet. Wenn man die einzelnen Teilsignale in einem großen Vektor zusammenfasst, bekommt man so viele Einträge, wie es lichtempfindliche Zellen auf der Retina gibt. Diesen kann man als Datenvektor und als Punkt im 10^8-dimensionalen Raum verstehen, in dem die biologische „Datenverarbeitung“ stattfindet. Entsprechend kann man den zur visuellen Beobachtung einer bewegten Szene gehörenden Datenstrom als Trajektorie in diesem hochdimensionalen Raum auffassen.

Biologische Reizverarbeitungssysteme sind aber nicht die einzigen Fälle, in denen hochdimensionale Datenströme verarbeitet werden müssen. Auch bei technischen Anwendungen ist man oft mit vielen Dimensionen konfrontiert. Ein Beispiel ist die Steuerung einer Anlage für Rückstandsverbrennung in einem Chemiewerk der Firma Wacker, die von Albrecht (1999) entwickelt wurde. Auch dort stammen die hochdimensionalen Daten von der Vielzahl der Sensoren, die den zeitlichen Verlauf von Druck, Temperatur, Zusammensetzung der Brennstoffe, ihre Fließgeschwindigkeit etc. anzeigen.

Die Untersuchung solcher Datensätze ist schwierig. In zwei oder drei Dimensionen kann durch die Betrachtung einfacher Graphiken festgestellt werden, ob sie Strukturen oder Muster beinhalten. Bei höheren Dimensionen ($d > 3$) wird diese Methode der Datenanalyse unmöglich, da die Projektion in eine zweidimensionale Abbildung im allgemeinen die Datenstrukturen verbirgt. Es wird also nötig, den Datensatz mit mathemati-

schen Methoden zu analysieren, um auf diese Weise seine wichtigsten Eigenschaften, funktionale Abhängigkeiten bzw. Korrelationen zwischen den Koordinatenwerten, zu finden. Eine Vielzahl hochdimensionaler Beobachtungen muss auf einen kleinen Satz von Modellparametern reduziert werden, mit denen der Datensatz hinreichend genau beschrieben werden kann. Diese Aufgabe heißt *Dimensionsreduktion*.

Bei der Dimensionsreduktion mit der Absicht, die wichtigsten Eigenschaften eines Datensatzes zu finden, oder ihn eventuell besser darstellen zu können, handelt es sich um ein Standardproblem der klassischen Statistik. Eine bevorzugte Methode ist die Approximation der Verteilungsdichte p des gegebenen Datensatzes durch eine Mischung von Basisfunktionen, die durch wenige Parameter vollständig beschrieben sind (Duda & Hart, 1973). Diese sollten auch in hohen Dimensionen einfach handhabbar sein, weshalb man oft Gauß'sche Glockenkurven (GF) in d Dimensionen als Basisfunktionen verwendet, die sich in eindimensionale Funktionen faktorisieren lassen,

$$\mathrm{GF}(\mathbf{x};\mathbf{c},\sigma) = \prod_{i=1}^{d} \mathrm{GF}(x_i;c_i,\sigma) \quad \forall \mathbf{x} \in \mathbb{R}^d. \tag{1-1}$$

Dabei ist $\mathbf{c} \in \mathbb{R}^d$ das Zentrum der Funktion und σ die Standardabweichung. Beim Übergang zu multivariaten Normalverteilungen ändert sich bis auf eine Drehung des Koordinatensystems nichts an der Faktorisierung.

Die parametrische Dichteschätzung mit einer gewichteten Summe multivariater Normalverteilungen wurde von Duda & Hart (1973) ausführlich beschrieben. Im folgenden Abschnitt möchte ich diese Methode und den *multivar*-Algorithmus von Kloppenburg & Tavan (1997) vorstellen, welcher die Schätzung durch sequentielle Parameteradaption durchführt. Erst nach der rein mathematischen Betrachtung werde ich in Abschnitt 1.2 genauer auf die biophysikalischen Interpretationen eingehen, die das *multivar*-Netzwerk als effektives Modell für das Ein-/Ausgabe-Verhalten von Nervenzellverbänden ausweisen.

1.1 Dichteschätzung mit einer Mischung multivariater Normalverteilungen

Das Ziel einer Dichteschätzung besteht darin, für einen gegebenen Datensatz $\mathcal{X} \subset \mathcal{M}$ im sogenannten *Merkmalsraum* $\mathcal{M} \subset \mathbb{R}^d$ eine Approximation $\hat{p}$ der zugrundeliegenden Verteilungsdichte p zu finden. In praktischen Anwendungen enthält der Datensatz immer endlich viele Punkte,[1]

$$\mathcal{X} := \{\mathbf{x}_\alpha \in \mathcal{M} \mid \alpha = 1,\ldots,T\}. \tag{1-2}$$

[1] Es handelt sich um ein Approximationsproblem mit *unvollständigen* Daten. Die Verteilungsdichte, nach welcher der Datensatz generiert wurde, kann nicht beliebig genau bestimmt werden. Statt der zugrundeliegenden Verteilungsdichte könnte man auch die Häufigkeit der Datenpunkte schätzen. Es handelte sich dann strenggenommen um ein Interpolations- und kein Dichteschätzungsproblem. Im folgenden wird kein Unterschied mehr zwischen diesen beiden Sichtweisen gemacht.

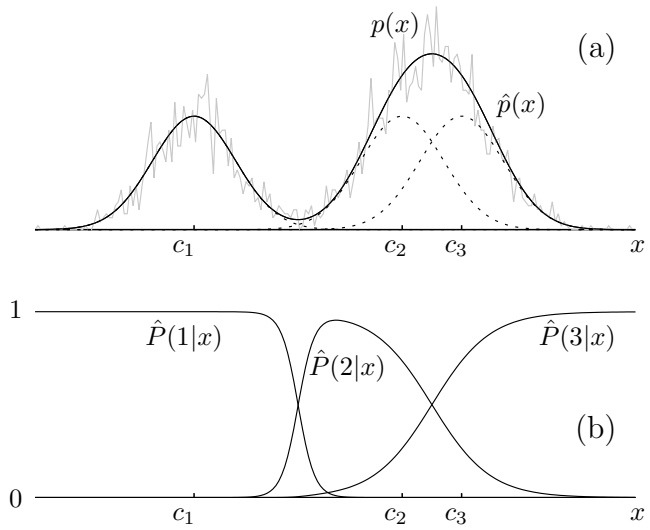


Abbildung 8: (a) Approximation der Verteilungsdichte p (graue Linie) durch eine Mischung $\hat{p}$ aus drei Normalverteilungen (durchgezogene Linie). Jede der drei Glockenkurven (gepunktet eingezeichnet) mittelt Datenpunkte x nur in ihrem Zuständigkeitsbereich, der entsprechend Gleichung (1-10) in (b) eingezeichnet ist. Die drei Funktionen in (b) bilden eine unscharfe Partitionierung des Merkmalsraumes. Sie lassen sich an jeder Stelle zu 1 aufaddieren.

Hier soll als Approximation $\hat{p}$ eine gewichtete endliche Mischung von M multivariaten Normalverteilungen verwendet werden,

$$\hat{p}(\mathbf{x};\boldsymbol{\theta}) := \sum_{r=1}^{M} \frac{\hat{P}_r}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_r|^{1/2}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_r)^{\mathrm{T}}\boldsymbol{\Sigma}_r^{-1}(\mathbf{x}-\mathbf{c}_r)\Big)\,, \tag{1-3}$$

die als Parameter einen Satz von Zentren $\mathbf{c}_r$, von Kovarianzmatrizen $\boldsymbol{\Sigma}_r$ und Gewichtungen $\hat{P}_r$ hat. Diese sind in der Variablen $\boldsymbol{\theta}$ zusammengefasst,

$$\begin{aligned} \boldsymbol{\theta} &:= (\theta_1, \ldots, \theta_M), \\ \theta_r &:= \{\mathbf{c}_r, \boldsymbol{\Sigma}_r, \hat{P}_r\}. \end{aligned} \tag{1-4}$$

Es soll also eine *parametrische* Dichteschätzung durchgeführt werden. Das Ergebnis einer solchen Schätzung in einer Dimension könnte wie in Abbildung 8a aussehen, in der mit drei Normalverteilungen approximiert wurde.

Das *Maximum-likelihood*-Prinzip

Bei der Suche nach dem besten Modell für p handelt es sich um ein Optimierungsproblem der Parameter $\boldsymbol{\theta}$. Zur Lösung dieser Aufgabe führt das bekannte *Maximum-likelihood* (ML)-Kriterium (Duda & Hart, 1973), das im folgenden kurz beschrieben werden soll. Wäre zur Generierung des Datensatzes $\mathcal{X}$ statt der unbekannten Verteilungsdichte p die Approximation $\hat{p}$ zugrundeglegt worden,[2] dann wäre die Wahrschein-

[2] Wahrscheinlichkeiten und W-Dichten, die *geschätzte Modelle* sind, tragen im folgenden ein Hütchen (ˆ). Sie hängen alle parametrisch von $\boldsymbol{\theta}$ ab. Um die Notation zu vereinfachen, wird $\boldsymbol{\theta}$ an den meisten

lichkeitsdichte jedes einzelnen Punktes $\mathbf{x}\in\mathcal{M}$ gerade $\hat{p}(\mathbf{x};\boldsymbol{\theta})$. Die Wahrscheinlichkeitsdichte des ganzen Datensatzes $\mathcal{X}$ wäre entsprechend

$$\hat{p}(\mathcal{X};\boldsymbol{\theta}) = \prod_{\alpha=1}^{T} \hat{p}(\mathbf{x}_\alpha;\boldsymbol{\theta}), \tag{1-5}$$

vorausgesetzt, die Einzelereignisse $\mathbf{x}_\alpha$ seien statistisch voneinander unabhängig. Der Wert dieser Wahrscheinlichkeitdichte sollte hoch sein, da ja genau dieser Datensatz gemäß der Verteilungsdichte p gezogen wurde, für die $\hat{p}$ eine Approximation darstellt. Da keine weiteren Informationen über p zur Verfügung stehen, sollte sie den größten überhaupt mit dem Modell $\hat{p}$ erreichbaren Wert annehmen. Genau dies besagt das ML-Kriterium. Die *Likelihood* der Parameter $\boldsymbol{\theta}$ auf dem Datensatz, definiert als

$$L(\boldsymbol{\theta}\|\mathcal{X}) := \hat{p}(\mathcal{X};\boldsymbol{\theta}) \tag{1-6}$$

muss unter Variation von $\boldsymbol{\theta}$ maximiert werden. Statt L kann natürlich auch jede streng monoton steigende Funktion von L maximiert werden. Überlegungen aus der Informationstheorie (s. Kullback & Leibler, 1951) und der statistischen Physik (Jaynes, 1957) legen die Verwendung des Logarithmus nahe. Die sogenannte *Log-likelihood* der Parameter $\boldsymbol{\theta}$ auf dem Datensatz ist dann

$$\begin{aligned} l(\boldsymbol{\theta}\|\mathcal{X}) &= \ln L(\boldsymbol{\theta}\|\mathcal{X}) = \ln\Big(\prod_{\mathbf{x}\in\mathcal{X}} L(\boldsymbol{\theta}\|\mathbf{x})\Big) \\ &= \ln\Big(\prod_{\mathbf{x}\in\mathcal{M}} \hat{p}(\mathbf{x};\boldsymbol{\theta})^{h(\mathbf{x})}\Big) = \sum_{\mathbf{x}\in\mathcal{M}} h(\mathbf{x}) \ln \hat{p}(\mathbf{x};\boldsymbol{\theta}) \\ &= T \sum_{\mathbf{x}\in\mathcal{M}} p(\mathbf{x}) \ln \hat{p}(\mathbf{x};\boldsymbol{\theta}), \end{aligned} \tag{1-7}$$

wobei $h(\mathbf{x})$ die Häufigkeit eines Punktes $\mathbf{x}$ im Datensatz und $p(\mathbf{x}) := h(x)/T$ ist. Einen noch besseren Einblick in die Bedeutung dieser Größe bekommt man, wenn man die „Informationstheoretische Entropie“ $\big(-\sum_{\mathbf{x}} p(\mathbf{x}) \ln p(\mathbf{x})\big)$, die ein Maß für das Unbekannte im Datensatz ist (Jaynes, 1963), addiert. Durch die Relation

$$\sum_{\mathbf{x}\in\mathcal{M}} p(\mathbf{x}) \ln \frac{p(\mathbf{x})}{\hat{p}(\mathbf{x};\boldsymbol{\theta})} \geq 0 \tag{1-8}$$

mit Gleichheit genau dann, wenn $\hat{p}=p$, wird diese Größe zu einer Art „Abstand“ der beiden Verteilungsdichten p und $\hat{p}$, auch *Kullback-Leibler-Distanz* oder *relative Entropie* genannt. Sie ist jedoch kein echtes Abstandsmaß, da sie nicht symmetrisch ist. Erst in dieser Form wird die Übertragung des Ausdrucks (1-7) in einen kontinuierlichen Merkmalsraum sinnvoll, da er erst hier invariant unter Koordinatentransformation wird (Jaynes, 1963).[3] Mit der Distanz (1-8) oder mit

$$E(\boldsymbol{\theta}) := \int_{\mathcal{M}} p(\mathbf{x}) \, \ln \frac{p(\mathbf{x})}{\hat{p}(\mathbf{x};\boldsymbol{\theta})} \, d\mathbf{x} \tag{1-9}$$

Stellen weglassen.

[3] Im folgenden wird nur noch die allgemeinere Notation mit Integralen verwendet.

ist nun eine geeignete Fehlerfunktion $E(\boldsymbol{\theta})$ für die Aufgabe der Dichteschätzung gefunden. Von Dersch (1995) wurde bewiesen, dass die Minimierung von E tatsächlich zu einer Approximation der Dichte p selbst führt, und nicht zu einer verzerrten Dichte p^γ, wie das etwa beim Kohonenalgorithmus in einer Dimension der Fall ist (vgl. Ritter, Martinetz & Schulten, 1992; Dersch, 1995; ein Beispiel für diesen Algorithmus wird auch an späterer Stelle in Abbildung 18 gegeben). In Appendix C.1 wird das gleiche Ergebnis noch einmal mit Hilfe der Variationsrechnung gezeigt, um diese Methode für spätere Zwecke zu etablieren. Zwanglos ergibt sich außerdem die Stabilität des stationären Zustandes.

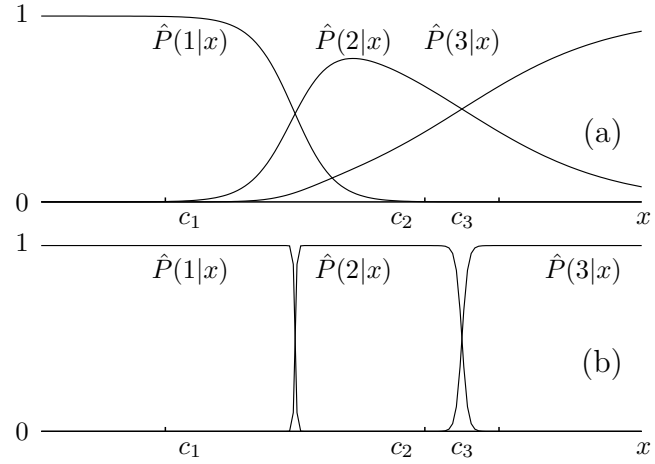


Abbildung 9: Sehr unscharfe (a) und scharfe (b) Partitionierung des Merkmalsraumes aus Abb. 8.
In (a) ist die Standardabweichung der Normalverteilungen 1.42-mal so groß wie die in Abb. 8, in (b) nur 0.28-mal.

Unscharfe Partitionierung

Der Ansatz, die Schätzung $\hat{p}$ als Mischung von Normalverteilungen zu schreiben, führt (mehr oder minder künstlich) M Klassen ein, denen die Datenpunkte zugeordnet werden. Legt man wieder die Approximation $\hat{p}$ der Verteilungsdichte zugrunde, so wird die Wahrscheinlichkeit, dass ein Punkt $\mathbf{x}$ von der Normalverteilung r „generiert“ wurde, nach dem Bayes'schen Satz berechnet,

$$\hat{P}(r|\mathbf{x};\boldsymbol{\theta}) = \hat{P}_r\,\hat{p}(\mathbf{x}|r;\theta_r)\Big/\sum_{r'=1}^{M}\hat{P}_{r'}\,\hat{p}(\mathbf{x}|r';\theta_{r'}) \tag{1-10}$$

$$= \frac{\hat{P}_r\,\hat{p}(\mathbf{x}|r;\theta_r)}{\hat{p}(\mathbf{x};\boldsymbol{\theta})}\,, \tag{1-11}$$

mit den multivariaten Normalverteilungen

$$\hat{p}(\mathbf{x}|r;\theta_r) := \exp\Big(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_r)^{\mathrm{T}}\boldsymbol{\Sigma}_r^{-1}(\mathbf{x}-\mathbf{c}_r)\Big)\;\Big/\;\big((2\pi)^d|\boldsymbol{\Sigma}_r|\big)^{1/2}\,. \tag{1-12}$$

Die bedingten Wahrscheinlichkeiten $\hat{P}(r|\mathbf{x})$ können als *Zuständigkeiten* der Gauß'schen Glockenkurven aus Abb. 8a für den Punkt $\mathbf{x}$ verstanden werden. Die Funktionen $\mathbf{x} \mapsto \hat{P}(r|\mathbf{x};\boldsymbol{\theta})$ sind in Abb. 8b dargestellt. Dadurch, dass sie jeden Punkt zu einer

Glockenkurve zuordnen, diese Zuordnung aber nur mit einer gewissen Wahrscheinlichkeit geschieht, bilden sie eine *unscharfe Partitionierung* von $\mathcal{M}$ (Kloppenburg, 1996). Je nach Varianz der Normalverteilungen ist diese Partition unterschiedlich scharf. In Abbildung 9a ist eine sehr unscharfe Partitionierung mit großen Varianzen dargestellt, die Einteilung in Abb. 9b dagegen ist schon fast eine scharfe Mengeneinteilung des Merkmalsraumes.

Entwicklung nach lokalen Momenten

Die notwendigen Bedingungen für die Minimierung von E lauten für die Zentren, Kovarianzmatrizen und Gewichte, die gemeinsam die Parameter des Modells bilden,

$$\mathbf{c}_r = \left\langle \hat{P}(r|\mathbf{x};\boldsymbol{\theta})\,\mathbf{x} \right\rangle_{\mathcal{X}} \Big/ \left\langle \hat{P}(r|\mathbf{x};\boldsymbol{\theta}) \right\rangle_{\mathcal{X}}, \tag{1-13}$$

$$\boldsymbol{\Sigma}_r = \left\langle \hat{P}(r|\mathbf{x};\boldsymbol{\theta})\,(\mathbf{x}-\mathbf{c}_r)(\mathbf{x}-\mathbf{c}_r)^{\mathrm{T}} \right\rangle_{\mathcal{X}} \Big/ \left\langle \hat{P}(r|\mathbf{x};\boldsymbol{\theta}) \right\rangle_{\mathcal{X}} \quad \text{und} \tag{1-14}$$

$$\hat{P}_r = \left\langle \hat{P}(r|\mathbf{x};\boldsymbol{\theta}) \right\rangle_{\mathcal{X}}. \tag{1-15}$$

Sie definieren die optimale Approximation für p und wurden von Kloppenburg (1996) als *lokale Momentenentwicklung* bezeichnet. Diesen Begriff möchte ich kurz erläutern. Zunächst soll nur mit einer einzigen Normalverteilung geschätzt werden. Dann gilt $\hat{P}(r|\mathbf{x};\boldsymbol{\theta}) = 1$, und die drei Gleichungen vereinfachen sich zu

$$\mathbf{c} = \left\langle \mathbf{x} \right\rangle_{\mathcal{X}}, \tag{1-16}$$

$$\boldsymbol{\Sigma} = \left\langle (\mathbf{x}-\mathbf{c}_r)(\mathbf{x}-\mathbf{c}_r)^{\mathrm{T}} \right\rangle_{\mathcal{X}} = \left\langle \mathbf{x}\mathbf{x}^{\mathrm{T}} \right\rangle_{\mathcal{X}} - \mathbf{c}\mathbf{c}^{\mathrm{T}}, \tag{1-17}$$

$$P = 1. \tag{1-18}$$

Man sieht, dass $\mathbf{c}$ der Schwerpunkt der Verteilung und $\boldsymbol{\Sigma}$ die Kovarianzmatrix ist. Beide sind globale Mittelwerte von Potenzen der betrachteten Zufallsgröße X. Solche Mittelwerte von Potenzen – oder Erwartungswerte, falls die gesamte Verteilung zur Verfügung steht – nennt man *Momente* von X (Bandelow, 1989),

$$\left\langle\!\left\langle X \right\rangle\!\right\rangle_m := \mathcal{E}\left(X^m\right) = \int_{\mathcal{M}} p(\mathbf{x})\, x_i^m \cdots x_d^m \, d\mathbf{x}, \tag{1-19}$$

mit der Ordnung m. Man sieht, dass P das Moment nullter Ordnung und $\mathbf{c}$ dasjenige erster Ordnung ist. Für $\boldsymbol{\Sigma}$ in (1-17) benötigen wir eine Erweiterung der Definiton auf *gemischte zentrierte Momente* der Ordnung $m = m_1 + \cdots + m_d$,

$$\left\langle\!\left\langle X_i \right\rangle\!\right\rangle_{(m_1 \cdots m_d)} := \mathcal{E}\Big(\prod_{i=1}^{d} \big(X - \mathcal{E}(X)_i\big)^{m_i}\Big). \tag{1-20}$$

Die verschiedenen Momente zweiter Ordnung bilden also die Einträge der Kovarianzmatrix $\boldsymbol{\Sigma}$. Ähnlich wie sich Funktionen in einer Taylorreihe nach Potenzen entwickeln lassen, so gibt es auch für viele Verteilungsdichten p eine Entwicklung nach zentrierten Momenten steigender Ordnung. Die Entwicklung bis zur zweiten Ordnung reicht aus,

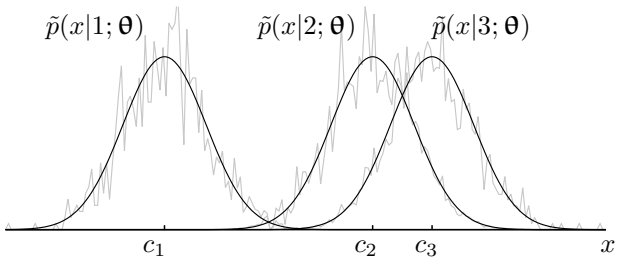


Abbildung 10: Die drei lokalen Verteilungsdichten, die nach Gleichung (1-21) von den einzelnen Normalverteilungen in Abb. 8a approximiert werden. Die Aufteilung der ursprünglichen Verteilungsdichte p durch die Partitionierung in Abb. 8b findet gerade so statt, dass jeder Teil selbst etwa die Form einer Normalverteilung hat.

um lineare Korrelationen in den Koordinatenwerten der Daten zu finden. Dazu muss die Kovarianzmatrix diagonalisiert werden. Orientiert man sich dabei hauptsächlich an den Eigenvektoren zu den größten Eigenwerten, so spricht man auch von *Hauptachsenanalyse* oder PCA (*Principal component analysis*).

Die Terme in (1-13) und (1-14), bei denen die Approximation mit mehreren Normalverteilungen durchgeführt wird, sind jedoch keine globalen Momente, sondern werden zusätzlich mit den Zuständigkeiten $\hat{P}(r|\mathbf{x};\boldsymbol{\theta})$ der einzelnen Normalverteilungen gewichtet. Somit erzeugen sie *lokale Statistiken* (Kloppenburg, 1996). Um dieses Konzept zu verstehen, stellen wir zunächst fest, dass sich für jede Komponente r des Mischungsmodells (1-3) ein zugehöriger Ausschnitt

$$\tilde{p}(\mathbf{x}|r;\boldsymbol{\theta}) := \hat{P}(r|\mathbf{x};\boldsymbol{\theta})\, p(\mathbf{x}) \,\Big/\, \big\langle \hat{P}(r|\mathbf{x};\boldsymbol{\theta}) \big\rangle_{\mathcal{X}} \quad \text{mit} \tag{1-21}$$

$$\big\langle \hat{P}(r|\mathbf{x};\boldsymbol{\theta}) \big\rangle_{\mathcal{X}} = \int\limits_{\mathcal{M}} \hat{P}(r|\mathbf{x};\boldsymbol{\theta})\, p(\mathbf{x})\, d\mathbf{x} \tag{1-22}$$

aus der Datenverteilung $p(\mathbf{x})$ angeben lässt. Das statistische Gewicht (1-22) der r-lokalen Verteilung (1-21) wird gelegentlich auch als *Load* der r-ten Komponente bezeichnet. Abbildung 10 zeigt die r-lokalen Verteilungen (1-21) für das Beispiel aus Abb. 8a als hellgraue Linien. Wie man dort sieht, werden sie durch Normalverteilungen approximiert. Dieser Approximation liegen die Näherungen

$$\hat{P}_r \approx \big\langle \hat{P}(r|\mathbf{x};\boldsymbol{\theta}) \big\rangle_{\mathcal{X}} \tag{1-23}$$

für die statistischen Gewichte $\hat{P}_r$ und

$$\hat{p}(\mathbf{x};\boldsymbol{\theta}) \approx p(\mathbf{x}) \tag{1-24}$$

für die Gesamtverteilung zugrunde; denn mit diesen Näherungen folgt aus (1-21) zunächst

$$\tilde{p}(\mathbf{x}|r;\boldsymbol{\theta}) \approx \hat{P}(r|\mathbf{x};\boldsymbol{\theta})\, \hat{p}(\mathbf{x};\boldsymbol{\theta}) \,/\, \hat{P}_r, \tag{1-25}$$

woraus durch Einsetzen der Bayes'schen Formel (1-10) unmittelbar

$$\tilde{p}(\mathbf{x}|r;\boldsymbol{\theta}) \approx \hat{p}(\mathbf{x}|r;\boldsymbol{\theta}_r) \tag{1-26}$$

folgt. Dies ist die Approximation der r-lokalen Verteilungen $\tilde{p}(\mathbf{x}|r;\boldsymbol{\theta})$ durch die Normalverteilungen $\hat{p}(\mathbf{x}|r;\boldsymbol{\theta}_r)$.

Die Definition (1-21) der r-lokalen Verteilungen $\tilde{p}(\mathbf{x}|r;\boldsymbol{\theta})$ induziert r-lokale Erwartungswerte

$$\big\langle f(\mathbf{x})\big\rangle_{r,\boldsymbol{\theta}} := \int\limits_{\mathcal{M}} f(\mathbf{x})\,\tilde{p}(\mathbf{x}|r;\boldsymbol{\theta})\,d\mathbf{x}, \tag{1-27}$$

die gemäß (1-21) durch die Partitionsfunktionen $\hat{P}(r|\mathbf{x};\boldsymbol{\theta})$ gegeben sind, d. h.

$$\big\langle f(\mathbf{x})\big\rangle_{r,\boldsymbol{\theta}} = \frac{\big\langle \hat{P}(r|\mathbf{x};\boldsymbol{\theta})\,f(\mathbf{x})\big\rangle_{\mathcal{X}}}{\big\langle \hat{P}(r|\mathbf{x};\boldsymbol{\theta})\big\rangle_{\mathcal{X}}}\,. \tag{1-28}$$

Ein Vergleich mit den Extremalitätsbedingungen (1-13) und (1-14) zeigt nun sofort, dass diese durch die ersten und zweiten r-lokalen Momente

$$\mathbf{c}_r = \big\langle \mathbf{x}\big\rangle_{r,\boldsymbol{\theta}} \quad \text{und} \tag{1-29}$$

$$\boldsymbol{\Sigma}_r = \big\langle (\mathbf{x}-\mathbf{c}_r)(\mathbf{x}-\mathbf{c}_r)^{\mathrm{T}}\big\rangle_{r,\boldsymbol{\theta}} \tag{1-30}$$

der Verteilungen $\tilde{p}(\mathbf{x}|r;\boldsymbol{\theta})$ ausgedrückt werden können. Diese beiden Gleichungen stellen Konsistenzbedingungen für die Parameter $\boldsymbol{\theta}$ dar, wobei nach (1-23) die zusätzliche Bedingung (1-15),

$$\hat{P}_r = \big\langle \hat{P}(r|\mathbf{x};\boldsymbol{\theta})\big\rangle_{\mathcal{X}}, \tag{1-31}$$

an die *Load* der r-ten Komponente gestellt werden muss.

Ziel jedes algorithmischen Verfahrens zur *Maximum-likelihood*-Dichteschätzung ist, mit einem sicher konvergierenden Verfahren die Parameter $\boldsymbol{\theta}$ so zu bestimmen, dass die Selbstkonsistenzbedingungen (1-29) bis (1-31) erfüllt werden.

1.1.1 Der *multivar*-Algorithmus

Ein *batch*-Algorithmus für den Gradientenabstieg auf $E(\boldsymbol{\theta})$ wurde von Dempster, Laird & Rubin (1977) als *Expectation-Maximization* (EM)-Algorithmus eingeführt. Seine sequentielle Version, die das Grundgerüst des *multivar*-Algorithmus darstellt, lautet (Albrecht et al. 2000)

(EM-1) Initialisiere die Parameter $\boldsymbol{\theta}(t=0)$

(EM-2) Ziehe einen zufälligen Punkt $\mathbf{x}_t$ aus dem Datensatz $\mathcal{X}$ und berechne die Zuständigkeiten der Modellkomponenten r nach dem Bayes'schen Satz (1-10)

(EM-3) Berechne die neue Schätzung durch sequentiellen stochastischen Gradientenabstieg. Verwende dabei die Update-Regeln $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \varepsilon\Delta\boldsymbol{\theta}$,

$$\Delta\mathbf{c}_r = \hat{P}(r|\mathbf{x}_t;\boldsymbol{\theta}(t))\,\big(\mathbf{x}_t - \mathbf{c}_r(t)\big), \tag{1-32}$$

$$\Delta\boldsymbol{\Sigma}_r = \hat{P}(r|\mathbf{x}_t;\boldsymbol{\theta}(t))\,\Big((\mathbf{x}_t - \mathbf{c}_r(t))\,(\mathbf{x}_t - \mathbf{c}_r(t))^{\mathrm{T}} - \boldsymbol{\Sigma}_r(t)\Big), \tag{1-33}$$

$$\Delta\hat{P}_r = \hat{P}(r|\mathbf{x}_t;\boldsymbol{\theta}(t)) - \hat{P}_r(t). \tag{1-34}$$

(EM-4) Setze die diskretisierte Zeit t auf $t+1$ und gehe zu (EM-2).

Die Lernregeln in (EM-3) haben alle die Form einer gleitenden Mittelwertsbildung, deren Prototyp,

$$\langle\vartheta\rangle^{\text{neu}} = (1-\eta)\,\langle\vartheta\rangle^{\text{alt}} + \eta\vartheta, \tag{1-35}$$

schon in (15) benutzt wurde. Dabei ist ϑ die zu mittelnde Größe, $\langle\vartheta\rangle$ ihr gleitender Mittelwert und η die sogenannte *Lernrate*. Auch dieser Algorithmus führt also, wie das einfachste ANN, eine Mittelwertsbildung durch, nur wird im Falle von (1-32) und (1-33) nicht global, sondern gemäß (1-29) und (1-30) r-lokal gemittelt. Lediglich (1-34) stellt eine globale Mittelwertsbildung dar. Aufgrund der gleitenden Mittelwertsbildungen – seien sie nun global oder lokal – ist das *multivar*-Netzwerk mit den Lernregeln (1-32) bis (1-34) eine zwanglose Verallgemeinerung des einfachsten ANN aus Abb. 5 auf lokale statt globaler Statistik. In diesen Regeln hängt die Lernrate immer auch gleichzeitig von der aktuellen Schätzung ab, was den Iterationsprozess stark nichtlinear werden lässt. Deswegen konvergiert er in vielen Fällen nicht in das globale, sondern lediglich in ein lokales Minimum von E (Kloppenburg, 1996).

Regularisierungsterme zur Stabilisierung

Dass die Lernraten $\varepsilon\hat{P}(r|\mathbf{x};\boldsymbol{\theta})$ in den Lernregeln (1-32) und (1-33) stark nichtlinear von den Parametern abhängen, zu deren Berechnung sie beitragen, macht den sequentiellen EM-Algorithmus instabil. Eine genaue Diskussion dieser Instabilitäten findet sich bei Duda & Hart (1973), Kloppenburg (1996) und Albrecht et al. (2000). An dieser Stelle soll nur einer der möglichen Mechanismen, die zu solchen Instabilitäten führen können, skizziert werden. So können beispielsweise Komponenten r der Mischung mit großer Varianz oder mit großem statistischen Gewicht $\hat{P}_r$ andere Komponenten r' verdrängen. Wie eine genauere Analyse zeigt, ist die Grundlage hierfür die *Kompetition* der Komponenten um Datenpunkte $\mathbf{x}$, die in der Normierung

$$\sum_{r=1}^{M} \hat{P}(r|\mathbf{x}) = 1 \qquad \forall\mathbf{x}\in\mathcal{M} \tag{1-36}$$

der Zuständigkeiten zum Ausdruck kommt. Wo eine Normalverteilung große Zuständigkeit hat, dort kann ihre Varianz und ihr Gewicht, auf Kosten der Varianzen und Gewichte aller übrigen Komponenten wachsen. Das Ergebnis ist in vielen Fällen, dass einige der eingesetzten Modellkomponenten aufgrund verschwindender Varianzen oder Gewichte nicht mehr zur statistischen Modellbildung beitragen.

Die Idee von Kloppenburg & Tavan (1997) zur Beseitigung dieser und anderer Instabilitäten ist die flexible Kopplung sowohl der Varianzen als auch der $\hat{P}_r$ an vorgegebene Werte. Im zugehörigen *multivar*-Algorithmus werden hierfür zunächst die Kovarianzmatrizen in Eigenwerte und orthonormale Eigenvektoren zerlegt,

$$\boldsymbol{\Sigma}_r = \sum_{i=1}^{d} \mathbf{w}_{ri}\sigma_{ri}\mathbf{w}_{ri}^{\mathrm{T}}\,. \tag{1-37}$$

Mit den Eigenwertgleichungen

$$\mathbf{w}_{ri} = \frac{1}{\sigma_{ri}^2}\Sigma_r \mathbf{w}_{ri}, \quad \sigma_{ri}^2 = \mathbf{w}_{ri}^{\mathrm{T}}\Sigma_r \mathbf{w}_{ri}. \tag{1-38}$$

können aus der Lernregel (1-33) für die Kovarianzmatrizen zwei äquivalente Regeln gewonnen werden,

$$\Delta\mathbf{w}_{ri} = \hat{P}(r|\mathbf{x}_t)\left(\frac{(\mathbf{x}_t - \mathbf{c}_r(t))\,(\mathbf{x}_t - \mathbf{c}_r(t))^{\mathrm{T}}\mathbf{w}_{ri}(t)}{\sigma_{ri}^2(t)} - \mathbf{w}_{ri}(t)\right), \tag{1-39}$$

$$\Delta(\sigma_{ri}^2) = \hat{P}(r|\mathbf{x}_t)\left(\left[(\mathbf{x}_t - \mathbf{c}_r(t))^{\mathrm{T}}\mathbf{w}_{ri}(t)\right]^2 - \sigma_{ri}^2(t)\right). \tag{1-40}$$

Die erste Gleichung hat die Form einer Richtungslernregel. Sie ist an diejenige von Oja (1982) angelehnt und führt, unabhängig von den Varianzen, eine Hauptachsenbestimmung durch (siehe dazu Rubner & Tavan, 1989; Albrecht et al. 2000). Um zu vermeiden, dass alle $\mathbf{w}_{ri}$ in dieselbe Richtung maximaler Varianz zeigen, müssen die $\mathbf{w}_{ri}$ nach jedem Lernschritt *orthogonalisiert* werden. Lässt man ferner in (1-39) den normierenden Zerfallsterm $(-\mathbf{w}_{ri})$ weg, so muss man nach jedem Lernschritt eine Gram-Schmidt-Orthonormierung (Fischer & Kaul, 1990) der Vektoren $\mathbf{w}_{ri}$ durchführen. Wie von Albrecht et al. (2000) gezeigt wurde, diagonalisieren die so gelernten $\mathbf{w}_{ri}$ die r-lokalen Kovarianzmatrizen $\left\langle(\mathbf{x}-\mathbf{c}_r)(\mathbf{x}-\mathbf{c}_r)^{\mathrm{T}}\right\rangle_{r,\boldsymbol{\theta}}$.

Der *multivar*-Algorithmus von Kloppenburg & Tavan (1997) lautet

(MV-1) Initialisiere die Parameter $\boldsymbol{\theta}(t=0)$

(MV-2) Ziehe einen zufälligen Punkt $\mathbf{x}_t$ aus dem Datensatz $\mathcal{X}$ und berechne die Zuständigkeiten nach dem Bayes'schen Satz (1-10)

(MV-3) Berechne das neue Codebuch nach den Updateregeln $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \varepsilon\Delta\boldsymbol{\theta}$, wobei

$$\Delta\mathbf{c}_r = \hat{P}(r|\mathbf{x}_t)\left(\mathbf{x}_t - \mathbf{c}_r(t) + \mathbf{R}\right) \tag{1-41}$$

mit einem kleinen Rauschen $\mathbf{R}$,

$$\Delta\mathbf{w}_{ri} = \frac{\hat{P}(r|\mathbf{x}_t)}{\sigma_{ri}^2(t)}\left(\mathbf{x}_t - \mathbf{c}_r(t)\right)\left(\mathbf{x}_t - \mathbf{c}_r(t)\right)^{\mathrm{T}}\mathbf{w}_{ri}(t) \tag{1-42}$$

$$\tag{1-43}$$

und orthonormiere die neuen Hauptachsen $\mathbf{w}_{ri}(t)$,

$$\Delta(\sigma_{ri}^2) = \hat{P}(r|\mathbf{x}_t)\left(\left[(\mathbf{x}_t - \mathbf{c}_r(t))^{\mathrm{T}}\mathbf{w}_{ri}(t)\right]^2 - \sigma_{ri}^2(t) + \mu\frac{\sigma^2 - \sigma_{ri}^2}{\langle\hat{P}(r|\mathbf{x})\rangle}\right), \tag{1-44}$$

$$\Delta\hat{P}_r = \left(\hat{P}(r|\mathbf{x}_t) - \hat{P}_r(t) + \nu\Big[\frac{1}{M} - \hat{P}_r(t)\Big]\right). \tag{1-45}$$

(MV-4) Setze die Lernparameter ε, σ, μ und ν auf neue Werte, und zwar nach vorgegebenen Funktionen $\varepsilon(t)$, $\sigma(t)$, $\mu(t)$ und $\nu(t)$

(MV-5) Zähle die diskretisierte Zeit t um eins weiter und gehe nach (MV-2).

Die Parameter μ und ν in (1-44) und (1-45) sind *Kopplungsparameter*, die verhindern, dass sich die Varianzen und die Gewichte beliebig an die Daten anpassen können, und die dafür sorgen, dass sie näherungsweise vorgegebene Werte annehmen. Wie Kloppenburg (1996) zeigte, werden bei großem μ die verschiedenen Varianzen σ_{ri}^2 näherungsweise an den vorgegebenen Wert σ^2 gebunden. Auf die gleiche Weise können sich die Gewichtungen $\hat{P}_r$ der Normalverteilungen bei großem ν nicht weit vom Vorgabewert $1/M$ entfernen. Dies sind die entscheidende Punkte am *multivar*-Algorithmus, da auf diese Weise die oben beschriebenen Instabilitäten bei der Kompetition behoben werden können.

In Schritt (MV-4) werden die Lernparameter neu berechnet. Die Kopplungsterme μ und ν werden verkleinert, wodurch die Kopplung der Varianzen und Gewichtungen schrittweise aufgehoben werden kann. Auch der Vorgabewert σ^2, der als der wahrscheinlichste aller Varianzen eingeführt wird, muss über einen großen Bereich von Werten verändert werden, wie im folgenden Abschnitt erklärt wird. Der Lernparameter ε wird ebenfalls verkleinert. Beim einfachsten ANN in der Einleitung klang schon an, dass $1/\varepsilon$ etwas mit der Dauer des Gedächtnisses zu tun hat. Durch Verkleinerung des Parameters wird die gleitende Mittelwertsbildung also immer genauer.

Durch diese Änderungen am ursprünglichen sequentiellen EM-Algorithmus erreichten Kloppenburg & Tavan (1997), dass der *multivar*-Algorithmus sowohl die Eigenschaften eines stabilen sequentiellen Dichteschätzers aufweist, als auch diejenigen eines Clusteringalgorithmus. Dieser zweite Aspekt soll im folgenden Abschnitt näher beleuchtet werden, da er für die vorliegende Arbeit von großer Bedeutung ist.

1.1.2 Clustereinteilung und σ-*Annealing*: Der *univar*-Algorithmus

Das Verfahren zur Dichteschätzung mit *multivar* kann in zwei Phasen eingeteilt werden. In der ersten Phase, dem sogenannten *univar*-Schritt, werden die Kopplungsparameter μ und ν groß und konstant gewählt. Ferner wird der Vorgabewert σ^2 für die Varianzen schrittweise, bei großem Anfangswert beginnend, verkleinert. Durch die Bindung der Varianzen σ_{ri}^2 an den Vorgabewert σ^2 gilt für die Kovarianzmatrizen nach (1-37) näherungsweise

$$\Sigma_r \approx \sigma^2 \sum_{i=1}^{d} \mathbf{w}_{ri} \mathbf{w}_{ri}^{\mathrm{T}} , \tag{1-46}$$

und wegen der Orthonormalität der Eigenvektoren $\mathbf{w}_{ri}$,

$$\Sigma_r \approx \sigma^2 I, \tag{1-47}$$

wobei I die Einheitsmatrix ist. Nach (1-47) entarten also die Komponenten (1-12) der Mischungsdichte (1-3) zu d-dimensionalen Gaußkugeln, d. h. univariaten Normal-

verteilungen, was die Namensgebung des ersten algorithmischen Teilschrittes erklärt. Aufgrund der Bindung $\sigma_{ri}^2 \approx \sigma^2$ bei großen μ werden nach (1-47) die Lernregeln (1-42) und (1-44) überflüssig, und es kann zunächst von einer Modelldichte mit univariaten Komponenten *identischer* Varianz σ^2 ausgegangen werden. Auf ähnliche Weise kann in der *univar*-Phase aufgrund der starken Kopplung der Gewichte $\hat{P}_r$ an den vorgegebenen Wert $1/M$ auf die Lernregel (1-45) verzichtet werden, wenn die $\hat{P}_r$ einfach auf

$$\hat{P}_r = \frac{1}{M} \tag{1-48}$$

gesetzt werden. Wie schon Dersch (1995) gezeigt hat, erzwingt diese Forderung die sogenannte *Load-balance*

$$\left\langle \hat{P}(r|\mathbf{x}, \boldsymbol{\theta}) \right\rangle_{\mathcal{X}} \approx \frac{1}{M}, \tag{1-49}$$

die für die Stabilität des *univar*-Algorithmus von außerordentlicher Bedeutung ist. Gleichung (1-48) besagt, dass jede Komponente der univariaten Mischungsdichte etwa gleich viele Datenpunkte repräsentieren soll.

Wegen des Wegfalls der Lernregeln (1-42) bis (1-45) im *univar*-Schritt bleibt lediglich die Lernregel (1-41) für die Zentren $\mathbf{c}_r$ der Modellkomponenten übrig. Im Verlauf dieser Arbeit werden fast alle Untersuchungen anhand nur dieser einen Lernregel durchgeführt.

Abbildung 11 zeigt in der oberen Reihe einen typischen Verlauf einer *univar*-Dichteschätzung. Hier wird eine grobe Skizze des Datensatzes gefunden, bei der sich die

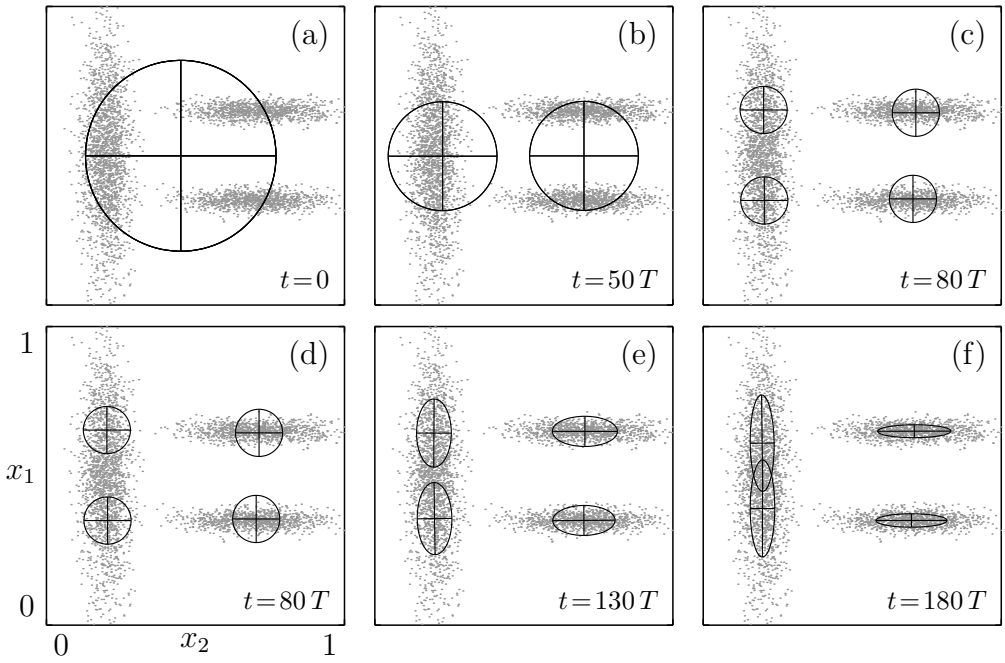


Abbildung 11: Dichteschätzungsverfahren von zweidimensionalen Daten mit *multivar*, zeitlich in der Reihenfolge von (a) bis (f). In der oberen Reihe ist die *univar*-Phase dargestellt, die zur Clustereinteilung der Daten führt. Unten trainieren die Normalverteilungen in der anschließenden Verfeinerungsphase in ihrem jeweiligen Zuständigeitsbereich die lokalen Hauptachsen und Varianzen nach.

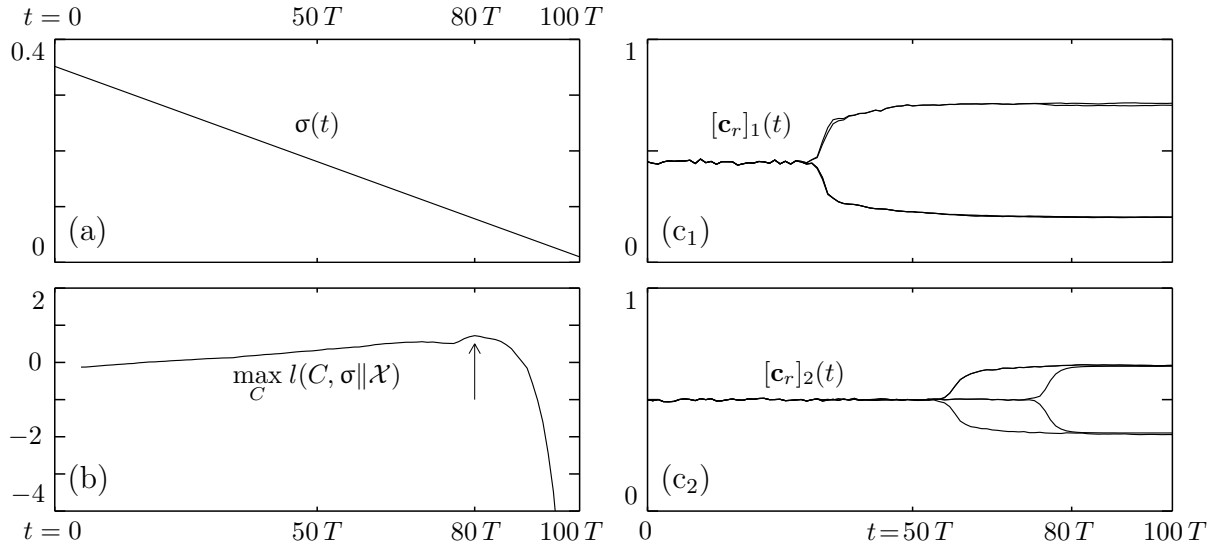


Abbildung 12: (a) Die Funktion $\sigma(t)$ der univariaten Lernphase in Abb. 11a bis c. Es werden nacheinander alle Varianzen durchprobiert. In (b) ist für jeden Zeitpunkt, also für jeden der σ-Werte, die maximale *Log-likelihood* aufgetragen, die durch Variation der Zentren $\mathbf{c}_r$ erreicht werden kann. Der größte dieser Maximalwerte liegt bei $80\,T$ (80 Iterationen über den Datensatz; Pfeil). In (c) sind die beiden Koordinaten der vier Zentren $\mathbf{c}_r$ aufgetragen. Man sieht deutlich, dass die Phasenübergänge im System plötzlich und schnell geschehen. In (c_1) ist der erste Phasenübergang zu sehen, der zwischen Abb. 11a und b abläuft. Beide Codebuchzentren brechen anschließend einzeln zu unterschiedlichen Zeitpunkten auf (c_2).

univariaten Normalverteilungen so auf dem Merkmalsraum verteilen, dass ihre Zentren $\mathbf{c}_r$ in den Häufungspunkten (*Clustern*) der Daten sitzen. Dabei wird $\sigma(t)$ nach einem vorgegebenen Schema reduziert.

In Abbildung 12a ist die lineare Funktion $\sigma(t)$ dargestellt, die im Lernprozess aus Abb. 11a bis c verwendet wurde. Für jeden Wert von σ gibt es, wenn man den Algorithmus als *Maximum-likelihood*-Dichteschätzer mit univariaten Normalverteilungen betrachtet, genau eine optimale Konstellation der Zentren $\mathbf{c}_r$. Jede Komponente sitzt etwa dort, wo sie nach der Stationaritätsbedingung (1-13) sein muss.

Die Entwicklung der für jeden Wert von $\sigma(t)$ jeweils maximierten *Log-likelihood* zeigt Abb. 12b. Wie man erkennen kann, nimmt die maximale *Likelihood* mit der Verkleinerung von $\sigma(t)$ lange Zeit zu, erreicht bei $t = 80\,T$ ein Maximum und fällt danach wieder ab. Hiermit ist klar, wie im *univar*-Algorithmus die optimale Varianz der univariaten Modelldichte bestimmt wird, nämlich einfach aus dem in Abb. 12b notierten Verlauf der Likelihoodkurve als Funktion des Führparameters σ.

Bei dem skizzierten Prozess spalten die Komponenten der univariaten Mischungsdichte aus einem zunächst vollständig entarteten Zustand (Abb. 11a) sukzessive auf (Abb. 11b und c). Dadurch wird eine Hierarchie der Zwischenzustände erzeugt. Die Abbildungen 12c beleuchten die Form dieser Aufspaltungen etwas detaillierter. Dort sind die beiden Koordinaten der Zentren $\mathbf{c}_r$ aus Abb. 11a–c als Funktionen der Zeit aufgetragen. Die Form der Aufspaltungen erinnert stark an Bifurkationen, wie sie in der Molekularfeldnäherung von Phasenübergängen zweiter Ordnung erhalten werden. Tatsächlich

kann diese Analogie, wie weiter unten noch genauer erklärt werden wird, auch formal exakt formuliert werden. Wie wir am Beispiel aus Abb. 11a–c gesehen haben, nimmt die *univar*-Phase eine Grobanalyse der Datenverteilung vor, bei der das Optimum des Führparameters σ^2 und die zugehörigen Zentren $\mathbf{c}_r$ der univariaten Modelle bestimmt werden.

Dieses Optimum wird als Ausgangspunkt der zweiten Lernphase (Abb. 11d–f) verwendet. Indem die Kopplung der σ_{ri}^2 an σ^2 durch Verkleinern von μ langsam aufgehoben und durch Iteration aller Lernregeln (1-41) bis (1-44) zusätzlich noch die lokalen Kovarianzmatrizen Σ_r der Modelle bestimmt werden, analysieren nun multivariate Normalverteilungen Orientierung und Geometrie der Cluster. Die Qualität der gleitenden Mittelwertsbildung ist dabei durch die Größe von ε bedingt. Eine ausführliche Diskussion der Rolle von ε findet sich in Abschnitt 2.1.3.

Die Lernregel (1-41) der Zentren, die im *univar*-Algorithmus zur *Maximum-likelihood*-Dichteschätzung führt, wurde schon von Rose, Gurewitz & Fox (1990) in einem etwas anderen Zusammenhang vorgeschlagen. Sie formulierten die Regel für einen *Clustering*-Algorithmus, der selbstorganisiert eine Zuordnung von Datenpunkten zu *Prototypen* durchführt. Das Verfahren dazu, das sie *fuzzy clustering* nannten, beruht auf ähnlichen unscharfen Partitionsfunktionen, wie sie in Abb. 9 gezeigt sind. Auch bei ihnen gibt es einen Parameter σ^2, der die Schärfe der Einteilung bestimmt. Für jede der Partitionen existiert ein Prototyp, welcher die zugehörige Menge von Datenpunkten repräsentieren soll. Für ihr Modell konnten Rose, Gurewitz & Fox ein mechanisch/statistisches Analogon formulieren und in der Molekularfeldnäherung lösen. Der Parameter σ^2 bekommt dabei die Bedeutung einer *Temperatur*, die von großen nach kleinen Werten „abgekühlt“ wird, um die Zuordnung der Punkte zu ihren Prototypen zu verbessern. Deshalb wird ein solches Verkleinern von σ^2 auch *Annealing* (Abkühlen) genannt. Die Autoren begründeten die Lernregel (1-41) mit einem Gradientenabstieg auf einer Funktion, die sie als freie Energie interpretieren konnten. Ihnen entging jedoch, dass die Regel in der sequentiell-stochastischen Form wie sie hier verwendet wird, ebenfalls die *Log-likelihood* eines univariaten Mischungsdichtemodells maximiert. Dieser Zusammenhang wurde von Kloppenburg & Tavan (1997) aufgedeckt, die erklärten, welche Verbindung zwischen Dichteschätzung im *Maximum-likelihood*-Sinne durch eine Mischung univariater Normalverteilungen (mit identischen Gewichten und Varianzen) und dem Problem des *fuzzy clustering* besteht. Ihr *univar*-Algorithmus kombiniert beide Sichtweisen.

Die Theorie von Rose, Gurewitz & Fox hat den Vorteil, dass sie die Form der Aufspaltungen erklärt, und zwar als Phasenübergänge mit Führparameter σ und Ordnungsparameter $(\mathbf{c}_r - \mathbf{c}_{r'})$ (Abb. 12c). Für große Werte von σ ist das *Codebuch*, wie man die Menge $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_M\}$ der Zentren auch nennt, vollständig entartet. Beim *Annealing* bricht es sukzessive auf, wenn σ kritische Werte σ_{krit} annimmt, bis seine Entartungen vollständig aufgehoben sind. Rose, Gurewitz & Fox (1990) zeigten, dass der erste Phasenübergang gerade dann geschieht, wenn σ^2 und der größte Eigenwert der Kovarianzmatrix $\Sigma_{\mathcal{X}} = \left\langle (\mathbf{x}-\mathbf{c}_r)(\mathbf{x}-\mathbf{c}_r)^{\mathrm{T}} \right\rangle_{\mathcal{X}}$ des Datensatzes, also die größte relevanten Struktur

in den Daten, gleich werden,

$$\sigma^2_{\text{krit}} = \max\{\lambda \mid \Sigma_{\mathcal{X}}\mathbf{y} = \lambda\mathbf{y}\}\,. \tag{1-50}$$

Die Richtung der Aufspaltung ist dann parallel zur dazugehörigen Hauptachse (von Abb. 11a nach b liegt sie waagerecht). Mit dieser Erkenntnis kann der Initialisierungsschritt (MV-1) des *multivar*- wie des *univar*-Algorithmus präzisiert werden. Jeder Lernprozess sollte mit so großem σ beginnen, dass der erste Aufspaltungsprozess noch nicht stattgefunden hat oder gerade stattfindet. Diese Initialisierung wurde in Abb. 11a gewählt:

(MV-1) Initialisiere alle Zentren $\mathbf{c}_r$ im Schwerpunkt des gesamten Datensatzes und mache die Varianzen so groß wie die größte Varianz des Datensatzes

$$\mathbf{c}_r = \langle \mathbf{x} \rangle_{\mathcal{X}} \quad \text{für alle } r, \tag{1-51}$$

$$\sigma^2 = \max\{\lambda \mid \Sigma_{\mathcal{X}}\mathbf{y} = \lambda\mathbf{y}\}, \quad \text{und} \quad \mu \gg 1, \tag{1-52}$$

$$\hat{P}_r = \frac{1}{M}, \quad \text{und} \quad \nu \gg 1. \tag{1-53}$$

Beim multivariaten Training passen sich die Werte der σ_{ri} datengetrieben an die Varianzen der Cluster an (Abb. 11d bis f). In diesem Sinne sind die Varianzen σ_{ri} Netzwerkparameter, für die es eine geeignete Lernregel gibt. Beim univariaten Training wird σ dagegen als Führparameter behandelt, der die Stärke der Kompetition während des Lernprozesses steuert. Die optimale Form von $\sigma(t)$ als Funktion der Zeit im *Annealing*-Prozess ist nicht durch das *Maximum-likelihood*-Prinzip bestimmt, das ja nur Aussagen über die optimalen Netzwerkparemeter $\boldsymbol{\theta}$ im stationären Zustand erlaubt. Sie hat zwar bei beliebig langsamem Training keinen Einfluss auf den stationären Endzustand, doch in allen praktisch relevanten Fällen ist wichtig, wie das genaue Abkühl-Schema aussieht.

Für das Thema dieser Arbeit, nämlich die Beschreibung der Probleme, die auftauchen, sobald der gelernte Datenstrom zeitliche Korrelationen aufweist, sind die beiden Phasen des *multivar*-Lernprozesses unterschiedlich interessant. Während der zweiten Phase passen sich die multivariaten Normalverteilungen genau an die lokale Geometrie der Datencluster an. Dazu ist notwendig, dass die Partitionierung von $\mathcal{M}$ bereits genügend scharf ist, dass also insbesondere die Phasenübergänge schon stattgefunden haben. Zeitlich korrelierte Daten haben in diesem Zustand wenig Einfluss auf das Verhalten der Parameter. Dies wird anhand von Abb. 6b in der Einleitung klar. Gäbe es – wie in Abbildung 13 – zwei Normalverteilungen, dann hätte sich jedes der Zentren auf einen der beiden Systemzustände spezialisiert, und die beiden Kurven würden sich von ihren lokalen Mittelwerten nicht mehr weit entfernen. Die Optimierung der lokalen Kovarianzmatrizen ist also ein Aspekt, bei dem eine Kopplung der Lerndynamik an die Systemdynamik von untergeordneter Bedeutung ist.

Wesentlich interessanter bei der Verwendung von on-line Daten sind dagegen die Veränderungen des Clusteringprozesses. Hier wird sich das on-line Lernproblem in aller Deutlichkeit zeigen. Der detaillierten Beschreibung der auftretenden Phänomene ist

Kapitel 2 gewidmet. Zunächst soll jedoch im folgenden Abschnitt näher auf die Verbindungen zwischen *multivar*-Algorithmus und biologischer Reizverarbeitung eingegangen werden.

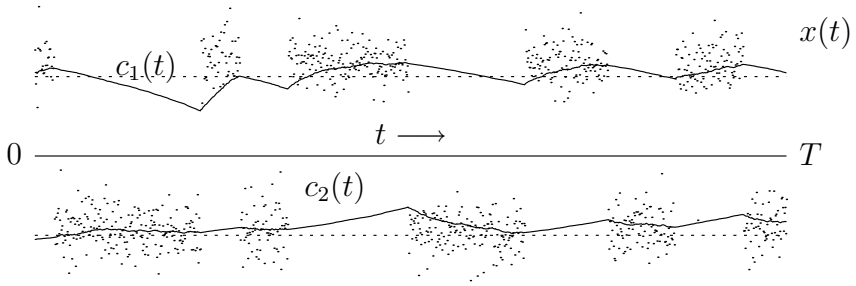


Abbildung 13: Der Lernprozess aus Abbildung 6b, diesmal nicht als Bildung eines globalen, sondern zweier lokaler Mittelwerte, deren Optimalwerte bei den gepunkteten Linien liegen. Aufgetragen sind die beiden Zentren c_r der Normalverteilungen als Funktionen der Zeit. Da die Zentren aufgrund der Kompetition fast nur ihre „eigenen“ Daten sehen, können sie kaum noch wie in Abb. 6b von den Daten des jeweils anderen Clusters angezogen werden. Der Lernprozess von *multivar* ist hier in seiner zweiten Phase gezeigt, denn die Aufspaltung in zwei voneinander entfernte Zentren hat bereits stattgefunden.

1.2 Neuronale Interpretation von *multivar*

In einigen Teilen des menschlichen Gehirns, v. a. in der Hirnrinde, liegt die Verschaltung der Nervenzellen in Schichten vor (sechs Schichten in der Groß- und drei in der Kleinhirnrinde). Dabei dienen manche hauptsächlich als Eingabe-, andere als Ausgabeschichten, die restlichen werden als Verarbeitungsschichten aufgefasst. Es lassen jedoch nicht nur Gehirnteile mit expliziter Schichtenstruktur als Ein-/Ausgabe-Einheiten ansehen, sondern praktisch alle. Für die *effektive Modellierung* eines solches Ein- und Ausgabe-Verhalten bietet sich ein Neuronales Netzwerk aus zwei Knotenschichten an. Alle relevanten physiologischen Prozesse, die von der Ein- zur Ausgabe führen, müssen dabei in der Verarbeitungsabbildung $\mathcal{A}$ berücksichtigt werden.

Für den *multivar*-Algorithmus und das zugehörige Netzwerk, das ebenfalls aus zwei Schichten besteht, möchte ich die Prinzipien, auf denen die Modellierung von $\mathcal{A}$ beruht, am Beispiel der visuellen Informationsverarbeitung zwischen Retina und primärem visuellem Kortex (V1) vorführen. Abbildung 14 zeigt ein vereinfachtes Bild der Verschaltung zwischen Retina und dem Kortexbereich V1. Die lichtempfindlichen Rezeptoren auf der Netzhaut stellen die obere Schicht dar, in diesem Fall mit einem Erregungsmuster, das von einem balkenförmigen Lichtfleck herrührt. Darunter ist der V1 skizziert, der wegen des Musters auf der Retina ebenfalls eine bestimmte Erregung aufweist. Im Gehirn befinden sich dazwischen noch weitere Schichten, nämlich zunächst die Zellen direkt hinter der Retina, die das Signal der Lichtrezeptoren verarbeiten, anschließend der seitliche Kniehöcker, der das Signal ein weiteres Mal verarbeitet und an den V1 weitergibt. Diese Schichten sind durch synaptische Verbindungen miteinander verknüpft, die alle eine gewisse Übertragungsstärke haben. Im Bild werden diese Verknüpfungen in einem einzigen synaptischen Baum $\mathbf{S}_r$ zusammengefasst.

Die Kunst der Abstraktion mittels Neuronaler Netze besteht darin, die beiden Zwischenschichten und ihre Verbindungen durch ein effektives Modell mit einer einzigen, möglichst einfachen Abbildung $\mathcal{A}$ zu beschreiben. Wie man vom komplizierten physiologischen Bild in Abb. 14 zum einfachen *multivar*-Netzwerk in Abbildung 15 kommt, soll auf den folgenden Seiten durch die kombinierte Darstellung von physiologischen Prinzipien und ihren mathematischen Entsprechungen deutlich werden. Die Ausführungen lehnen sich an die von Dersch (1995) an und sind dort genauer nachzulesen. Ich werde nicht auf neurophysiologische Details eingehen, diese finden sich in medizinischen Standardwerken wie Schmidt & Schaible (2000) oder Schmidt, Thews & Lang (2000).

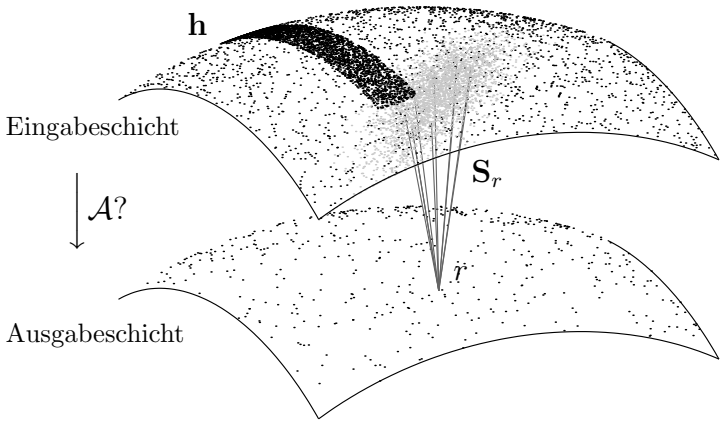


Abbildung 14: Eine abstrahierte Retina und ein primärer visueller Kortex. Der Erregungsbalken $\mathbf{h} \in \mathbb{R}^N$ auf der oberen Schicht, die aus N Lichtsensoren besteht, wird durch den synaptischen Baum $\mathbf{S}_r$ und diverse nicht eingezeichnete Zellschichten an die r-te Nervenzelle auf dem V1 weitergeleitet.

1.2.1 Die Aktivität eines Neurons

Grundsätzlich bestehen die Signale, die eine Nervenzelle an die nächste weitergibt, aus einzelnen *Spikes*, kurzen Aktivitätspulsen, die sich mit konstanter Geschwindigkeit im Axon fortbewegen. In vielen Bereichen des Gehirns ist die genaue Abfolge der *Spikes* für die Informationskodierung wichtig. Das ist besonders in den entwicklungsgeschichtlich älteren Teilen der Fall, wo wenig bis gar keine Lernprozesse stattfinden, aber auch in Verschaltungen, die explizit auf die Zeitinformation von Signalen angewiesen sind, wenn sie etwa Zeitabstände von Ereignissen messen sollen. Mit dieser Eigenschaft der Kodierung beschäftigen sich die *Single-spike* Modelle (Rieke et al. 1999). Sie müssen auch alle Verzögerungszeiten von *Spikes* in Axonen und Dendriten berücksichtigen, um physiologienahe Aussagen machen zu können. Dieser Detailreichtum macht es unmöglich, große Netze zu simulieren.

Die Aktivitäten a_r des *multivar*-Netzwerks haben kaum noch etwas mit Zellmembranpolarisierungen oder *Spikes* zu tun. Ihre Modellierung gehorcht mehr mathematisch/algorithmischen Notwendigkeiten als dem Ziel, die Kommunikation zwischen einzelnen Nervenzellen nachzubilden. Man kann sie sich als mittlere *Feuerraten* der

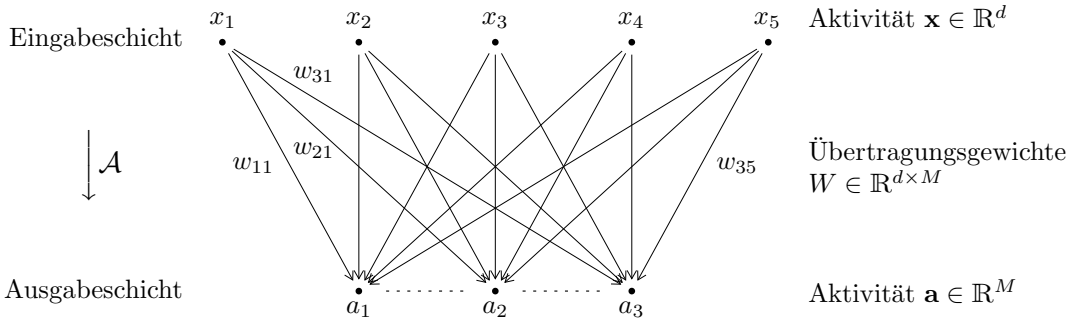


Abbildung 15: Das *multivar*-Netzwerk. Es stellt das Reizmuster auf der Retina, die aus $N \gg d$ Nervenzellen besteht, vereinfacht durch $\mathbf{x} \in \mathbb{R}^d$ dar. Die gesamte visuelle Verarbeitung wird in der Abbildung $\mathcal{A}$ zusammengefasst. Die gestrichelt eingezeichneten effektiven Verbindungen sind keine synaptischen Verknüpfungen, sondern werden durch globale Normierung der Aktivierungen erzeugt.

Nervenzellen vorstellen, handelt sich aber beim Versuch einer physiologischen Interpretation das Problem ein, dass die *Update*-Dynamik, die durch die Abbildung $\mathcal{A}$ gegeben ist, keinerlei Gedächtniseffekte im Somapotential berücksichtigt. Die *zeitliche Annahme*, die dem *multivar*- und anderen künstlichen Neuronalen Netzen zugrundeliegt, lautet deshalb:

- Es wird ein effektiver Zeittakt von Signalen vorgegeben. In jedem Zeitschritt wird ein Reizmuster $\mathbf{h}$ auf der Eingabeschicht präsentiert und die Aktivitätsantwort der nachgeschalteten Schichten berechnet.

Im physiologienahen Schema in Abb. 14 ist dargestellt, wie die Berechung von Aktivitäten der Ausgabe-Zellschicht funktioniert. Das schwarz eingezeichnete Erregungsmuster auf der Schicht der lichtempfindlichen Sensoren hat die Form eines Balkens. Die grau eingezeichneten Sinneszellen besitzen Verbindungen zur Zelle r auf dem V1. Diejenigen Sinneszellen, die sowohl dem grauen wie auch dem schwarzen Bereich angehören, geben ihre Aktivierung an r weiter, und zwar durch dessen synaptischen Baum $\mathbf{S}_r$. Dabei hat jede Verbindungsleitung eine spezifische Stärke, mit der sie das Signal hemmt oder verstärkt. So bekommt man als Vorschrift zur Berechnung der Aktivität des r-ten Neurons

$$\mathcal{A}_{\mathbf{S}_r}:\ \mathbf{h} \mapsto a_r(\mathbf{h}) = \sum_{q=1}^{N} h_q S_{rq} = \mathbf{h}^{\mathrm{T}}\,\mathbf{S}_r \quad \forall r. \tag{1-54}$$

Dabei sind $\mathbf{h} \in \mathbb{R}^N$ und $\mathbf{S}_r \in \mathbb{R}^N$ Vektoren mit so vielen Einträgen wie es Sinneszellen auf der Eingabeschicht gibt, a_r ist das Skalarprodukt beider. Die gleiche Abbildung wurde übrigens schon beim Perzeptron in (5) verwendet.

1.2.2 Der Merkmalsraum

Wenn man die Informationsverarbeitung im Gehirn verstehen möchte, muss man nach der *Bedeutung* der Nervenzellen und ihrer Aktivitätsmuster fragen. Es ist nicht leicht, diese Bedeutung festzustellen. Ein Beispiel, bei dem dies gelungen ist, liefern die Untersuchungen von Hubel & Wiesel (1974) am primären visuellen Kortex (V1) von Affen und Katzen. Es eignet sich deshalb gut, um die Modellierung durch ein Neuronales Netz zu demonstrieren. Die Autoren stellten fest, dass kleine geordnete Bereiche, die sie deshalb auch *Orientierungssäulen* nannten, stark reagieren, wenn ein Helligkeitsbalken bestimmter Position, Orientierung und Bewegungsrichtung auf die Retina des Labortieres projiziert wird (siehe auch Schmidt & Schaible, 2000). Es scheint also so zu sein, als wäre der V1 so mit der Retina verschaltet, dass er eine Balkendetektion durchführt. Jede Zelle dort repräsentiert somit einen Balken definierter Lage, Orientierung, Geometrie und Bewegung. Allgemein nennt man denjenigen Reiz auf der Eingabeschicht, der eine Nervenzelle der nachgeschalteten Schicht maximal erregt, ihren *Bestreiz* und identifiziert seine Eigenschaften mit der *Bedeutung* der Zelle. Im Fall des V1 sind die Bestreize also Balkenmuster.

An dieser Stelle definiert man zum Zweck der Abstraktion und – damit einhergehend – zur Motivation des *multivar*-Netzwerkes den *Merkmalsraum* $\mathcal{M}$ als hochdimensionalen reellen Raum, der von den Eigenschaften der Bestreize, den *Features*, aufgespannt wird (Dersch, 1995). Im gegebenen Beispiel sind dies die Breite und Länge des Helligkeitsbalkens sowie sein Orientierungswinkel, ferner die beiden Ortskoordinaten auf der Retina und seine Bewegungsrichtung. Ein Lichtbalken mit diesen Eigenschaften stellt nun genau einen Punkt im Merkmalsraum dar. Der Grund für die Definition liegt darin, dass beliebige Muster auf der Retina gerade nach denjenigen Eigenschaften sortiert werden, die bei den betrachteten Nervenzellen zu Bestreizen führen. So kann ein Erregungsmuster in seine *Bedeutungsinhalte* zerlegt werden.

Der Bestreiz einer Nervenzelle auf der Verarbeitungsschicht ist also nicht nur ein durch Breite, Länge, Position und Orientierung wohldefinierter Lichtfleck auf der Retina, sondern gleichzeitig ein Punkt im Merkmalsraum. Hier sieht man die *doppelte Beschreibungsweise*, die in der Neuroinformatik immer notwendig ist. Zum einen haben die betrachteten Größen einen biologischen Hintergrund (den Lichtfleck auf der Retina und das daraus folgende Aktivitätsmuster der Sinneszellen), auf der anderen Seite werden sie durch mathematische Begriffe (den Punkt im Merkmalsraum) beschrieben.

Als *rezeptives Feld* wird derjenige Bereich der Netzhaut bezeichnet, von dem aus eine kortikale Zelle erregt werden kann. Die Verbindungen vom rezeptiven Feld zu dieser Zelle bilden ihren *synaptischen Baum*. Entsprechend kann das rezeptive Feld im Merkmalsraum definiert werden, und zwar als diejenigen Punkte $\mathbf{x} \in \mathcal{M}$, die eine Zelle erregen können. Analoges gilt für den synaptischen Baum.

Die Transformation $\mathcal{T}$, mit der die Beschreibungsweise von den Nervenzellenschichten in den Merkmalsraum gebracht wird, ist in Abbildung 16 dargestellt. Die Position, Länge, Breite etc. des Helligheitsbalkens $\mathbf{h}$ definieren die Koordinaten des Punktes $\mathbf{x}(\mathbf{h}) \in \mathcal{M}$. Als *Position* von $\mathbf{h}$ wird sein Schwerpunkt in den Koordinaten z_1 und

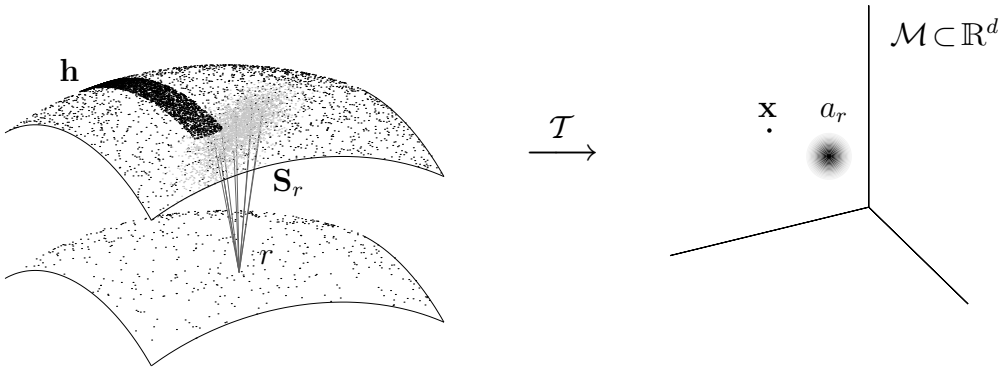


Abbildung 16: Die Transformation $\mathcal{T}$ von den Schichten der Nervenzellen in den Merkmalsraum $\mathcal{M}$. Der Reizbalken $\mathbf{h}$ (schwarz eingezeichnet) wird auf den Punkt $\mathbf{x}$ abgebildet. Die Aktivitäten des Neurons r, definiert durch sein rezeptives Feld (grau eingezeichnet) und den synaptische Baum $\mathbf{S}_r$ wird in die Funktion $a_r\colon \mathcal{M} \to \mathcal{M}$ überführt. Die Erregung von r ist im linken Bild $\mathbf{h}^{\mathrm{T}}\,\mathbf{S}_r$, im rechten $a_r(\mathbf{x})$. Jedem Koordinatenwert x_i des Reizmusters im Merkmalsraum entspricht in Abb. 15 die Aktivität x_i des i-ten *virtuellen* Eingabeneurons.

z_2 der Retina angenommen. Man bekommt als erste und zweite Koordinate von $\mathbf{x}$ (vgl. Dersch, 1995),

$$x_i = \mathbf{z}_i^{\mathrm{T}} \frac{\mathbf{h}}{\|\mathbf{h}\|_1} = \sum_q \mathbf{z}_{qi} h_q \Big/ \sum_q h_q, \quad \text{mit} \quad i = 1, 2, \tag{1-55}$$

wobei q der Index einer Nervenzelle auf der Eingabeschicht und h_q ihre Aktivierung ist. Die dritte Komponente x_3, die Orientierung des Balkenmusters, kann durch eine Hauptkomponentenanalyse bestimmt werden. Wie in (1-55) der Schwerpunkt von $\mathbf{h}$, so wird nun analog die Kovarianzmatrix $\mathbf{\Sigma}_{\mathbf{h}}$ berechnet,

$$\mathbf{\Sigma}_{\mathbf{h}} = \sum_q \left[\mathbf{z}_q - \tbinom{x_1}{x_2}\right] \left[\mathbf{z}_q - \tbinom{x_1}{x_2}\right]^{\mathrm{T}} h_q \Big/ \sum_q h_q. \tag{1-56}$$

x_3 ist die Orientierung des Eigenvektors mit dem größeren Eigenwert. Die vierte und fünfte Koordinate, Länge und Breite des Musters $\mathbf{h}$, sind dann die Standardabweichungen in Richtung der Hauptachse und normal dazu. Die Bewegungsrichtung des Schwerpunktes bildet schließlich die restlichen beiden Koordinaten. Auf diese Weise hat man eine Abbildung $\mathcal{T}$ von Reizmustern nach Punkten im Merkmalsraum bekommen,

$$\mathcal{T}\colon \quad \mathcal{H}(\mathbb{Z}^2) \to \mathcal{M}\colon \quad \mathbf{h} \mapsto \mathbf{x}(\mathbf{h}). \tag{1-57}$$

Dabei ist $\mathcal{H}(\mathbb{Z}^2)$ der Raum aller Reizmuster, die auf der Retina als Teilmenge des $\mathbb{Z}^2$ entstehen können.[4] Der Merkmalsraum ist durch geeignete Definition seiner Achsen, der *Features*, gerade so gemacht, dass Bestreizmuster nach Punkten abgebildet werden. Für alle anderen Muster verwendet man im *multivar*-Netzwerk die folgende Näherung:

[4] Hier ist die Retina zuächst als Gitter aufgefasst worden. In einer allgemeinere Formulierung wäre sie Teilmenge des $\mathbb{R}^2$, und die Summen in (1-55) und (1-56) müssten durch Integrale ersetzt werden.

- Alle Reizmuster sollen einem Balken so ähnlich sein, dass ihre Transformierte so stark lokalisiert ist, dass man das Reizmuster durch einen Punkt $\mathbf{x} \in \mathcal{M}$ ersetzen kann, $\mathcal{T}\colon \mathbf{h} \mapsto \mathbf{x}$.

Somit ist jedes Reizmuster im Merkmalsraum definiert. Es muss nun noch eine Möglichkeit gefunden werden, die Abbildung $\mathcal{A}$ im Merkmalsraum zu beschreiben. Dazu berechnet man die Aktivierungen a_r der Nervenzelle r für alle möglichen Balkenmuster $\mathbf{h}$. Im Merkmalsraum wird so jedem Punkt $\mathbf{x}$ ein Aktivitätswert $a_r(\mathbf{x})$ zugeordnet, der nur von $\mathbf{h}$ und dem synaptischen Baum $\mathbf{S}_r$ abhängt. Auf diese Weise bekommt man $a_r\colon \mathcal{M} \to \mathcal{M}$ als Funktion im Merkmalsraum.

Die Aktivierung eines Neurons durch ein normiertes Reizmuster h, das nun als Funktion auf der (kontinuierlichen) Retina $\mathbb{R}^2$ aufgefasst wird, und das den Schwerpunkt $\mathbf{x}$ hat (vgl. Gleichung 1-55),

$$\mathbf{x} = \int_{\mathbb{R}^2} h(\mathbf{z})\mathbf{z}\, d\mathbf{z} \tag{1-58}$$

ist durch die kontinuierliche Version von (1-54) gegeben,

$$a_r(\mathbf{x}) = \int_{\mathbb{R}^2} h(\mathbf{z}) S_r(\mathbf{z})\, d\mathbf{z}. \tag{1-59}$$

Diese Gleichung besitzt eine formale Ähnlichkeit mit einer Entwicklung nach Basisfunktionen. h ist aus dem Satz von Funktionen, nach denen der synaptische Baum $S_r\colon \mathcal{M} \to \mathcal{M}$ entwickelt wird. Der Wert $a_r(\mathbf{x})$ des Integrals ist dann der Entwicklungskoeffizient, der zu h gehört. Deshalb kann man die Funktion a_r als die „Balken-Transformierte“ des synaptischen Baumes verstehen.[5] Die Funktion a_r im Merkmalsraum kann also mit dem transformierten synaptischen Baum des Neurons r identifiziert werden,

$$\mathcal{T}\colon\ S_r \to a_r, \tag{1-60}$$

wobei a_r hier auch gleichzeitig noch die Aktivierung des r-ten Neurons ist, erst im nächsten Abschnitt wird eine Unterscheidung zwischen Aktivierung a_r und transformiertem synaptischem Baum nötig. Sie hat ihr Maximum gerade an der Stelle, wohin der Bestreiz der Nervenzelle r durch $\mathcal{T}$ abgebildet wird. Diesen Punkt nennt man auch den *virtuellen Ort* des Neurons.

Nun kann die dritte zentrale Näherung formuliert werden, die dem *multivar*-Netzwerk zugrundeliegt:

- Jeder synaptische Baum $a_r(\mathbf{x})$ soll hinreichend lokalisiert sein. Hinreichend heißt hier, dass man seine Eigenschaften durch eine Entwicklung bis zum zweiten Moment – also durch ihren Schwerpunkt $\mathbf{c}_r$und ihre Kovarianzmatrix $\mathbf{\Sigma}_r$ – genügend genau erfasst hat. Er ist also eine Gaußfunktion, $a_r(\mathbf{x}) = \mathrm{GF}(\mathbf{x}; \mathbf{c}_r, \mathbf{\Sigma}_r)$.

[5] Diese Transformation als Entwicklung nach Balkenmuster ist natürlich nicht im strengen Sinne eine Entwicklung nach einem Satz von Basisfunktionen, wie es z. B. die Fouriertransformation ist. In einem anschaulichen Sinne wird dennoch, wie auch bei echten Entwicklungen, die Übereinstimmung der Gewichtungsfunktion S_r mit jedem Balkenmuster h bestimmt.

Die Verarbeitungsabbildung (1-54) lässt sich jetzt als einfache Funktionenauswertung schreiben,

$$\begin{aligned}\mathcal{A}_{\{\mathbf{c}_r,\Sigma_r\}}\colon\ \mathbf{x}\mapsto a_r(\mathbf{x}(\mathbf{h})) &= \exp\Big(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_r)^{\mathrm{T}}\Sigma_r^{-1}(\mathbf{x}-\mathbf{c}_r)\Big)\frac{1}{(2\pi)^{d/2}|\Sigma_r|^{1/2}} \\ &= \mathbf{h}^{\mathrm{T}}\,\mathbf{S}_r \quad \forall r.\end{aligned} \tag{1-61}$$

Die Einführung des Merkmalsraumes vereinfacht die Berechnung von Aktivitäten ganz erheblich. Statt komplizierte Aktivierungsmuster mit einem hochdimensionalen Skalarprodukt weiterzuleiten, werden sie in den Merkmalsraum gebracht und dort als einfache Punkte und als Funktionen mit wenigen Parametern verarbeitet. Wie sehr dieses Vorgehen den Aufwand reduziert, wird in Abschnitt 1.2.6 ausgeführt.

Im Merkmalsraum, der in Abschnitt 1.1 einfach als reeller Raum vorausgesetzt wurde, steckt nun ein wichtiger Teil der Modellierung durch das MVNN, nämlich die Definition geeigneter *Features*. Bei konkreten Problemen muss die Transformation $\mathcal{T}$ in einer Vorverarbeitung des Datensatzes verwirklicht sein. Zu dem einfachen Bild in Abb. 15 kommt man, indem man die i-te Koordinate des Punktes $\mathbf{x}$ als die Aktivität des i-ten Knotens in der Eingabeschicht versteht.

1.2.3 Kompetition durch Aktivitätsnormierung

Die Aktivierung von Nervenzellen wird nicht nur durch die Verbindung zwischen Schichten gesteuert, sondern auch durch Verbindungen innerhalb einer Schicht. Diese *Lateralverschaltung* kann im Neuronalen Modell entweder explizit simuliert, oder durch eine globale Normierung der Aktivität einer Schicht ersetzt werden (Willshaw & von der Malsburg, 1976). In *multivar* wird der zweite Ansatz verwirklicht, wie durch die gepunkteten Querverbindungen in Abb. 15 angedeutet wurde. Es wird sowohl die Aktivität $\mathbf{h}$ der Eingabeschicht, als auch die der Verarbeitungsschicht normiert,

$$\sum_q h_q = 1 \quad \text{und} \quad \sum_r a_r = 1, \tag{1-62}$$

wodurch sich auch die Abbildung $\mathcal{A}$ aus (1-61) verändert. Jede Einzelaktivität $a_r(\mathbf{x})$ wird durch die Gesamtaktivität $A(\mathbf{x})$ der Schicht dividiert,

$$\mathcal{A}_{\{\mathbf{c}_r,\Sigma_r\}}\colon\ \mathbf{x}\mapsto a_r(\mathbf{x}(\mathbf{h})) = \frac{\mathrm{GF}(\mathbf{x};\mathbf{c}_r,\Sigma_r)}{A(\mathbf{x})} \quad \forall r, \tag{1-63}$$

$$A(\mathbf{x}) := \sum_{i=1}^{M} \mathrm{GF}(\mathbf{x};\mathbf{c}_i,\Sigma_i). \tag{1-64}$$

Die Funktionen a_r sind nun keine einfachen Gaußfunktionen mehr, sondern werden *generalisierte* Gaußfunktionen genannt. Sie bilden nun die tatsächlichen Aktivitäten des Modells und unterscheiden sich nun von den synaptischen Bäumen, die im letzten Abschnitt ebenfalls mit a_r bezeichnet wurden. Die Aktivitäten können nicht mehr beliebige Werte annehmen, sondern erhalten eine größere Aktivität nur noch auf Kosten

von Aktivitäten der gesamten Schicht. Die Neuronen *kompetieren* um die gemeinsame Aktivität. Durch diese effektive Beschreibung der Lateralverschaltung wird vermieden, die Lateralverbindungen explizit simulieren zu müssen. Es müssen keine hemmenden Querverbindungen berücksichtigt werden, und auch die Verknüpfungen der Eingabe- zur Verarbeitungsschicht können als rein exzitatorisch angenommen werden.

An dieser Stelle ist die Modellbildung so weit, dass die Aktivität a_r mit der bedingten Klassenwahrscheinlichkeit $\hat{P}(r|\mathbf{x})$ identifiziert werden kann,

$$a_r(\mathbf{x};\boldsymbol{\theta}) \equiv \hat{P}(r|\mathbf{x};\boldsymbol{\theta}). \tag{1-65}$$

Beide erfüllen die Normierung in (1-62), und mit der Gleichung (1-63) wurde der Bayessche Satz (1-10) wiederentdeckt. Der Vergleich der Nenner in beiden Gleichungen liefert zusätzlich die Identität von Approximation $\hat{p}(\mathbf{x};\boldsymbol{\theta})$ und der Gesamtaktivität $A(\mathbf{x})$,

$$A(\mathbf{x};\boldsymbol{\theta}) \equiv \hat{p}(\mathbf{x};\boldsymbol{\theta}). \tag{1-66}$$

Am Zähler sieht man, dass die gewichtete Normalverteilung $\hat{P}\,\hat{p}(\mathbf{x}|r;\boldsymbol{\theta})$ zum Bild des synaptischen Baumes im Merkmalsraum gemacht wird. Die Näherung für a_r auf Seite 33 ist also nichts anderes als die zentrale Modellannahme, dass die Verteilungsdichte der Reize mit einer Mischung von Normalverteilungen approximiert werden kann.

Physiologisch kann die globale Aktivitätsnormierung als Ressourcennormierung gerechtfertigt werden. Die Vorstellung dahinter ist, dass immer viele Nervenzellen gemeinsam von wenigen Blutgefäßen versorgt werden. Wenn man sich die Aktivität als Ressourcen verbrauchenden Zustand vorstellt, kann eine Nervenzelle nur auf Kosten der Aktivität anderer Zellen erregt werden. Das ML-Prinzip bedeutet hier, dass dieser grundlegende Organisationsprozess der Kompetition nach (1-66) und (1-7) die mittlere Aktivität der gesamten Schicht maximiert. Die Forderung nach *Load-balance* (1-49) sorgt wegen (1-65) gleichzeitig dafür, dass die mittleren Aktivitäten der einzelnen Neuronen ähnliche Werte aufweisen. Letztere Bedingung kann durch physiologische Befunde motiviert werden. So wurde am Beispiel der somatosensorischen Hirnrinde in Amputationsversuchen festgestellt, dass Kortexneuronen ihre rezeptiven Felder verändern, wenn sie lange nicht erregt werden. Sie beginnen, sich an der Kodierung noch aktiver Bereiche zu beteiligen (Schmidt & Schaible, 2000).

Die beiden oben skizzierten Prinzipien, die *Aktivitätsmaximierung* und *Load-balance* (Albrecht et al. 2000), bilden das Kernstück des *multivar*-Algorithmus. Sie wurden als ML-Prinzip und als Kopplung der Gewichtungen $\hat{P}_r$ an den Wert $1/M$ in den *multivar*-Algorithmus eingebaut. Auf diese Weise fügen sich die beiden Sichtweisen, mathematische und neuronale, zwanglos ineinander. Ein weiteres Beispiel stellt die Lernregel (1-41) dar, die im folgenden Abschnitt neuronal motiviert werden soll.

1.2.4 Hebb'sches Lernen

Durch die Verwendung von Aktivität und synaptischer Stärke legt man ein Modell zugrunde, das aus zwei Größen mit unterschiedlichen Zeitskalen besteht. Beim Lernen wird durch die schnell veränderlichen Aktivitätsmuster eine langsame Veränderung der synaptischen Gewichte hervorgerufen. Dieser Vorgang wurde zuerst von Hebb (1949, S. 62) postuliert:

> »When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.«

Diese Feststellung, auch als *Hebb'sche Lernregel* bezeichnet, legt die Art der Verstärkung und die beteiligten Nervenzellen nicht fest. Es sind verschiedene Szenarien denkbar, die zu einer Verstärkung der synaptischen Verbindung zweier Zellen führen (Sejnowski & Tesauro, 1989). Ein häufiger Ansatz ist jedoch die Verwendung des Korrelationsproduktes aus prä- und postsynaptischer Aktivität, im Falle des ANN in Abb. 14 also des Produktes $a_r h_q$, als Wachstumsterm für die Verbindungsstärke S_{rq}. Eine solche Lernregel hat dann die Form

$$S_{rq}(t+1) = S_{rq}(t) + \varepsilon a_r(t) h_q(t). \tag{1-67}$$

Die Bedingung der *wiederholten oder ständigen* Aktivierung wurde schon in der Einleitung als Mittelungsprozess hervorgehoben. Die Lernregel (1-67) führt jedoch zu unbeschränktem Wachstum, da nur positive Terme addiert werden. Zieht man $a_r(t)S_{rq}(t)$ als Beschränkungsterm ab, so bekommt man die *kompetitive* Form der Hebb'schen Lernregel,

$$S_{rq}(t+1) = S_{rq}(t) + \varepsilon a_r(t)\big(h_q(t) - S_{rq}(t)\big). \tag{1-68}$$

Bei ihrer Anwendung bleibt die Norm von $\mathbf{S}_r$ erhalten, denn mit der Normierung von $\mathbf{h}$ aus (1-62) folgt sofort auch

$$\sum_{q=1}^{N} S_{rq}(t) = 1 \quad \Longrightarrow \quad \sum_{q=1}^{N} S_{rq}(t+1) = 1 \quad \forall r. \tag{1-69}$$

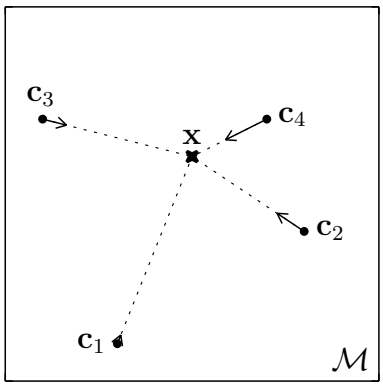


Abbildung 17: Schrittweite von vier Neuronen im Merkmalsraum, entsprechend der kompetitiven Hebb'schen Lernregel (1-70). Die Zentren rutschen in die Richtung des Reizes $\mathbf{x}$, aber nur einen gewissen Anteil $\varepsilon a_r < 1$ der Gesamtstrecke.

Im Merkmalsraum findet man eine äquivalente kompetitive Lernregel, indem man den Reiz $\mathbf{h}$ und den synaptischen Baum $\mathbf{S}_r$ in den Merkmalsraum transformiert. Anwendung der Gleichungen (1-55) und (1-56) auf (1-68) liefert die Koordinaten der Gleichung

$$\mathbf{c}_r(t+1) = \mathbf{c}_r(t) + \varepsilon a_r(t)\big(\mathbf{x}(t) - \mathbf{c}_r(t)\big). \tag{1-70}$$

Dabei wurde ausgenutzt, dass das Zentrum $\mathbf{c}_r$ der Aktivierungsfunktion a_r gleichzeitig auch der virtuelle Ort von r ist, sofern a_r genügend lokalisiert ist. Diese Lokalisierung wird durch die Annahme für $\mathbf{h}$ auf Seite 33 garantiert.

Gleichung (1-70) ist die Lernregel (1-41) des *multivar*-Algorithmus. Es ist demnach nicht nur die Verarbeitungsabbildung $\mathcal{A}$ des MVNN sowohl in mathematischer und neuronaler Sichtweise in Übereinstimmung gebracht, sondern es wurde auch seine Lernregel $\mathcal{P}$ als kompetitives Hebb'sches Lernen identifiziert. Man kann sie leicht geometrisch veranschaulichen, und zwar als Bewegung des virtuellen Ortes $\mathbf{c}_r$ auf den Reiz $\mathbf{x}$ zu, wobei seine Schrittweite der Anteil $\varepsilon a_r < 1$ an der Gesamtstrecke ist (Abbildung 17). Auch für die Lernregel (1-42), die eine Hauptachsenbestimmung durchführt, gibt es eine geeignete neuronale Formulierung (Rubner & Tavan, 1989; Albrecht et al. 2000).

1.2.5 Kortikale Merkmalskarten

Die Lernregel (1-70) ist ein Spezialfall eines Lernszenarios, das die Entstehung neuronaler Karten erklärt. Untersuchungen am visuellen System von Säugetieren zeigen, dass benachbarte Orte der Netzhaut im V1 ebenfalls benachbart abgebildet werden (retinotope Abbildung). Die Größe des kodierenden Bereiches im V1 ist proportional zur Gangliendichte auf der Retina (Schmidt & Schaible, 2000). Eine solche nachbarschaftserhaltende und gleichzeitig dichteorientierte Repräsentation nennt man eine *topographische Merkmalskarte*.

Besonders deutlich wird der Begriff am Beispiel des somatosensorischen Rindenfeldes. Dort bildet sich eine Repräsentation des gesamten sensorischen Nervensystems im Körper aus, die man als *Homunculus* bezeichnet. Dabei wird die (im wesentlichen) zweidimensionale Hautoberfläche auf die Kortexoberfläche abgebildet. Nebeneinander liegende Tastrezeptoren werden von nebeneinander liegenden Kortexzellen verarbeitet. Die kortikale Merkmalskarte ist also auch hier *nachbarschaftserhaltend*.

Ein weiteres Beispiel für die Verwendung einer kortikalen Karte ist das Gehör der Fledermaus, deren Hörzentrum eine Repräsentation der Tonhöhe und seiner Amplitude liefert (Rieke et al. 1999). Diese Merkmalskarte ist nicht in dem einfachen Sinne topologieerhaltend wie der Homunculus, doch wenn man ihn im geeigneten Merkmalsraum betrachtet (Frequenz, Amplitude, evtl. Phase und Veränderung der Frequenz), dann sieht man, dass die Erhaltung der Nachbarschaftsbeziehung schon in der Definition des Merkmalsraumes liegt. Dieser wird gerade von denjenigen *Features* aufgespannt, die das *Nebeneinander* auf dem Kortex möglichst gut wiedergeben, wie auch schon im letzten Abschnitt klar geworden ist.

Anhand solcher Beispiele wird die These formuliert, dass die Organisation in neuronalen Merkmalskarten ein universelles Prinzip der Verarbeitung von Sinnesreizen, vielleicht sogar des gesamten Gehirns ist (Dersch, 1995 und Schmidt & Schaible, 2000, S.214).

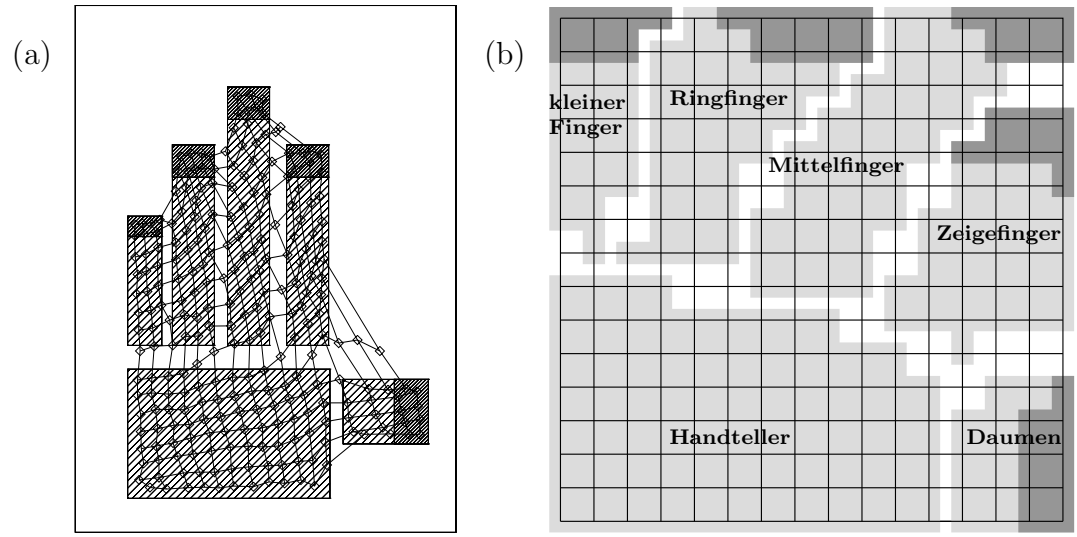


Abbildung 18: (a) Beispiel einer topographischen Karte nach einem Lernprozess des Kohonen-Netzes. Die dicht schraffierten Flächen sind Bereiche hoher Reizdichte, und entsprechend ist dort auch die Dichte der virtuellen Orte der Neuronen höher. In (b) ist die Zuordnung der Reizbereiche zu den Neuronen auf dem kortikalen Gitter zu sehen. Bilder aus Dersch (1995).

Das bekannteste Beispiel eines Neuronalen Netzes zur Erstellung einer kortikalen Karte ist der Algorithmus von Kohonen (1982) (siehe auch Ritter, Martinetz & Schulten, 1992), dessen Lernregel für die virtuellen Orte $\mathbf{c}_r$ der Neuronen im Merkmalsraum auch in der Form (1-70) geschrieben werden kann, jedoch mit einer anderen Aktivierungsfunktion als beim *multivar*-Algorithmus. Abbildung 18a zeigt das Ergebnis eines solchen Lernprozesses, einmal im Merkmalsraum (a) und einmal auf dem Kortex, der hier aus einem zweidimensionalen Gitter besteht. Im Merkmalsraum positionieren sich die virtuellen Orte so, dass ihre Dichte $D(\mathbf{c})$ (definiert durch den Grenzübergang beliebig vieler Neuronen) dort groß ist, wo auch die Reizdichte $p(\mathbf{x})$ groß ist. Eine genaue Analyse des funktionalen Zusammenhangs von D und p ist bei Dersch (1995) zu finden. Die physikalische Nachbarschaftsbeziehung der Neuronen wird im Kohonen-Algorithmus über ein festes Gitter modelliert. Abbildung 18b zeigt dieses Gitter und die Zuordnung zu den Orten auf der Reizfläche, die in Abb. 18a vorgenommen wurde. Das Ergebnis ist eine typische nachbarschaftserhaltende und dichteorienterte Merkmalskarte.

Merkmalskarten durch lokale PCA

Das MVNN, das in dieser Arbeit verwendet wird, hat keine vorgegebene Topologie. Seine räumliche Organisation beruht ausschließlich auf gegenseitiger Kompetition im Merkmalsraum nach Gleichung (1-10), weshalb die Merkmalskarten, die durch Anwenden der Lernregel (1-41) entstehen, nicht nachbarschaftserhaltend, sondern lediglich dichteorientiert sind. Eine neuronale Karte des Datensatzes aus Abb. 18 mit dem *univar*-Algorithmus bei gleicher Neuronenzahl ($M = 256$) würde ähnliche virtuelle Ort

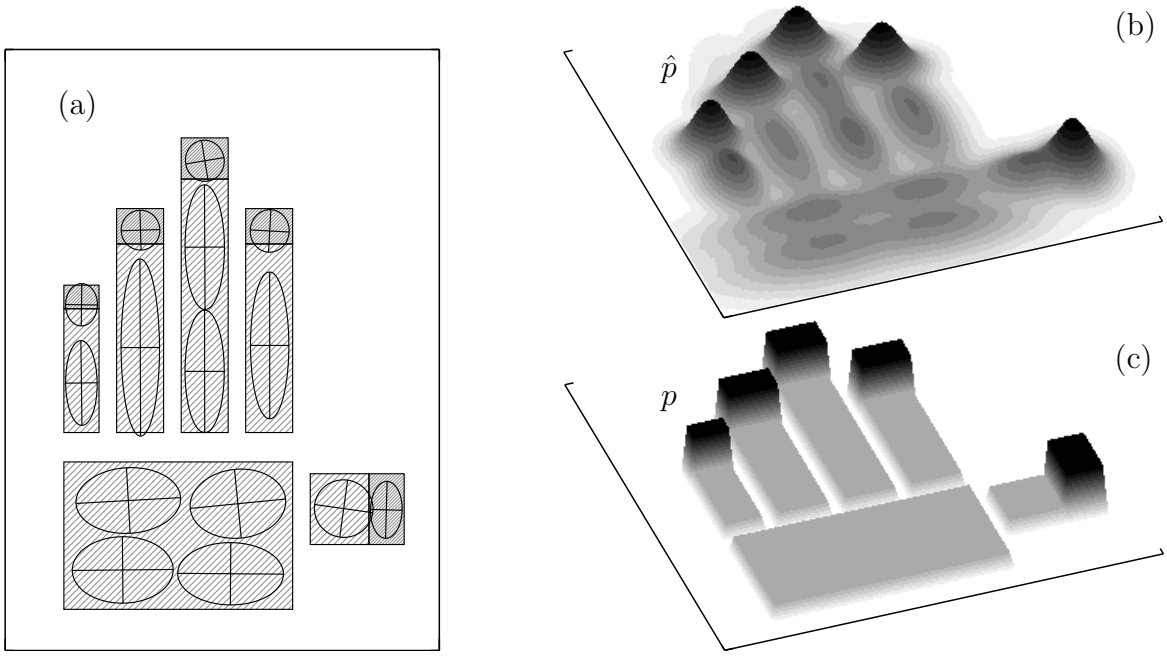


Abbildung 19: Eine Dichteschätzung durch lokale PCA mit *multivar*. Jede der Normalverteilungen in (a) überdeckt etwa gleich viele Datenpunkte, weshalb sie an den Fingerspitzen, wo die Datendichte höher ist, schmaler und höher sind als an den Fingern und am Handteller. Die langestreckte Form der Normalverteilungen an den Fingern zeigt deutlich die lokale PCA, wie man an den Hauptachsen sehen kann, die hier mit der Länge $1.4\sigma_{ir}$ eingezeichnet sind. Am Handteller wird die Kompetition deutlich, die vier Neuronee versuchen gemeinsam, eine konstante Funktion zu approximieren. In (b) und (c) sind Ergebnis $\hat{p}$ der Dichteschätzung und die Verteilungsdichte p dargestellt.

aufweisen, nur dass die Dichte D der Orte und p gleich wären.[6]

Eine multivariate Dichteschätzung des Hand-Datensatzes ist in Abbildung 19 zu sehen. Die Anzahl der Neuronen wurde dramatisch reduziert, weshalb nun nicht nur die virtuellen Orte der Neuronen zur Dichteschätzung benutzt werden können, sondern auch die Achsen ihrer synaptischen Bäume. In diesem Fall sind nun die *Merkmale* der Hand deutlicher zu sehen als in der Repräsentation durch viele Neuronen. So lassen sich etwa die Fingerspitzen gut erkennen, an denen die Reizdichte höher ist als an den restlichen Teilen der Hand, da sie durch jeweils ein einziges Neuron repräsentiert werden. Auch die Finger selbst lassen sich als langgestreckte Formen gut erkennen. Wenn also eine optimale Dichteschätzung mit wenigen Neuronen zur Verfügung steht, lassen sich Detektoren für Reize an den Fingerspitzen oder an den Handflächen finden. Dies erzeugt aus *primären Features* und einer lokalen Hauptkomponentenanalyse *sekundäre Features*. Hier klingt ein Prozess der automatischen Feature-Extraktion durch sukzessive Schätzung und Vereinfachung des Merkmalsraumes an, der aber hier nicht beschrieben werden kann, da er den Rahmen der vorliegenden Diplomarbeit sprengen würde.

[6] Zunächst wären nach dem Dichteschätzungsprinzip p und die Mischung univariater Normalverteilungen – also die Faltung von D mit einer zentrierten univariaten Normalverteilung – gleich. Bei beliebig dichter Besetzung kann aber auch die Varianz der Normalverteilung beliebig reduziert werden, was in $p = D$ resultiert.

1.2.6 Dimensionsreduktion

Nachdem die einzelnen Schritte der neuronalen Modellbildung durch das MVNN besprochen wurden, möchte ich hier ihre Wirkung im Sinne der Dimensionsreduktion noch einmal zusammenstellen.

- Es wurde mit einem Aktivitätsmuster $\mathbf{h}$ auf der Eingabeschicht begonnen, die je nach Modellierung Teilmenge des $\mathbb{Z}^2$ oder des $\mathbb{R}^2$ ist. Der Raum aller möglichen Muster ist wegen der Normierung (1-62) dann die Einheitskugel entweder im 10^8-dimensionalen Raum aller Rezeptorzellen, oder im unendlichdimensionalen Raum aller integrablen Funktionen[7] $\mathrm{L}^1(\mathbb{R}^2)$.
- Der Merkmalsraum wurde daraufhin gerade so definiert, dass die Näherung für $\mathbf{h}$ auf Seite 33 gilt. Dann kann die Beschreibung der Reizmuster auf drastische Weise vereinfacht werden. Sie werden Punkte des Merkmalsraumes, $\mathbf{x} \in \mathcal{M} \subset \mathbb{R}^d$, können also durch d reelle Zahlen charakterisiert werden.
- Die *Gauß'schen Neuronen*, die durch die zweite Näherung auf Seite 33 eingeführt werden, sind durch ihre Zentren und Kovarianzmatrizen beschrieben, also durch $d^2 + 2d$ Parameter. Auf diese Parameter wird die gesamte Information über den synaptischen Baum $\mathbf{S}_r$ reduziert.
- Die Vereinfachungen des Merkmalsraumes betreffen jeden Reizvektor einzeln. Durch die Erstellung einer neuronalen Merkmalskarte wird der gesamte Datensatz aus T zeitlich geordneten Reizen vereinfacht. Wenn dazu M Neuronen verwendet werden, dann bleiben also zur Charakterisierung des Datensatzes nur $Md(d+2)$ Parameter übrig.

Insgesamt ist durch die vereinfachenden Annahmen des letzten Abschnitts eine Beschreibung der Umwelt des MVNN durch $Md(d+2)$ Parameter möglich geworden. Es scheint also so zu sein, als habe man den größten Teil der durch die Aktivitätsmuster übermittelten Information weggeworfen. Genau dies ist aber eines der Prinzipien neuronaler Kodierung. Es ist nicht das Ziel, alle Details der präsentierten Daten zu wiederholen, sondern eine vereinfachte Repräsentation zu erhalten, die auf die wichtigsten Eigenschaften reduziert ist. Diese Repräsentation wird auch das *effektive Modell* genannt, welches sich das MVNN von seiner Umwelt bildet.

[7] Die Notation hält sich hier und im folgenden an Fischer & Kaul (1990).

Kapitel 2

On-line Lernen mit *univar*

In der Einleitung wurde anhand des einfachen ANN aus Abb. 6 und seiner Lernregel (15) der Widerstreit zwischen Lernen und Vergessen aufgezeigt, der für alle sequentiellen stochastischen Lernalgorithmen der Neuroinformatik charakteristisch ist, sobald zeitlich korrelierte Daten aus on-line Beobachtungen gelernt werden. Anhand von Abb. 6b wurde das Problem der *Kopplung von Lern- und Systemdynamik* erklärt.

Es soll nun untersucht werden, welche Probleme tatsächlich bei dem Versuch auftreten, die Verteilungsdichte einer on-line Datensequenz mit dem im letzten Kapitel eingeführten MVNN zu approximieren. Wie bereits anhand von Abb. 13 begründet wurde, ist die erste Phase der multivariaten Dichteschätzung, die *univar*-Phase, für die dynamischen Aspekte des Lernens weit anfälliger als die zweite. Hier und in den restlichen Kapiteln der Arbeit wird deshalb von dem univariaten Fall des Dichtemodells (1-3) ausgegangen, bei dem alle Komponenten identische Varianzen und Gewichte haben,

$$\Sigma_r = \sigma^2 I \quad \text{und} \quad P_r = 1/M \qquad \forall r \in \{1, \ldots, M\}. \tag{2-1}$$

Da die wesentliche Eigenschaft des *univar*-Algorithmus das hierarchische Clustering ist, das durch den Aufspaltungsprozess des Codebuchs während des σ-Annealings erzeugt wird, muss das Problem des on-line Lernens anhand der Phasenübergänge demonstriert werden. Wie der erste Abschnitt qualitativ zeigen wird, kann auch hier eine starke Kopplung der Lern- and die Systemdynamik auftreten und die Phasenübergänge beeinflussen. Im zweiten Abschnitt werden die Effekte dieser Kopplung anhand eines stark vereinfachten prototypischen Modells quantitativ analysiert.

Da der Begriff des *on-line Lernens* in der Neuroinformatik nicht durchgängig mit derselben Bedeutung verwendet wird,[1] möchte ich noch einmal kurz definieren, was in dieser Arbeit unter einem *on-line* lernenden ANN verstanden wird:

- Alle Datenpunkte werden in ihrer natürlichen Reihenfolge nacheinander (sequentiell) präsentiert, wodurch zeitliche Korrelationen im Datenstrom erhalten bleiben.

[1] Die meisten Autoren verwenden *on-line* synonym zu *sequentiell*. In den wenigen Arbeiten, die sich mit zeitlich korrelierten Daten beschäftigen, ist deshalb von *natürlichem on-line* Lernen im Gegensatz zum *randomisierten Lernen* (Heskes & Wiegerinck, 1998) oder von *veränderlichen Umgebungen* (Heskes & Kappen, 1993) die Rede. Ich werde bei der einfachen Unterscheidung zwischen *sequentiell* und *on-line* bleiben.

- Nach jedem Datenpunkt $\mathbf{x}$ werden sowohl alle Netzwerk-, als auch alle Lernparameter aktualisiert.
- Dem ANN ist zunächst keine der im System enthaltenen Raum- oder Zeitskalen bekannt. Es muss jede Parameteradaption allein aufgrund seines eigenen „Wissens“ bestimmen, also aufgrund der aktuellen Netz- und Lernparameter, des Datenpunktes sowie der funktionalen Form der Abbildungen $\mathcal{A}$ und $\mathcal{P}$. In praktischen Anwendungen darf diese Forderung natürlich etwas abgeschwächt werden. So ist eine Abschätzung der maximalen Varianz der Daten zu Zwecken der Initialisierung von Vorteil (vgl. etwa (MV-1)). Ferner kann für jedes organische oder technische System, das die Beobachtungsdaten detektiert, eine maximale Zeitskala in Form seiner gesamten Lebens- und Lerndauer vorausgesetzt werden.

2.1 Die Kopplung von Lern- und Systemdynamik

2.1.1 Die Dynamik des *univar*-Lerners

Die Lernregel (1-41) des *univar*-Algorithmus macht das Codebuch $C(t) = \{\mathbf{c}_1(t), \ldots, \mathbf{c}_M(t)\}$ zu einem *zeitdiskreten dynamischen System*, welches sich nach der Differenzengleichung

$$\Delta\mathbf{c}_r(t) = \frac{\Delta\mathbf{c}_r(t)}{\Delta t} = \varepsilon\, a_r(\mathbf{x}_t)\,\big(\mathbf{x}_t - \mathbf{c}_r(t)\big) \tag{2-2}$$

entwickelt. Dabei ist $\Delta t \equiv 1$ der elementare Zeittakt, nichts anderes als die Dauer zwischen zwei präsentierten Datenpunkten. Ersetzt man die Beweglichkeit $\varepsilon a_r(\mathbf{x}_t)$ des r-ten Neurons formal durch die Lernrate

$$\tilde{\varepsilon}_r(t) := \varepsilon a_r(\mathbf{x}_t), \tag{2-3}$$

so wird am Vergleich mit Gleichung (15) aus der Einleitung deutlich, dass (2-2) nichts anderes als die Dynamik einer gleitenden Mittelwertsbildung ist. Ist $\tilde{\varepsilon}_r$ sehr klein, dann kann das zeitdiskrete dynamische System (2-2) durch das entsprechende *kontinuierliche* ersetzt werden, das der gewöhnlichen Differentialgleichung

$$\frac{d\mathbf{c}_r}{dt}(t) = \tilde{\varepsilon}_r(t)\,\big(\mathbf{x}_t - \mathbf{c}_r(t)\big). \tag{2-4}$$

genügt. Wenn $\tilde{\varepsilon}$ zusätzlich konstant ist, und wenn es für dieses System einen stationären Attraktor $\mathbf{c}_r^*$ gibt, also einen stationären stabilen Zustand, auf den es sich zubewegt, so geschieht dies exponentiell mit der Relaxationszeit $T_r = 1/\tilde{\varepsilon}_r$,

$$\mathbf{c}_r(t) - \mathbf{c}_r^* = \big(\mathbf{c}_r(0) - \mathbf{c}_r^*\big)\,\exp(-t/T_r). \tag{2-5}$$

Wie bereits mehrfach angeklungen, ist die konstante Lernrate $\tilde{\varepsilon}_r$ dann eine Art *Konvergenzgeschwindigkeit* und ihr Inverses die Zeitskala, auf der diese Konvergenz stattfindet. Es müssen hier mehrere Fälle unterschieden werden:

(a) Bei vollständig entarteten Codebüchern gilt immer $a_r(\mathbf{x}_t) = 1/M$, die Relaxationszeit in (2-5) ist demnach zeitlich konstant

$$T_r = \frac{M}{\varepsilon} =: T_L \,. \tag{2-6}$$

Sie wird hier mit T_L abgekürzt, da sie im folgenden noch mehrfach verwendet wird.

(b) Auch wenn das Codebuch nicht entartet ist, kann im Fall des *univar*-Lernens randomisierter Daten trotzdem noch von

$$T_r = \frac{M}{\varepsilon} \tag{2-7}$$

ausgegangen werden, denn wegen der Eigenschaft der *Load-balance* (1-49) weichen die mittleren Werte der Zuständigkeiten $a_r(\mathbf{x}_t)$ nur wenig von $1/M$ ab. Da die Daten randomisiert sind, ist für diese Mittelung ein sehr kurzes Zeitfenster ausreichend, und man bekommt effektiv wieder die konstante Relaxationszeit M/ε. Auch der Attraktor $\mathbf{c}_r^*$ der Dynamik ist, wie im Beispiel zu Abb. 11 und 12 deutlich wurde, lediglich von σ abhängig, also zeitlich konstant.

(c) Werden dagegen on-line Daten gelernt, kann nicht mehr von *Load-balance* ausgegangen werden. Es kann gut sein, dass manche Neuronen eine Weile lang nur schwach angesprochen werden, andere dagegen stark. Bei solchen Daten ist außerdem kein zeitlich konstanter Attraktor $\mathbf{c}_r^*$ der Dynamik mehr gegeben, vielmehr entwickelt sich dieser selbst mit der Zeit. Somit ist im Sinne einer exponentiellen Annäherung des Codebuchzentrums $\mathbf{c}_r(t)$ an seinen *momentanen* Attraktor $\mathbf{c}_r^*(t)$ durch die Differentialgleichung (2-4) nur noch die *instantane Relaxationszeit* gegeben,

$$T_r(t) = \frac{1}{\varepsilon a_r(\mathbf{x}_t)} \,. \tag{2-8}$$

Gedächtniskerne

Für den randomisierten Lerner in (b) oben ist mit Gleichung (2-5) die vollständige zeitliche Entwicklung des dynamischen Systems (2-4) gegeben. Aus dem Anfangswert $\mathbf{c}_r(0)$ kann bei Kenntnis von T_r und dem stationären $\mathbf{c}_r^*$ das Codebuchzentrum $\mathbf{c}_r(t)$ zu jedem Zeitpunkt t berechnet werden. Für on-line Daten lässt sich dies, da nur instantane Konvergenzzeiten zur Verfügung stehen, nicht so leicht aufschreiben. Dennoch gibt es immer einen zeitlichen *Propagator*, der dasselbe leistet. Durch rekursives Einsetzen der *Update*-Regel (2-2) in sich selbst können die aktuellen Codebuchzentren immer in die Form

$$\mathbf{c}_r(t) = k_r(t)\, \mathbf{c}_r(0) + \sum_{t'=0}^{t-1} g_r(t,t')\, \mathbf{x}(t') \tag{2-9}$$

gebracht werden. Der Propagator, der $\mathbf{c}_r(0)$ auf $\mathbf{c}_r(t)$ abbildet, ist hier formal in den Termen $k_r(t)$, $g_r(t,t')$ und in dem gesamten bis zum Zeitpunkt t beobachteten Datenstrom ausgedrückt. Dies ist bereits in der Einleitung für den Spezialfall eines einzelnen

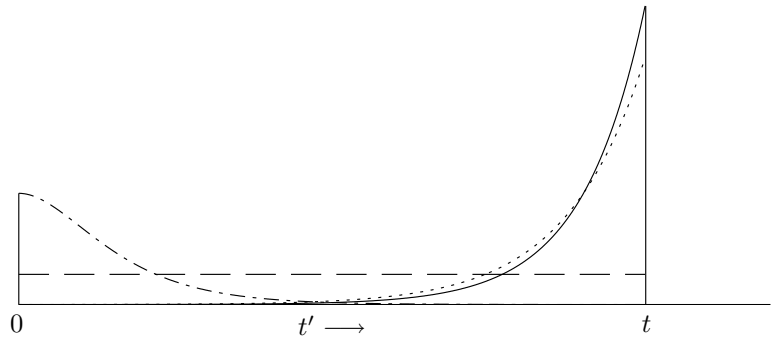


Abbildung 20: Gedächtniskerne $g_r(t,t')$ eines einzelnen Neurons zum aktuellen Zeitpunkt t. Dargestellt sind die in Appendix A auch analytisch bestimmten Kerne für verschiedene *Annealing*-Schemata: für konstantes $\varepsilon = \varepsilon_0$ (durchgezogene Linie), zeitlich inverses $\varepsilon(t) = 1/t$ (gestrichelte Linie) und exponentiell abgekühltes $\varepsilon(t) = \varepsilon_0 \exp(-t/\tau)$ (gepunktete Linie für $\tau = 5t$, Linie aus Strichpunkten für $\tau = 0.1t$). Die beiden Kerne für konstantes und langsam exponentiell vermindertes ε haben die erwartete exponentielle Form mit der Zerfallsdauer $1/\varepsilon(t)$. Sie gleiten über den Datenstrom, wobei sie ihre Form kaum verändern.

Neurons vorgeführt worden. In Appendix A ist die allgemeine Herleitung zu finden, die auch beliebige *Annealing*-Schemata für den Lernparameter $\varepsilon(t)$ erlaubt, der in (2-5) noch als konstant angenommen wurde. Dort wird auch gezeigt, dass der Term $k_r(t)$, welcher angibt, wie sehr der Anfangswert $\mathbf{c}_r(0)$ noch in $\mathbf{c}_r(t)$ enthalten ist, mit fortschreitender Zeit immer weiter zerfällt. Die Funktion $g_r(t,\cdot)$, die für jeden Zeitpunkt t' die Gewichtung des damals präsentierten Datenpunktes $\mathbf{x}_{t'}$ in der Berechnung des aktuellen $\mathbf{c}_r(t)$ liefert, wird *Gedächtniskern* genannt. Die Gedächtniskerne sind Gewichtungsfunktionen in einem Mittelungsfenster, das über den Datenstrom „gleitet." Formal wird dieser Vorgang durch die Summe in (2-9) ausgedrückt, die nichts anderes als eine diskrete Faltung des Datenstromes $\big(\mathbf{x}(t)\big)_t$ mit dem Faltungskern $g_r(t,t')$ ist.

Die Gedächtniskerne der in Appendix A beschriebenen Spezialfälle sind in Abbildung 20 gezeigt. Wie zu erwarten war, sind sie dort Null, wo $t' > t$. Der generische Fall mit konstantem ε fällt zu vergangenen Zeiten hin exponentiell ab. Die Zeitskala dieses exponentiellen Zerfalls ist identisch mit derjenigen, die in (2-5) als Zeitskala der exponentiellen Annäherung an den konstanten Attraktor $\mathbf{c}_r^*$ angegeben ist. Was die Bestimmung einer typischen Zeitskala des lernenden MVNN betrifft, sind also in diesem Fall die beiden Beschreibungsweisen der Codebuchentwicklung durch (2-5) und (2-9) äquivalent. Allerdings setzt die Formulierung mit Gedächtniskern keinen stationären Attraktor $\mathbf{c}_r^*$ voraus. Sie darf an dieser Stelle also als zwanglose Verallgemeinerung der einfachen Beschreibung einer Dynamik als exponentielle Konvergenz betrachtet werden.

Kopplung der Zeitskalen an die Raumskala

An der Definition (2-8) der instantanen Relaxationszeiten $T_r(t)$ sieht man sofort, dass diese sowohl einen räumlichen wie auch einen zeitlichen Anteil haben. Der Term ε ist rein *zeitlicher* Natur, er beeinflusst die Bewegung aller Codebuchzentren in gleicher

Weise, unabhängig von ihrem Ort. In $a_r(\mathbf{x}_t)$ wird dagegen die *räumliche* Zuständigkeit des r-ten Neurons ausgedrückt (vgl. die Partitionsfunktionen in Abb. 8b). Ein Codebuchzentrum mit hoher Zuständigkeit kann sich schnell bewegen und hat deshalb eine kürzere instantane Relaxationszeit als ein weniger zuständiges. Je kleiner die Raumskala σ des lernenden MVNN ist, umso unterschiedlicher werden die Zuständigkeiten (Abb. 9).

Dadurch, dass a_r in der Definition (2-8) verwendet wird, werden die Zeitskalen $T_r(t)$ des lernenden MVNN direkt an seine Raumskala σ gekoppelt. Diese enge Verbindung von Raum- und Zeitskalen ist eine notwendige Eigenschaft jeder Beschreibung des on-line lernenden *univar*.

Die enge Verknüpfung und die Tatsache, dass die Relaxationszeiten in (2-8) nur lokal und instantan sind, machen es schwierig, Aussagen über die *effektive* Dynamik eines on-line lernenden Codebuchs zu treffen. Aus der Summenformel

$$\sum_{r=1}^{M} \frac{1}{T_r(t)} = \varepsilon = \frac{1}{T_{\min}} \tag{2-10}$$

wird lediglich klar, dass die kleinstmögliche Zeitskala $T_{\min} = 1/\varepsilon$ zu jedem Zeitpunkt kompetitiv unter den einzelnen Codebuchzentren aufgeteilt wird. $1/\varepsilon$ ist ein geeigneter Maßstab, auf dem Zeitskalen $T_r(t)$ gemessen werden können.

2.1.2 Die Systemdynamik

Um die Kopplung der Lern- an die Systemdynamik bei einem *univar*-Training analysieren zu können, benötigen wir Modelle für dynamisch veränderliche Umwelten, deren Eigenschaften bekannt sind. Für diese Modelle bieten sich stochastische Prozesse an, die als Datengeneratoren verwendet werden können.

In Abb. 6b in der Einleitung wurde beispielsweise ein Markov-Prozess als Datengenerator verwendet, der zwei mögliche Zustände besitzt. Die beiden Zustände selbst liefern verrauschte Datenpunkte. Die Dynamik eines solchen Markov-Prozesses kann durch die *erwarteten Lebensdauern* $T_{S,\alpha}$ seiner beiden Zustände $\alpha \in \{1, 2\}$ charakterisiert werden. Diese sind durch

$$T_{S,\alpha} = \frac{1}{1 - P(\alpha|\alpha)} \tag{2-11}$$

gegeben, wobei $P(\alpha|\alpha)$ die Übergangswahrscheinlichkeit während eines Zeitschrittes Δt des Zustandes α in sich selbst ist (Sonner, 1997, Anhang A). Wenn die beiden Zustände dasselbe statistische Gewicht haben, sind die beiden erwarteten Lebensdauern gleich,

$$T_S = \frac{1}{1 - P(1|1)} = \frac{1}{1 - P(2|2)} \,. \tag{2-12}$$

In den folgenden Abschnitten werden hauptsächlich Datengeneratoren mit zwei Zuständen betrachtet. Diese können verrauscht (Abb. 6b) oder punktförmig sein.

Außer den Markov-artig schaltenden Datengeneratoren werden auch *deterministisch* schaltende verwendet. Bei ihnen sind erwartete und tatsächliche Lebensdauern gleich, ihre Zeitskalen T_S sind vorgegeben. Auch die deterministisch schaltenden Systeme werden hier sowohl mit verrauschten als auch mit unverrauschten Zuständen verwendet.

Für randomisierte Datenströme lässt sich formal ebenfalls eine Lebensdauer von Zuständen angeben. Ein randomisierter Datenstrom wie in Abb. 6a kann so erzeugt werden, dass man einen stochastischen Prozess mit zwei Zuständen zugrundelegt, die ohne Beachtung des jeweils letzten Zustandes zufällig angenommen werden. Dieser Prozess ist der triviale Fall eines Markov-Systems, denn aus den Übergangswahrscheinlichkeiten $P(\alpha|\alpha)$ werden einfache statistische Gewichte $P(\alpha)$ der Zustände. Nach Gleichung (2-11) ist in diesem Fall

$$T_{S,\alpha} = \frac{1}{1 - P(\alpha)} \,. \tag{2-13}$$

In Abb. 6a haben beide Zustände gleiches statistisches Gewicht, dort gilt also $T_S = 2$. Jeder randomisierende Datengenerator mit K gleich gewichteten Zuständen hat nur eine einzige Zeitskala, nämlich

$$T_S = \frac{1}{1 - 1/K} = \frac{K}{K - 1} \,. \tag{2-14}$$

Da der triviale Fall $K = 1$ außer acht gelassen werden darf, gilt im randomisierten Fall immer $T_S \leq 2$. Mit anderen Worten, die Systemdynamik ist immer so schnell, das die Lerndynamik nicht ankoppeln kann.

Auch bei nicht-Markov'schen Systemen gibt es Korrelationszeiten und mittlere Lebensdauern. Aus Gründen der Anschaulichkeit werde ich mich jedoch auf einfache Systeme beschränken.

2.1.3 Phasenübergänge durch σ-*Annealing*

Die Phasenübergänge des Codebuchs, die in Abschnitt 1.1.2 anhand von Abb. 11 und 12 beschrieben wurden, sind Reaktionen des lernenden MVNN auf die Ausdehnungen der räumlichen Strukturen im Datensatz. Annealing- und der daraus resultierende Aufspaltungsprozess können so verstanden werden, dass der Datensatz auf verschiedenen Raumskalen σ „betrachtet“ (lokal gemittelt) wird, und dass jeweils diejenigen Strukturen neu „erkannt“ werden, deren Ausdehnung in der Größenordnung von σ liegt. Da σ deterministisch verkleinert wird, werden die kleineren Raumskalen nach den größeren entdeckt. In der Sequenz der Aufspaltungen wird dadurch eine *Hierarchie* von Modellen mit zunehmend feinerer Auflösung erzeugt. Für den ersten Phasenübergang müssen, wie schon mit Gleichung (1-50) von Rose, Gurewitz & Fox (1990) bewiesen, die größte Varianz des Systems, die als globales Moment berechnet wird, und der Parameter σ^2 des Lerners gleich sein. Die Gleichheit von *räumlichen Skalen* in System und Lerner (als solche möchte ich die beiden Varianzen hier verstehen) führt zum ersten Phasenübergang. Anschließend findet, wie auf den Seiten 18ff erklärt, eine Berechnung

von lokalen Momenten statt. Auch hier lässt sich eine ähnliche Gleichheit von zweitem lokalem Moment σ_S^2 des Systems und σ_{krit}^2 vermuten, die den nächsten Phasenübergang verursacht. Insgesamt ist der *univar*-Algorithmus auf randomisierten Daten also ein *Raumskalendetektor*.

Randomisierte Daten

Bereits bei trivialer Systemdynamik, etwa bei randomisierten Daten ($T_S \leq 2$), treten bei $\sigma = \sigma_S$ nichttriviale Effekte auf. Das *kritische Langsamwerden* dort ist eine bekannte Eigenschaft von Phasenübergängen zweiter Ordnung. Wird in einem *univar*-Training σ relativ schnell abgekühlt, so kommt es aufgrund des kritischen Langsamwerdens zu einer Verzögerung der Phasentrennung. Abbildung 21 zeigt diese *Retardierung* anhand des randomisierten Datensatzes aus Abb. 6a. Sie wird zusätzlich noch durch die Wahl des Lernparameters ε in der Lernregel (1-41) verstärkt, der bei kleinen Werten dazu führt, dass sich mit dem gesamten Lernverhalten des MVNN auch die Phasentrennung verzögert.

Man sieht, wie sich durch die verrauschten Daten ein leichtes Schwanken auf die Codebuchzentren überträgt und wie der Aufspaltungsprozess verzögert wird. Das optimale Aufspaltungsverhalten ist zum Vergleich gepunktet eingezeichnet. Der Parameter ε, der invers in die Relaxationszeit des Codebuchs nach den Gleichungen (2-6) bis (2-8) eingeht, ist mit dem Wert $\varepsilon = 0.05$ auf der einen Seite zu groß, das Rauschen verhindern zu können, auf der anderen Seite ist er jedoch zu klein, das Codebuch schnell genug aufspalten zu lassen. Um solche Retardierungseffekte zu vermeiden, muss der Abkühlprozess am Phasenübergang so langsam durchgeführt werden, dass das MVNN auch bei kleinem ε die Möglichkeit hat, das kritische Langsamwerden zu überwinden und in den Optimalzustand zu relaxieren. Die gepunktete optimale Kurve wurde auf diese Weise bestimmt.

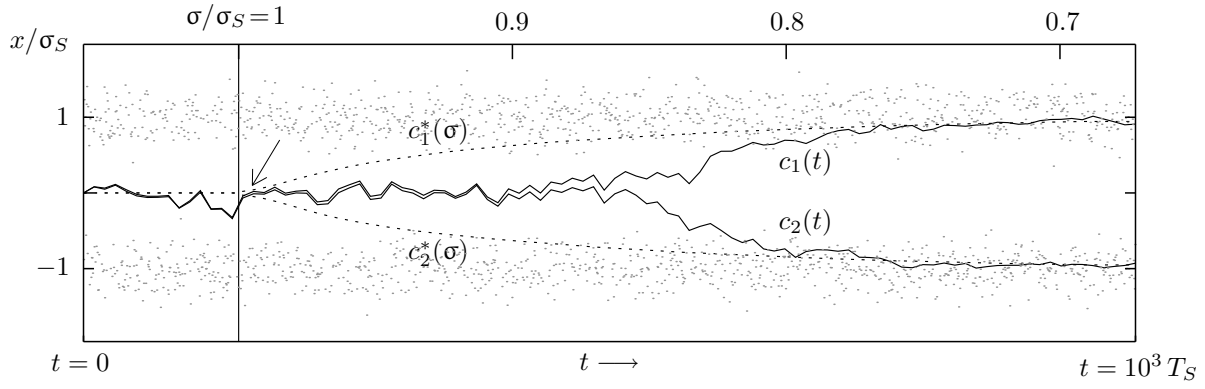


Abbildung 21: Rasches σ-*Annealing* bei randomisiertem Lernen ($T_S = 2$). Das Codebuch besteht aus zwei Zentren univariater Normalverteilungen. Bei relativ kurzem Lernprozess ($T = 2000 = 10^3 T_S$, $T_L = 40 = 20 T_S$, schwarze Linien) und dementsprechend schnellem Abkühlen wird die Quasi-Entartung der beiden Codebuchzentren zwar am erwarteten Punkt bei $\sigma = \sigma_S$ aufgehoben (Pfeil), die Relaxationszeit ist jedoch in der Nähe des Phasenübergangs zu groß, als dass die Codebuchzentren $\mathbf{c}_r$ bei der schnellen Kühlung ihre Gleichgewichtslagen $\mathbf{c}_r^*$ finden könnten. Das jeweils optimale Codebuch ($\mathbf{c}_r^*$, gepunktete Line), das genau bei der Standardabweichung σ_S aufbricht, wurde durch extrem langsames *Annealing* bestimmt.

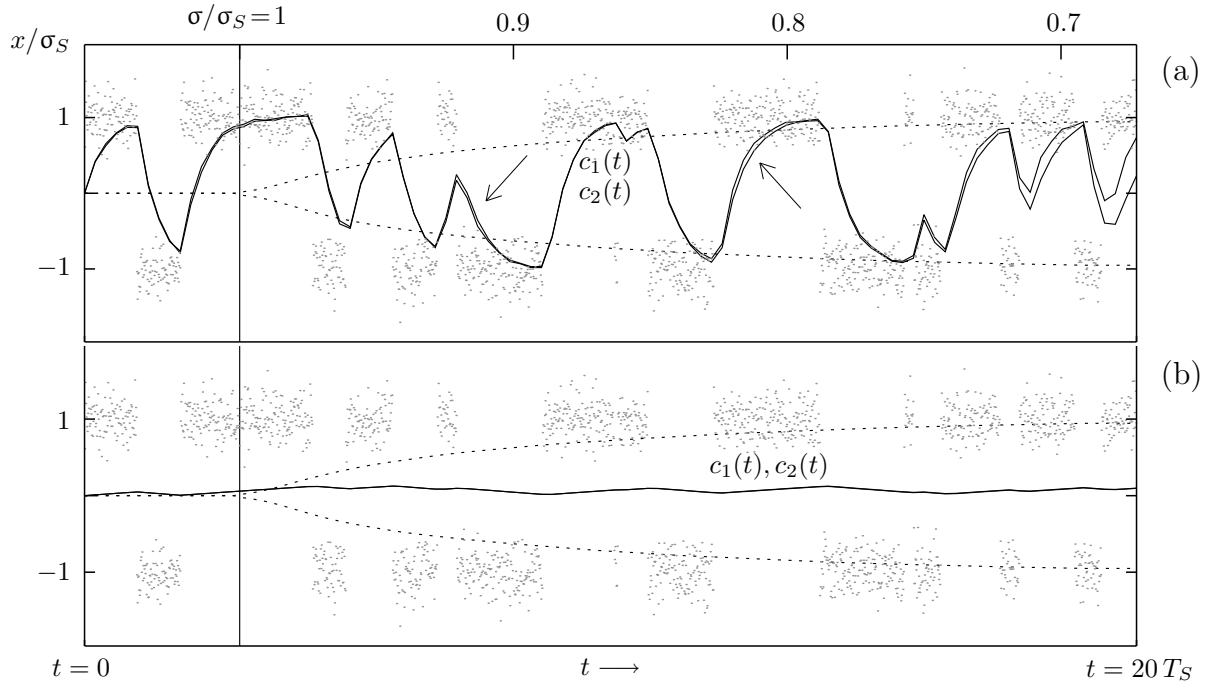


Abbildung 22: Rasches σ-*Annealing* bei on-line Lernen, dynamisch stark gekoppeltes und ungekoppeltes Codebuch. Verwendet wurde ein Datensatz des Markov-Prozesses aus Abb. 6b. Er besteht aus zwei Zuständen mit gleicher Gewichtung und hat nach (2-12) die Zeitskala $T_S = 100$. Wegen der langen Lebensdauern im System ist das Codebuch nicht in der Lage, den Phasenübergang zu vollziehen. Die oberen Kurven (a) wurden mit relativ großem Lernparameter berechnet, woraus sich Relaxationszeiten $T_r \approx T_L = 0.4\,T_S$ ergeben, sie sind also wesentlich kürzer als T_S, und das Codebuch ist so beweglich, dass es, auch wenn sein Optimalzustand längst aufgebrochen wäre, immer wieder an den Orten der Systemzustände quasi entartet. Dazwischen gibt es kurze Momente, an denen die Entartung aufgehoben wird (Pfeile). Erst bei $\sigma < 0.7\sigma_S$ sieht man den ersten Ansatz einer Aufspaltung. In (b) wurde die Dynamikkopplung durch eine wesentlich kleinere Lernrate aufgehoben, $T_r = 200\,T_S$, aufgehoben. Dafür wird das Codebuch jedoch viel zu unbeweglich, um während der kurzen Gesamtlerndauer von $T = 20\,T_S$ überhaupt aufspalten zu können.

On-line Daten

Ganz anders sieht die Situation bei zeitlich korrelierten Datenpunkten aus. Die zeitliche Entwicklung des Codebuchs beim Lernen des ursprünglichen Datensatzes ist in Abbildung 22 dargestellt. Dieser wurde von demselben Markovprozess erzeugt wie derjenige in Abb. 6b.

In Abb. 22a wurden wieder die gleichen Lernparameter ε und σ verwendet wie beim randomisierten Lernen oben in Abb. 21. Dementsprechend sind die Relaxationszeiten der $\mathbf{c}_r$ nach (2-6) wieder $T_r \approx 40$, nur sind sie diesmal kleiner als die Systemzeitskala, $T_r \approx 0.4\,T_S$, im Gegensatz zum randomisierenden Lernen, wo $T_r = 50\,T_S$ gilt. Hier wird das *Problem des on-line Lernens* deutlich, der Aufspaltungsprozess kann durch die langen Lebensdauern im Datenstrom nicht in der Form stattfinden, wie er beim randomisierten Lernen geschieht – und wie man es sich für den hierarchischen Aufspaltungsprozess wünschen würde. Solange nur ein Systemzustand gezeigt wird, laufen beide Neuronen dorthin, bis sie angekommen und quasi entartet sind. Beim anschließenden Wechsel des Systemzustandes entfernen sich die beiden zwar wegen

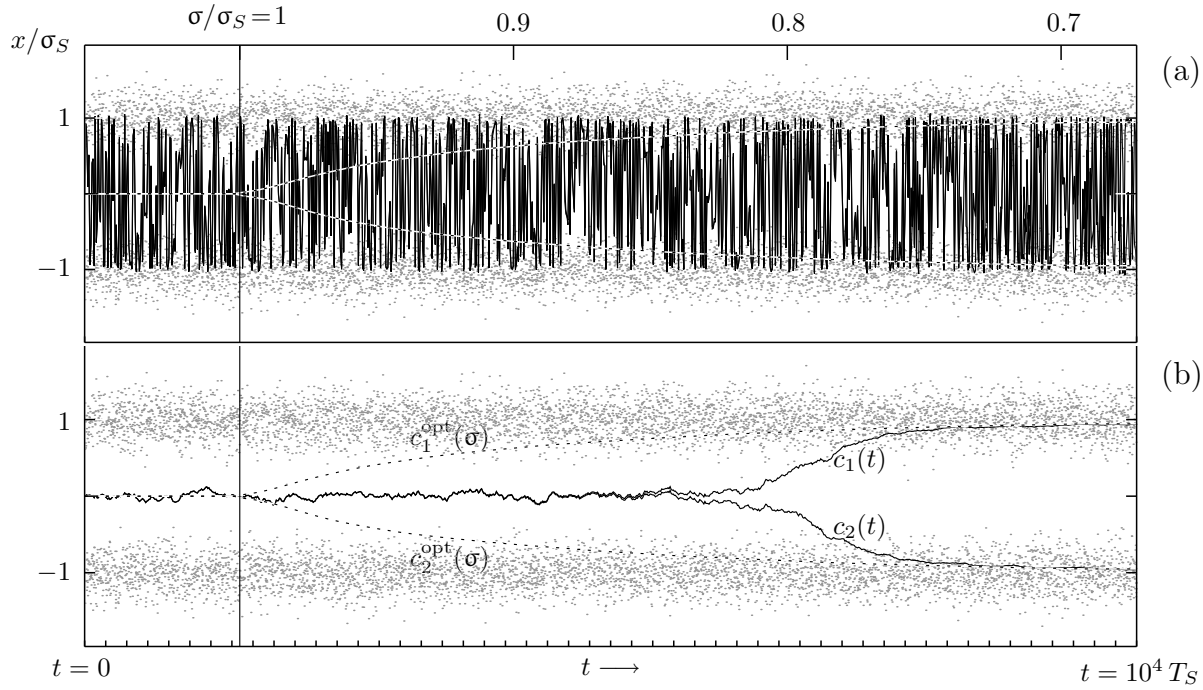


Abbildung 23: Langsames σ-*Annealing* bei on-line Lernen. Die hier gelernten Daten sind dieselben wie in Abb. 22, die Lebensdauern im System sind jedoch nicht mehr erkennbar, weil der Gesamtlernprozess mit $T = 10^6$ Datenpunkten sehr lang ist. Dadurch konnte viel langsamer abgekühlt werden, was in (b), wo die gleichen Lernparameter wie in Abb. 22b verwendet wurden, zu einem ähnlichen Verhalten führt wie bei dem randomisierten Lernen in Abb. 21. Offenbar kann bei ungekoppelter Dynamik durch Verlängerung der Gesamtlerndauer das Verhalten des randomisierten Lernens wiederhergestellt werden. Die Retardierung ist jedoch extrem (hier etwa $2.5 \cdot 10^5$ Punkte). In (a) dagegen, wo der Lerner stark an das Sytem gekoppelt ist, findet immer noch keine Aufspaltung statt. Hier wurde wieder $T_r = 0.4\, T_S$ verwendet. Aus Auflösungsgründen ist nicht zu sehen, dass gegen Ende des Lernprozesses wieder der erste Ansatz einer Aufspaltung vorhanden ist, wie in Abb. 22a auch.

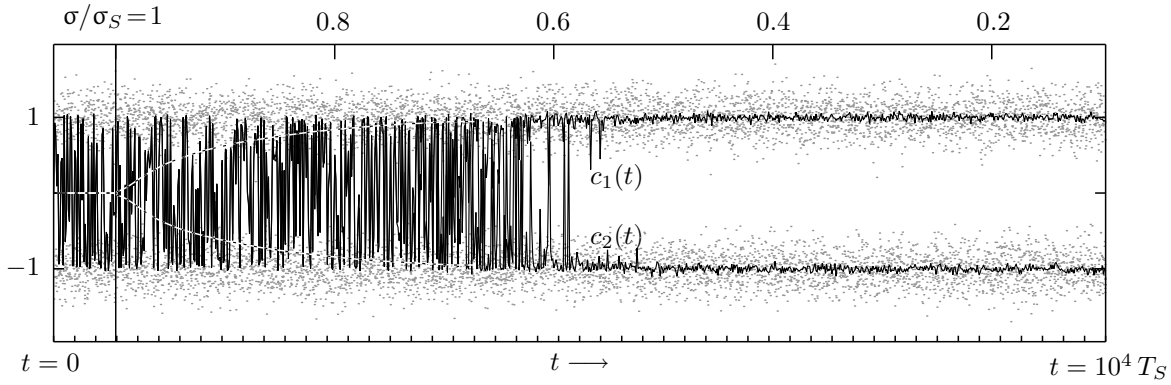


Abbildung 24: Langsames σ-*Annealing* bei on-line Lernen. Das Szenario ist dasselbe wie in Abb. 23a, wieder ist $T_r = 0.4\, T_S$. Um den Aufspaltungsprozess zu sehen, wurde jedoch noch weiter, bis $\sigma = 0.1\sigma_S$ abgekühlt. Die leichten Aufspaltungseffekte, die in Abb. 22a zu sehen waren, und die es auch in 23a gibt, führen hier zum deutlich sichtbaren Phasenübergang ($\sigma_{\text{krit}} \approx 0.65\sigma_S$).

der gegenseitigen Kompetition kurzfristig voneinander (z. B. bei den beiden Pfeilen in Abb. 22a), können sich aber wieder am Ort des neuen Systemzustandes sammeln. Auf diese Weise wird die Kompetition zwischen den beiden Neuronen immer wieder aufgehoben, und zwar auch dann, wenn σ eigentlich so klein ist, dass sie entsprechend der optimalen Kurve (wieder gepunktet eingezeichnet) bereits zu einer Aufspaltung geführt haben sollte. Dies ist in allen Bereichen von Abb. 22a mit $\sigma < \sigma_S$, also rechts vom senkrechten Strich der Fall, sieht man vom Bereich kleiner $\sigma < 0.7\sigma_S$ ab; hier scheint sich eine Aufspaltung abzuzeichnen.

Wir bezeichnen diesen Fall eines schnell relaxierenden Codebuchs, das sich an wechselnde Systemzustände anpassen kann, als „dynamisch gekoppelt."

Natürlich lässt sich dieser Fehler des Lernverfahrens, die dynamische Kopplung des Codebuchs an die Systemzustände, zunächst durch eine drastische Verkleinerung von ε beheben (Abb. 22b, ε ist um einen Faktor 500 kleiner als in 22a).[2] Dadurch kommt es jedoch wieder zu den erwähnten Retardierungseffekten. Mit $T_r = 200\,T_S$ ist die Verzögerung diesmal so stark, dass sie die Aufspaltung des Codebuchs während der kurzen Gesamtlerndauer von $T = 20\,T_S$ vollständig verhindert. Der Lernprozess müsste also entsprechend langsamer durchgeführt werden. Das Ergebnis eines 500-mal langsameren Annealingprozesses auf denselben Daten ist in Abbildung 23b zu sehen. Hier ist $T = 10^4\,T_S$, wodurch das Verhalten des Codebuchs als Funktion des Lernparameters σ ähnlich wird wie beim randomisierten Lernen in Abb. 21, wo $T = 10^3\,T_S$ gilt. Dieses Ergebnis zeigt, dass bei *dynamisch entkoppeltem* Lernen, bei der entweder ε klein oder T_S klein sein muss, die Retardierung durch ein kleineres ε problemlos durch langsameres σ-Abkühlen aufgehoben werden kann.

Der dynamischen Kopplung dagegen, die in 22a stattfindet, ist durch langsameres Annealing nicht beizukommen. Sie bleibt auch in Abb. 23a über weite Bereiche der σ-Werte bestehen. Wie in 22a bricht das Codebuch erst bei $\sigma \approx 0.7\sigma_S$ langsam auf, was jedoch wegen der geringen Auflösung in Abb. 23a nicht zu bemerken ist. Abbildung 24 zeigt wieder den Abkühlprozess aus Abb. 23a, er wird jedoch bis zu wesentlich kleineren σ/σ_S-Werten durchgeführt. Jetzt wird eine Aufspaltung sichtbar, und zwar bei $\sigma \approx 0.6\sigma_S$. Die dynamische Kopplung kann demnach durch weiteres Abkühlen aufgehoben und so ein Phasenübergang erzwungen werden. An dem in Abb. 24 dargestellten Verhalten $c_r(\sigma)$ ändert sich auch bei langsamerem Abkühlen nichts.

Die Verschiebung der Aufspaltung zu kleineren Werten σ/σ_S ist im dynamisch gekoppelten Fall ($T_r \approx 0.4\,T_S$) demnach kein Artefakt, wie es die Retardierung durch zu kleines ε ist, sondern eine echte Eigenschaft der Lerndynamik.

Der Phasenübergang, der bei $\sigma/\sigma_S = 1$ erwartet wird, tritt im Fall starker Kopplung immer bei zu kleinen Werten von σ/σ_S auf. Dies ist im Zusammenhang mit der Dichteschätzung durch *univar* nicht erwünscht, denn es kommt wesentlich auf den hierarchi-

[2] Ein ähnlicher Vergleich von Lernkurven mit großen und kleinen ε findet sich schon in Abb. 6b in der Einleitung. Dort wurde das Problem des on-line Lernens nicht am Phasenübergang demonstriert, sondern an der globalen Mittelwertsbildung. Jede einzelne lokale multivariate Geometrieoptimierung in der zweiten Phase eines *multivar*-Trainings weist ähnliches dynamisches Verhalten auf wie dasjenige in Abb. 6b.

schen Aufspaltungsprozess an, der als Raumskalendetektor fungiert. Wenn sich schon die erste Aufspaltung stark „verspätet“ hat, kann auch die hierarchisch nachgeordnete nächste Aufspaltung nicht stattfinden. Das Ergebnis ist dann ein teilweise entartetes, teilweise verteiltes Codebuch, für das keine vernünftige Eigenschaft im Sinne einer stationären Statistik (wie z. B. die *Load-balance*) mehr garantiert werden kann.

Zusammenfassend kann also festgestellt werden, dass das mögliche Auftreten einer dynamischen Kopplung von Lern- und Systemdynamik das zentrale *Problem des on-line Lernens* für den behandelten *univar*-Algorithmus ist. Nach den bisher diskutierten Beispielen ist die Kopplung dann zu erwarten, wenn $T_r \lesssim T_S$. Somit tritt sie bei on-line Lernen zwangsläufig auf, denn dort

- sind die im System enthaltenen Zeitskalen T_S a-priori unbekannt, und außerdem
- muss die Relaxationszeit des Lerners zunächst sehr kurz gewählt werden, um Retardierungseffekte bei der hierarchischen Analyse von räumlichen Datenstrukturen zu vermeiden.

Deswegen ist die dynamische Kopplung bei $T_r \lesssim T_S$ der erwartete Standardfall des on-line lernenden *univar*.

2.1.4 Phasenübergänge durch ε-*Annealing*

Das im letzten Abschnitt verwendete σ-*Annealing* ist bei on-line Lernen nicht die einzige Art, einen Phasenübergang zu provozieren. Es wurde argumentiert, dass die Dynamik des stochastischen Prozesses im Vergleich zur Beweglichkeit des Codebuchs zu langsam sei. In Abb. 23b wurde daraufhin die Beweglichkeit des MVNN durch einen

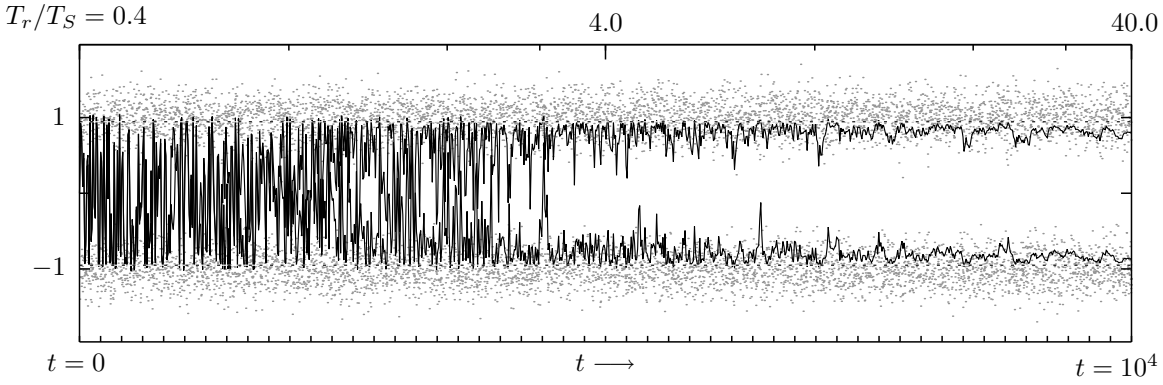


Abbildung 25: ε-*Annealing* beim on-line Lernen. Hier wurde vom Zustand der Dynamikkopplung (Abb. 22a) ausgegangen und nicht σ, sondern ε verkleinert (exponentiell von $5 \cdot 10^{-2}$ bis $5 \cdot 10^{-4}$). Dadurch wird die Lerndynamik sukzessive verlangsamt, von Relaxationszeiten $T_r/T_S = 0.4$ bis 40, und auf diese Weise die Kopplung aufgehoben. Der gesamte Lernprozess wurde bei $\sigma = 0.8\sigma_S$ durchgeführt, bei einer Raumskala also, wo das optimale Codebuch längst aufgespalten wäre. Sobald die Entkopplung der Dynamiken stattfindet, muss der Phasenübergang nachgeholt werden. Dies ist im Bild deutlich zu sehen.

viel kleineren Lernparameter ε stark reduziert, wodurch die dynamische Kopplung vermieden wurde ($T_r \gg T_S$) und sich wieder das Verhalten des randomisierenden Lerners einstellte.

Ferner wurde anhand des Beispiels aus Abb. 22a klar, dass das dynamisch gekoppelte Codebuch auch bei kleinen Parametern σ noch quasi entartet bleibt. Erst bei sehr kleinen Werten von σ/σ_S kann ein Phasenübergang stattfinden (Abb. 24). Statt σ noch weiter abzukühlen, soll nun alternativ versucht werden, das Aufbrechen des Codebuchs ausschließlich durch Abkühlen von ε zu erreichen. Dadurch wird, da $1/\varepsilon$ linear in die Lerndynamik eingeht, diese schrittweise verlangsamt, bis die Kopplung aufgehoben ist ($T_r \gg T_S$). Auf diese Weise erhält man in Abbildung 25 einen ähnlichen *dynamischen Phasenübergang*, wie er auch schon in Abb. 24 zu sehen war. Voraussetzung ist natürlich, dass σ kleiner als σ_S ist, damit überhaupt ein Phasenübergang stattfinden kann.

Der dynamische Phasenübergang in Abb. 24 und 25 findet vom dynamisch stark gekoppelten Zustand aus statt. Die *Entkopplung von Lern- und Systemdynamik* wird in beiden Beispielen durch die Phasentrennung verursacht. Gleichzeitig gilt umgekehrt, dass der Phasenübergang stattfinden kann, weil die Dynamik entkoppelt wurde. Aufspaltung und Dynamikentkopplung bedingen einander also gegenseitig. Der Übergang kann also auf zwei Weisen erreicht werden. Einerseits wird die Lerndynamik nach (2-8) auf natürliche Weise verringert, wenn ε verkleinert wird (Abb. 25); andererseits kann sie durch extremes σ-*Annealing* verlangsamt werden, was primär die Raumskala des Lerners verkleinert, im vorliegenden dynamikgekoppelten Fall aber ebenfalls als Reduzierung der Lerndynamik interpretiert werden darf, da die instantanen Relaxationszeiten (2-8) auch durch σ gesteuert werden. Dies soll im folgenden Abschnitt weiter ausgeführt werden.

2.1.5 Effektive Relaxationszeiten

Wie bereits oben bei den Gedächtniskernen angemerkt wurde, reichen die instantanen Relaxationszeiten $T_r(t)$ nicht aus, die effektive Lerndynamik bei der Verarbeitung von on-line Daten zu beschreiben. Bei zeitlichen Korrelationen im Datenstrom können die Gedächtniskerne $g_r(t,\cdot)$ nach der in Appendix A angegebenen Prozedur berechnet werden. Sie hängen von der gesamten bis zum Zeitpunkt t präsentierten Datensequenz und von den jeweiligen Codebuchkonfigurationen ab und können, wie wir an Beispielen sehen werden, auch ganz andere Formen als in Abb. 20 annehmen. Es stellt sich nun die Frage, wie man aus einer beliebigen Funktion $g_r(t,\cdot)$ die effektive Länge des Gedächtnisses bestimmen soll. Da jedes Gedächtnis, vom aktuellen Zeitpunkt beginnend, in die Vergangenheit reicht und prototypisch die Form einer Exponentialkurve hat, liegt es nahe, die Funktion $g_r(t,\cdot)$ durch eine Exponentialfunktion $s \mapsto \alpha \exp(-s/\tau)$ zu approximieren. Diese ist vollständig durch ihre beiden Parameter α und τ (nulltes und erstes Moment) bestimmt. Die Zerfallsrate τ dieser Funktion ist dann die *effektive* Relaxationszeit des Codebuchzentrums $\mathbf{c}_r$.

Auf der positiven reellen Halbachse $\mathbb{R}_+$ funktioniert die Approximation mit einer Ex-

ponentialfunktion ähnlich wie die Momentenentwicklung in Abschnitt 1.1. Das nullte (Normierung) und das erste Moment einer Funktion $g(t-t')$ (auf $\mathbb{R}_+$) sollen denjenigen der Exponentialfunktion gleich sein,[3]

$$\int_0^\infty g(s)\,ds \stackrel{!}{=} \int_0^\infty \alpha\,\exp(-s/\tau)\,ds = \alpha\tau, \quad \text{und} \tag{2-15}$$

$$\int_0^\infty g(s)\,s\,ds \stackrel{!}{=} \int_0^\infty \alpha\,\exp(-s/\tau)\,s\,ds = \alpha\tau^2. \tag{2-16}$$

Deswegen sind die Parameter zur exponentiellen Approximation durch folgende Integrale gegeben,

$$\tau = \int_0^\infty g(s)\,s\,ds \Big/ \int_0^\infty g(s)\,ds \quad \text{und} \tag{2-17}$$

$$\alpha = \frac{1}{\tau}\int_0^\infty g(s)\,ds. \tag{2-18}$$

Die Größe τ aus (2-17) ist die gesuchte effektive Länge des Gedächtniskerns $g_r(t,\cdot)$ und somit die effektive Relaxationszeit $\overline{T}_r$ des r-ten Codebuchzentrums. Sie umfasst die gesamte Vergangenheit des Lerners, ist also ein besseres Maß für die effektive Dynamik als die instantane Relaxationszeit $T_r(t)$. Allerdings verändern sich auch die $\overline{T}_r(t)$ mit der Zeit t, denn je nach Aktivierung $a_r(t)$ bekommt ein Neuron ein langes oder ein kurzes Gedächtnis.

Mit $\overline{T}_r(t)$ kann nun demonstriert werden, wie die oben angekündigte Verlangsamung der Lerndynamik durch das σ-*Annealing* verursacht wird, wenn sich das MVNN im Zustand stark gekoppelter Dynamik befindet. Dazu soll das Beispiel oben noch weiter vereinfacht werden. Es wird eine Datenquelle betrachtet, die nicht als Markovsystem schaltet, sondern deterministisch nach je T_S Datenpunkten. Die Zustände seien nicht verrauscht, sondern exakt (± 1). Dann ergeben sich Codebuchentwicklungen, wie sie in Abbildung 26 in der linken Spalte gezeigt sind. Sie alle sind idealisierte Ausschnitte aus Abb. 24, von oben nach unten für jeweils kleineres σ.

In Abb. 26a ist das Codebuch quasi entartet. Die Gedächtniskerne der beiden Neuronen sind identisch und haben die nach (2-6) erwartete Länge, nämlich $\overline{T}_r(t) = 40$. Die Gedächtniskerne sind nur für den letzten eingezeichneten Codebuchzustand dargestellt. Mit ihrer Länge $\overline{T}_r(t)$ reichen sie nur unwesentlich über jeweils einen Systemzustand hinaus, denn mit $T_S = 80 = 2\,\overline{T}_r(t)$ ist die Systemdynamik langsamer als die Lerndynamik. Dadurch ergeben sich die starken Schwankungen der Codebuchzentren, ähnlich wie in Abb. 22a.

Darunter, in Abb. 26b, bricht das Codebuch im Laufe seines Weges von einer Datenquelle zur anderen immer wieder deutlich auf. Die gleiche Situation konnte bereits an den beiden Pfeilen in Abb. 22a beobachtet werden, hier ist sie nun etwas besser sichtbar. Das kurzfristige Aufbrechen beruht auf der – im Vergleich zu Abb. 26a –

[3] Hier wird zur Vereinfachung der Notation das Argument von g umgedreht, $g(t-t') = g_r(t,t')$. Durch t' statt s ausgedrückt, wird $g_r(t,t')$ von $-\infty$ bis t integriert.

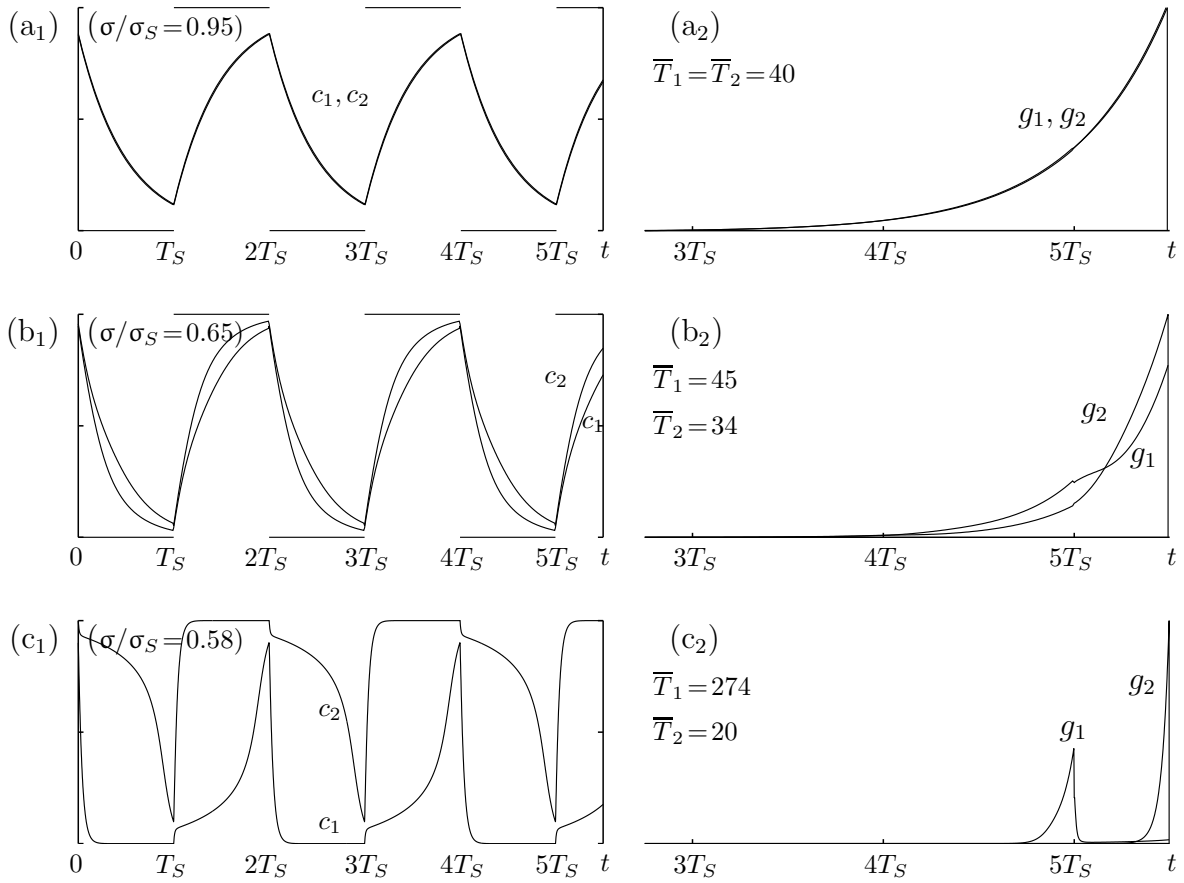


Abbildung 26: Codebuchentwicklungen und Gedächtniskerne für ein einfaches System. Gezeigt sind drei Ausschnitte eines dynamischen Phasenübergangs, ähnlich demjenigen in Abb. 24. Die Datenpunkte werden bei (± 1) präsentiert, immer abwechselnd T_S Punkte lang. Um die verschiedenen Lernverhalten bei ein- und demselben $\varepsilon = 0.05$ vorführen zu können, wurde von verschieden großen T_S ausgegangen. In (a) ist $T_S = 80$, in (b) $T_S = 120$ und in (c) $T_S = 530$. Durch die unterschiedlichen Zeiten und unterschiedliche Varianzen, $\sigma/\sigma_S = 0.95$, 0.65 und 0.58 von (a) bis (c), ergeben sich verschieden starke dynamische Kopplungen. In der obersten Reihe ist das Codebuch aufgrund starker Kopplung quasi entartet, obwohl sein Parameter eigentlich kleiner als σ_S ist. Die Lerndynamik ist, da beide Gedächtniskerne die Länge $\overline{T}_r(t) = 2/\varepsilon = 0.5 T_S$ haben, schneller als die Systemdynamik. Deshalb kann das Codebuch nicht aufbrechen.
In (b) ist die Lerndynamik immer noch schneller als die Systemdynamik, die beiden Gedächtniskerne haben Längen von $\overline{T}_r(t)/T_S = 0.28$ und 0.37. Zwar wird kurzfristig die Dynamik der einzelnen Zentren einigermaßen langsam, doch reicht dies nicht aus, um sich vom System abzukoppeln. Selbst bei dem gegebenen Wert $\sigma/\sigma_S = 0.65$ ist unklar, ob der Phasenübergang bereits stattgefunden hat.
In (c) ist schließlich ein System gegeben, dessen Lerndynamik mit $\overline{T}_r(t)/T_S = 0.04$ und 0.51 kurzfristig so schnell bzw. so langsam wird, dass ein Aufbrechen und somit eine teilweise Entkopplung sichtbar wird. Die beiden Codebuchzentren kommen sich nie mehr sehr nahe.

verstärkten Kompetition, es wurde $\sigma = 0.65\sigma_S$ verwendet. Trotzdem sorgen die langen Lebensdauern $T_S = 120$ im System dafür, dass sich die Codebuchzentren immer wieder nahe kommen. Die zugehörigen Gedächtniskerne sind in Abb. 26b$_2$ zu sehen. Beide haben ihr Maximum beim aktuellen Zeitpunkt t, wo ihr Wert gerade $\varepsilon a_r(\mathbf{x}_t)$ ist (vgl. Appendix A, Gl. (A-8)). Deswegen sind sie unterschiedlich hoch. Da sich die Aktivitätswerte schon vorher unterschieden haben, sind die Kerne auch unterschiedlich lang. Die Länge des höheren lässt sich nach (2-17) als $\overline{T}_2(t) = 34 = 0.28\,T_S$ angeben. Die mittlere Relaxationszeit des zugehörigen Zentrums liegt demnach zwischen der minimalen Relaxationszeit $1/\varepsilon$, die es hätte, würde es alleine lernen, und derjenigen die es im entarteten Fall hätte ($M/\varepsilon = 40$). Entsprechend länger ist der Kern des anderen Neurons ($\overline{T}_1(t) = 45 = 0.37\,T_S$). Beide Kernlängen sind kleiner als T_S, weswegen das Codebuch an das System gekoppelt bleibt.

Anders sieht die Situation bei weiter verkleinertem σ aus. Codebuchentwicklungen wie in Abb. 26c$_1$ stellen sich ein, wenn ein Phasenübergang aus dem dynamisch stark gekoppelten Bereich heraus geschieht, also sowohl in Abb. 24 als auch in Abb. 25. Hier scheint die Phasentrennung bereits stattgefunden zu haben, denn die beiden Codebuchzentren behalten immer relativ großen Abstand voneinander. Jeweils ein Neuron wird von dem anderen durch den kompetitiven a_r-Term stark verdrängt, es schafft den Weg zur Datenquelle erst nach sehr vielen Datenpunkten (hier ist $T_S = 530$). Die Gedächtniskerne bekommen nun völlig unterschiedliche Gestalten, der hohe schmale Kern g_2 hat eine Länge von $\overline{T}_2 \approx 20 = 1/\varepsilon = 0.04\,T_S$, dieses Neuron ist demnach so beweglich wie es nur sein kann. Der andere Gedächtniskern g_1 hat dagegen eine deutliche Lücke vor dem Zeitpunkt t, für den das Gedächtnis dargestellt ist. Nach Gleichung (2-17) hat er die mittlere Länge $\overline{T}_1 \approx 0.51\,T_S$. Sie scheint eher in die Größenordnung von T_S zu kommen als die Längen der Kerne aus Abb. 26b. Eine quantitative Analyse ist an dieser Stelle noch nicht möglich. Mit ihr beschäftigt sich der nächste Abschnitt.

Mit den Gedächtniskernlängen $\overline{T}_r(t)$ ist also eine Größe gegeben, die verständlich macht, warum der Effekt der dynamischen Kopplung auftreten kann. Wenn alle Gedächtniskerne so kurz sind, dass sie nur unwesentlich über eine Lebensdauer im System hinausragen, dann ist jedes Codebuchzentrum so beweglich, dass es sich immer wieder an einen aktuellen Systemzustand annähern muss. Erst bei längeren Kernen ist eine Mittelung über mehrere Systemzustände möglich. Wie sich die Länge der Kerne mit σ und dem mittleren Abstand der Codebuchzentren verändert, ist in Abb. 26 vorgeführt worden. Es ist demnach auch im Zustand starker dynamischer Kopplung möglich, bei sehr kleinem σ und leichter Aufspaltung im Codebuch so lange Gedächtniskerne zu bekommen, dass effektiv über mehrere Systemzustände gemittelt wird. Dadurch entfernen sich die Zentren noch weiter voneinander, und das Codebuch bricht auf.

Eine Verlängerung von Gedächtniskernen durch ε-Verkleinerung ist immer möglich, da ε invers in die zunächst definierten Lernzeitskalen T_L und $T_r(t)$ eingeht. Auch für die Gedächtniskerne gilt ähnliches, da sie eine Verallgemeinerung der Lernzeitskala T_L sind und im Spezialfall von entartetem Codebuch deren $1/\varepsilon$-Verhalten reproduzieren müssen (Abb. 26a). Auch im Fall einer Aufspaltung aus dem stark gekoppelten Bereich heraus (Abb. 26c) spielt die minimale Relaxationszeit $1/\varepsilon$ eine Rolle. Wenn sie länger gewählt wird, gleichen sich die beiden sehr unterschiedlichen Kernlängen $\overline{T}_1(t)$ und $\overline{T}_2(t)$ wieder an.

2.1.6 Phasendiagramme für on-line Lernen

Mit den in den letzten beiden Abschnitten gefundenen Aussagen zum Verhältnis von Lern- und Systemzeitskalen, kann nun versucht werden, den Einfluss der Parameter ε und σ auf den Phasenübergang zu untersuchen. Vor dem ersten Phasenübergang, ausgehend von einem vollständig entarteten Codebuch, lässt sich das Verhältnis $T_S/\overline{T}_r(t)$ leicht durch ε ausdrücken, denn es ist immer $\overline{T}_r(t) = M/\varepsilon = T_L$. Im folgenden wird dieses Verhältnis mit

$$\tau_S := \frac{\varepsilon\, T_S}{M} \tag{2-19}$$

abgekürzt. Analog wird die Zeit t auf der Skala M/ε gemessen,

$$\tau := \frac{\varepsilon\, t}{M}\,. \tag{2-20}$$

Abbildung 27 zeigt die Zusammenfassung der Ergebnisse aus den Abbildungen 21 bis 25. Auf der waagerechten Achse ist das Verhältnis σ/σ_S, auf der senkrechten der Logarithmus von τ_S aufgetragen. In den Abbildungen 21 bis 23 wurde nur σ verändert, was die drei waagerechten Pfeile zeigen, in Abb. 25 wurde bei konstantem σ ausschließlich ε abgekühlt. Die geschätzten kritischen Parameterwerte der Phasenübergänge sind mit einem Kreis markiert. Man sieht, dass der Phasenübergang nicht immer bei $\sigma = \sigma_S$ auftritt, sondern dass es einen funktionalen Zusammenhang zwischen den kritischen Parameterwerten σ_{krit} und $\varepsilon_{\text{krit}}$ gibt, der bei großen $\varepsilon_{\text{krit}}$-Werten zu kleineren σ_{krit} verläuft. Dies ist der Effekt der dynamischen Kopplung.

Wie dieser Zusammenhang genau aussieht, ist an den bisher behandelten vier Beispielen nicht zu ersehen. Ich werde im folgenden Abschnitt versuchen, ihn für ein paar prototypische und dennoch einfache Beispiele analytisch zu bestimmen.

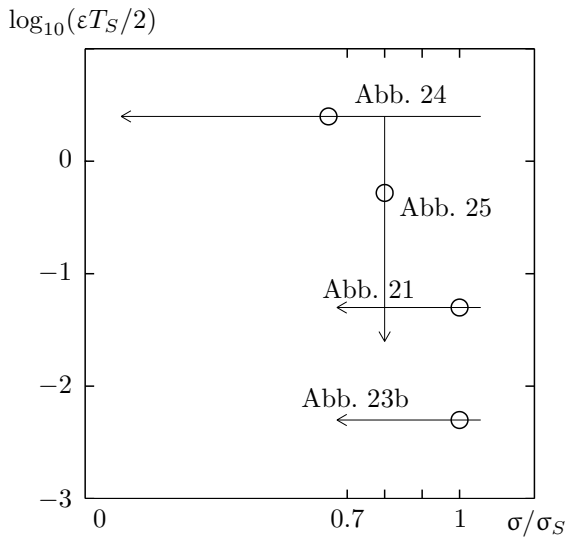


Abbildung 27: Darstellung des zeitlichen Weges der Lernparameter σ und ε in den Abbildungen 21 bis 25. Mit Kreisen sind die geschätzten kritischen Parameterwerte $(\sigma, \varepsilon)_{\text{krit}}$ markiert. Es kann ein funktionaler Zusammenhang zwischen den beiden vermutet werden.

2.2 Analytische Behandlung von prototypischen Spezialfällen

Im hierarchischen Aufspaltungsprozess in Abb. 11 findet jeder einzelne der Phasenübergänge für sich genommen relativ isoliert statt. Das ist durch Abb. 12c deutlich geworden. Bei jedem Phasenübergang bricht das Codebuch in zwei Teile auf, und in vielen Fällen kann davon ausgegangen werden, dass sich die beiden Teile anschließend kaum beeinflussen. Die darauffolgenden Übergänge finden also wieder etwa dieselben Bedingungen vor wie der erste. Das Verhalten beim ersten Phasenübergang ist dann prototypisch für alle weiteren, und es lohnt sich, ihn genauer zu betrachten.

Zu diesem Zweck soll nun das bereits in Zusammenhang mit Abb. 26 vorgestellte Modell eingehender diskutiert werden, da es das einfachste Beispiel für eine Datenfolge mit Clusterstruktur und ausgeprägten Zeitkorrelationen darstellt. Der Datensatz besteht nur aus zwei Werten ($x = \pm 1$), die jeweils T_S-mal nacheinander präsentiert werden. Er hat also genau eine Zeitskala T_S und eine Raumskala, die durch seine Standardabweichung $\sigma_S = 1$ gegeben ist. In einem *univar*-Lerner mit zwei Codebuchzentren kann es folglich nur einen einzigen Phasenübergang geben, der durch ein Paar von kritischen Parameterwerten $\left(\varepsilon T_S/2\right)_{\text{krit}}$ und $\left(\sigma/\sigma_S\right)_{\text{krit}}$ bestimmt ist.

Nach der analytischen Lösung dieses einfachsten Modells in drei Näherungen soll anschließend versucht werden, die gefundenen Aussagen auf Systeme mit den gleichen räumlichen, aber etwas allgemeineren zeitlichen Eigenschaften auszuweiten.

2.2.1 Drei analytisch lösbare Spezialfälle

Die analytische Bestimmung der kritischen Parameterwerte aus der Differenzengleichung (2-2) ist im allgemeinen Fall kaum möglich. Um dennoch quantitative Ergebnisse zu bekommen, muss man Näherungen einführen, die den zu untersuchenden Parameterbereich möglichst gut abdecken. Für das einfachste System aus zwei punktförmigen Datenquellen mit dem Zeitskalen-Verhältnis τ_S bieten sich drei Näherungen an, die in Abbildung 28 in der gleichen Reihenfolge visualisiert sind.

(a) Bei $\tau_S \ll 1$ ist ein sehr schnell schaltendes System oder ein sehr träges Codebuch gegeben. Lerner und System sind hier dynamisch entkoppelt, es gibt immer ein wohldefiniertes optimales Codebuch C^*, das wie die Datenquelle selbst vollständig symmetrisch ist.

(b) Bei langsameren Systemen oder flexibleren Codebüchern ($\tau_S \approx 1$) kommt es zur dynamischen Kopplung. Wie in Abb. 26b$_1$ gezeigt, sind sich zwei Codebuchzentren an ihren Umkehrpunkten am nächsten und entfernen sich dazwischen voneinander. Es wird hier notwendig, die gesamte Codebuchentwicklung während der Zeitspanne T_S zu beobachten und so den Abstand zum Zeitpunkt $(N+1) \cdot T_S$ aus dem Abstand zu $N \cdot T_S$ zu berechnen. Als Näherung ist hierzu notwendig, dass sich die Codebuchzentren nie weit voneinander entfernen.

(c) Bei sehr langen Lebensdauern T_S gilt auch $\tau_S \gg 1$, und man beobachtet das Verhalten aus Abb. 26c$_1$. Während fast der gesamten Zeitspanne T_S bewegt sich nur ein einziges Codebuchzentrum, das andere ist bereits an der Datenquelle angekommen und bleibt dort.

Die Grundlage der Berechnungen in diesem Abschnitt ist, dass die Lernregel (2-2) der Codebuchzentren durch die entsprechende Differentialgleichung (2-4) angenähert werden darf, was den Lösungsweg erheblich vereinfacht. Ich möchte sie hier in der äquivalenten Form

$$\frac{dc_r}{d\tau} = M a_r(x_\tau)\,\big(x_\tau - c_r(\tau)\big) \tag{2-21}$$

verwenden, wobei τ in (2-20) eingeführt wurde. Die Ersetzung der Differenzen- durch die Differentialgleichung ist genau dann gerechtfertigt, wenn die Schrittweiten Δc_r klein sind. Dies ist auf jeden Fall erfüllt, wenn

$$\varepsilon \ll 1, \tag{2-22}$$

das Codebuch also sehr unbeweglich ist ($T_L \gg 1$). In diesem Fall darf in allen drei Näherungen (a) bis (c) die Differentialgleichung verwendet werden. Für große ε stellt sich die umgekehrte Frage, für welche der drei Näherungen oben die Differentialgleichung (2-21) eine gute Approximation der Differenzengleichung (2-2) ist. An der Bedingung $\tau_S \ll 1$ des dynamisch entkoppelten Falls (a) sieht man, dass die Verwendung von (2-21) immer gerechtfertigt ist, denn dort gilt wegen $T_S \geq 1$

$$1 \gg \tau_S \geq \varepsilon/M. \tag{2-23}$$

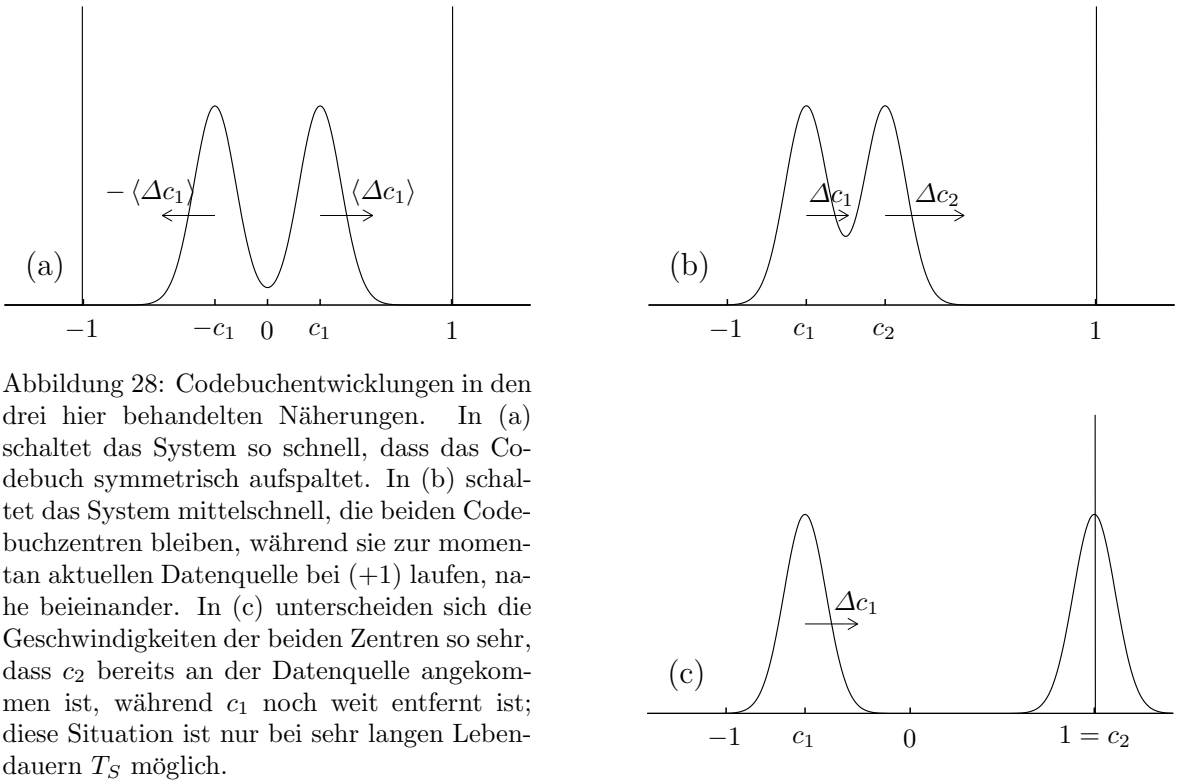


Abbildung 28: Codebuchentwicklungen in den drei hier behandelten Näherungen. In (a) schaltet das System so schnell, dass das Codebuch symmetrisch aufspaltet. In (b) schaltet das System mittelschnell, die beiden Codebuchzentren bleiben, während sie zur momentan aktuellen Datenquelle bei (+1) laufen, nahe beieinander. In (c) unterscheiden sich die Geschwindigkeiten der beiden Zentren so sehr, dass c_2 bereits an der Datenquelle angekommen ist, während c_1 noch weit entfernt ist; diese Situation ist nur bei sehr langen Lebendauern T_S möglich.

In den anderen beiden Fällen, (b) und (c), ist unter Umständen damit zu rechnen, dass bei großem ε *Diskretisierungseffekte* auftreten. Diese werden in Abschnitt 2.2.2 behandelt.

Approximation für sehr kleine τ_S

Sehr kleine Werte $\tau_S \ll 1$ bedeuten, dass während einer typischen Änderung von c_r, die ja auf der Zeitskala $T_L \gg T_S$ stattfindet, beide Datenquellen sehr oft präsentiert werden. Die Dynamik des Codebuchs ist also fast die gleiche wie bei einem dynamischen System, das in (2-21) nicht einzelne Punkte x *nacheinander* präsentiert bekommt, sondern *gleichzeitig* beide Punkte $x = \pm 1$. In einem solchen Lerner bleibt die Symmetrie des Codebuchs immer streng erhalten, und man kann von $c_2 = -c_1$ ausgehen. Die Differentialgleichung dieser Näherung mit zwei gleichzeitig feuernden Quellen lautet

$$\begin{aligned}\frac{d}{d\tau}c_1(\tau) &= 2a_1(1)\,(1 - c_1(t)) \;+\; 2a_1(-1)\,(-1 - c_1(t)) \\ &= -c_1(t) + \tanh\Big(\frac{c_1(t)}{\sigma^2}\Big),\end{aligned} \tag{2-24}$$

wobei ausgenutzt wurde, dass $a_1(-1) = a_2(1)$. Nullsetzen der Gleichung liefert das stationäre Codebuch C^*, das in Abbildung 29 dargestellt ist,

$$c_1^* = \tanh\Big(\frac{c_1^*}{\sigma^2}\Big) \quad \text{und} \quad c_2^* = -c_1^*. \tag{2-25}$$

Für $\sigma \leq 1$ hat diese Gleichung nur eine reelle Lösung, weshalb das stationäre Codebuch entartet ist. Für $\sigma > 1$ spaltet die Lösung in zwei stabile (durchgezogene Linien in Abb. 29) und einen instabilen Ast (gestrichelt) auf, es handelt sich also eine typische Stimmgabel-Bifurkation. Da $c_2 = -c_1$ angenommen wird, bestimmt sich der Phasenübergang im Lerner durch die Entartung der stabilen Attraktoren der Lerndynamik, was auf die altbekannte Bedingung führt,

$$\sigma_{\text{krit}} = 1 = \sigma_S. \tag{2-26}$$

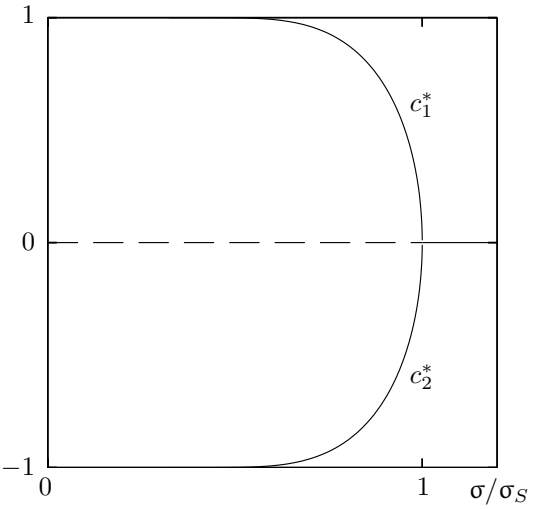


Abbildung 29: Das stationäre Codebuch für den Fall sehr kurzer Lebensdauern T_S bei sehr langsamer Codebuchbewegung ($\tau_S \ll 1$). Dargestellt sind die stabilen (durchgezogen) und die instabile Lösung (gestrichelt) der Näherung (2-25). Jedes Codebuch bewegt sich bei festem σ/σ_S symmetrisch auf die beiden stabilen Äste zu.

Die *Definition des Phasenübergangs* ist hier im dynamisch entkoppelten Fall also identisch mit derjenigen des randomisierenden *univar*: Er findet dort statt, wo das stabile stationäre Codebuch gerade nicht mehr entartet ist, ein on-line Lernproblem gibt es nicht.

Approximation für mittlere τ_S

Für größere Werte von τ_S kann nicht mehr von einer symmetrischen Aufspaltung ausgegangen werden. Beide Zentren bewegen sich nach der Differentialgleichung (2-21) mit unterschiedlicher Geschwindigkeit auf die gerade aktuelle Datenquelle zu, wie in Abbildung 28b durch verschieden lange Pfeile symbolisiert ist. Um den Phasenübergang zu bestimmen, wäre eigentlich die exakte Lösung des zugehörigen Anfangswertproblems erforderlich, die im allgemeinen Fall nicht in analytischer Form vorliegt.

Lediglich für den trivialen Fall, nämlich für entartete Codebücher lässt sich eine Lösung bestimmen. Sie besitzt einen Grenzzyklus, dessen Periode $2T_S$ dieselbe ist wie diejenige des Datenstroms (vgl. Abb. 26a$_1$). In Appendix B.1 werden in Gleichung (B-5) die Umkehrpunkte dieser Bewegung angegeben. In einem leicht aufgespalteten Codebuch, das sich in der Nähe des entarteten Grenzzyklus befindet, sind die Umkehrpunkte gleichzeitig diejenigen Punkte, an denen die Entfernung der Zentren während T_S *minimal* wird. Dazu sei nochmals auf die beiden Pfeile in Abb. 22a und auf Abb. 26b$_1$ verwiesen.

Um den Phasenübergang zu charakterisieren, nähert man sich ihm von der Seite aufgebrochener Codebücher. Je näher man ihm kommt, umso kleiner werden die Abstände, insbesondere die Minimalabstände während einer Periode. Solange ein gegebener kleiner Minimalabstand *stabil* ist, sich also nicht von Periode zu Periode verkleinert, bleibt die Phasentrennung erhalten. Wenn sich ein kleiner Abstand beliebig weiter verkleinert, befindet sich der Lerner auf der entarteten Seite der Phasenübergangskurve. Diese liegt also genau dort, wo *infinitesimal* kleine Codebuchabstände *stabil* bleiben. Es handelt sich hier also um eine *infinitesimale Definition des Phasenübergangs*. Übergänge nach dieser Definition werde ich vom *Typ I* nennen, im Unterschied zum grobskaligen *Typ II*, der weiter unten definiert wird.

Mit dieser Definition wird in Appendix B.1 der folgende Zusammenhang zwischen σ_{krit} und τ_S^{krit} gefunden,

$$\frac{\sigma_{\text{krit}}}{\sigma_S} = \left(1 + \tanh\left(\frac{\tau_S^{\text{krit}}}{2}\right)\right)\sqrt{\frac{1}{2\tau_S^{\text{krit}}}\Big(1 - \exp\big(-2\tau_S^{\text{krit}}\big)\Big)}\,. \tag{2-27}$$

Dabei wird angenommen, dass der Abstand der Codebuchzentren nicht nur an den Umkehrpunkten, sondern während der gesamten Periode klein bleibt (vgl. Abb. 26b$_1$ oder Abb. 51a). Dies ist, da die Abstände an den Umkehrpunkten ohnehin beliebig klein sind, keine sehr starke Einschränkung.

Abbildung 30 zeigt das durch (2-27) definierte Phasendiagramm. Man sieht deutlich die Abweichung des Phasenübergangs zu kleinen σ/σ_S-Werten, die schon in Abb. 27

zu bemerken war. Der senkrechte Verlauf bei $\log_{10}(\tau_S) \lesssim 0$ ist der Fall *ungekoppelter* Dynamik, der Phasenübergang liegt hier genau bei $\sigma = \sigma_S$. Der Bereich $\log_{10}(\tau_S) \gtrsim 0$, bei dem

$$\sigma_{\text{krit}} < \sigma_S \tag{2-28}$$

gilt, ist der Bereich der *dynamischen Kopplung*. Diese Form der Phasenübergangskurve ist der typische Fall für das on-line Lernen mit *univar*, wie wir im nächsten Abschnitt auch anhand einer Reihe numerischer Simulationen des behandelten Systems sehen werden. Die Kurve in Abb. 30 wird im Laufe dieses Kapitels noch einige Modifikationen und Korrekturen erhalten, ihr Aussehen jedoch im wesentlichen nicht mehr verändern. Sie ist somit das wichtigste Ergebnis dieses Kapitels.

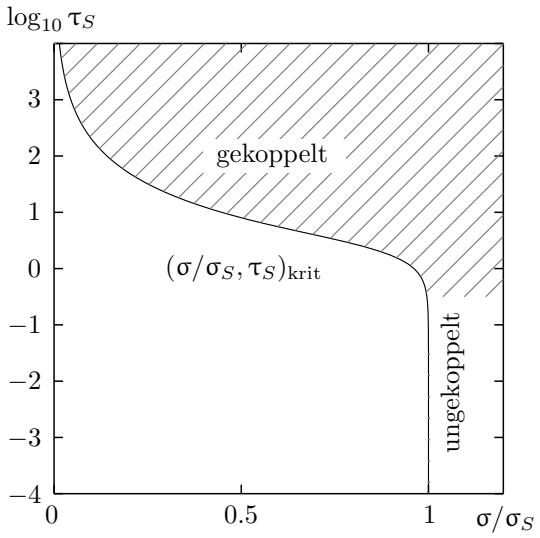


Abbildung 30: Phasendiagramm (Typ I). Die beiden Phasen sind durch die Kurve der kritischen Parameter entsprechend Gleichung (2-27) getrennt. Vorausgesetzt wurde, dass sich die Codebuchzentren während ihrer oszillierenden Bewegung nicht weit von einander entfernen.

Approximation für sehr große τ_S

Wenn das Verhältnis τ_S von System- zu typischer Lernzeitskala sehr groß ist, dann gelten die Näherungen, die zur Ableitung von (2-27) verwendet wurden, nicht mehr. Dies bedeutet, dass am oberen Teil der Kurve in Abb. 30 Korrekturen vorgenommen werden müssen. Bei großen τ_S-Werten wird die dynamische Kopplung so groß, dass sich die Codebuchzentren praktisch immer wieder am Zielpunkt versammeln können, auch wenn es zwischenzeitlich weit aufgespalten war. Dadurch ergibt sich die in Abb. $26c_1$ gezeigte Situation. Das näher liegende Zentrum läuft schnell auf die Datenquelle zu und verdrängt das andere stark. Erst nach langer Zeit kommt das zweite ebenfalls dort an.

Die passende Näherung für die Anfangsbedingung der Differentialgleichung (2-21) ist also, dass das erste Neuron bereits an der Quelle angekommen ist, während das andere noch etwa am Ort der alten Datenquelle sitzt, weil es stark verdrängt wird. Wie

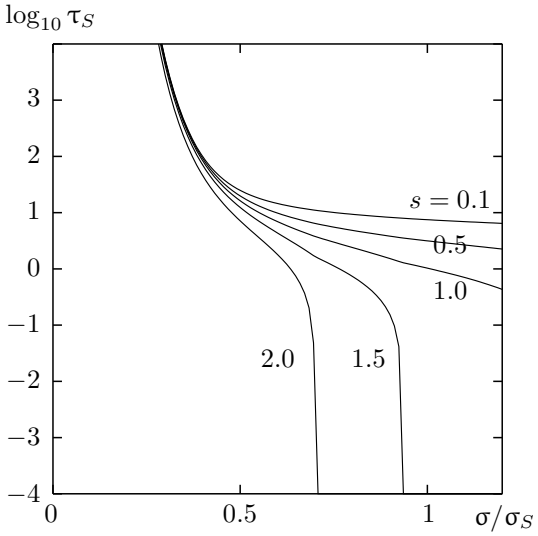


Abbildung 31: Phasendiagramm bei starker Dynamikkopplung (Typ II). Dargestellt sind die Kurven der kritischen Werte (τ_S, σ) nach Gleichung (2-29) für verschiedene Abstandsparameter $s=2$ (untere Linie) bis 0.1 (obere). Als um den Anfangsabstand korrigierte Breite wurde hier $\tilde{\sigma}=\sigma/1.4$ verwendet.

sofort aus Abb. 26c$_1$ ersichtlich, muss es dabei etwas vom alten Datenquellenort verschoben sein. Diese Verschiebung kann in den Berechnungen durch ein vergrößertes σ aufgefangen werden. Statt σ wird hier $\tilde{\sigma} = \sigma/|1 - c_1(0)|$ verwendet. Das Szenario in Abbildung 28c eignet sich für die Beantwortung der Frage, wie lange das zweite Neuron braucht, um sich beim anderen einzufinden. Mit Gleichung (B-21) in Appendix B.3 ist die Dauer τ_L angegeben, die das Neuron c_1 aus Abb. 28c für ein Stück dieses Übergangs benötigt (vgl. auch den Konturenplot in Abb. 53 dort).

Die Übergangsdauer kann direkt mit der Lebensdauer T_S verglichen werden. Ist sie kürzer, dann kann in der vorgegebenen Zeit die Annäherung der Zentren stattfinden, der Phasenübergang ist noch nicht erreicht. Die Anfangsbedingung in Abb. 28c kann in diesem Fall nicht in der umgekehrten Richtung für die nächste Lebensdauer T_S wiederholt werden, schließlich sind die Codebuchzentren dann fast entartet und vollziehen einen gemeinsamen Grenzzyklus. Ist die Übergangszeit dagegen länger als T_S, dann erreicht das nachfolgende Neuron das vorausgelaufene nicht (Abb. 26c$_1$). Sie behalten ihren großen Abstand in der Anfangsbedingung, und in der nächsten Periode ist wieder eines der beiden Zentren viel früher bei den Daten als das andere. In diesem Fall bleibt die Aufspaltung des Codebuchs erhalten.

Die Frage nach einem Kriterium für den Phasenübergang führt also auf eine *grobskalige* Definition (Typ II), da die Abstände in dieser Näherung während der gesamten Dynamik relativ groß bleiben müssen. Eine weite Aufspaltung zwischen zwei Wechseln im Systemzustand (vgl. Abb. 26c$_1$) ist durch den Anfangszustand in Abb. 28c fester Bestandteil der Näherung. Das Auffinden eines endlichen Abstandes, der während der gesamten Lerndauer nicht unterschritten wird, zeigt den Phasenübergang im stark dynamisch gekoppelten Regime an. Der Zusammenhang zwischen den kritischen Parametern wird in Appendix B.3 bestimmt. Dort wird ein Abstandsparameter s eingeführt und das Kriterium für den Phasenübergang so definiert, dass die beiden Zentren nach der Zeit τ_S gerade den Abstand $s\tilde{\sigma}$ haben sollen. Es werden die Kurven des Phasenübergangs gefunden, die durch

$$\tau_S(\tilde{\sigma}, s) = -\ln(s\tilde{\sigma}) + \frac{1}{4}\sum_{k=1}^{\infty} \frac{1}{k \cdot k!}\Big((2\tilde{\sigma}^2)^{-k} - (s^2/2)^k\Big) \qquad (2\text{-}29)$$

gegeben sind. Sie sind in Abbildung 31 für verschiedene Werte des Abstandsparameters s dargestellt. Wie man dort sieht, wird der genaue Wert von s im Bereich sehr großer τ_S irrelevant, weshalb man den oberen Teil der Kurve als gültige und richtige Aussage über den Phasenübergang akzeptieren muss. Dies ist gerade derjenige Bereich, für den das Modell konstruiert wurde.

Als eine erste Zusammenfassung ist in Abbildung 32 der Vergleich aller drei Näherungen mit Ergebnissen aus Simulationen dargestellt. Man sieht, dass die dort bestimmten Parameter sehr gut den theoretisch vorhergesagten Kurven folgen.

Die Simulationen wurden alle mit relativ kleinem ε durchgeführt ($\varepsilon = 0.01$), wodurch die Bedingung der Differentialnäherung (2-21) gut genug erfüllt ist. Um Retardierungseffekte zu vermeiden, musste sehr langsam abgekühlt werden. Die Phasenübergänge waren auch im dynamisch gekoppelten Bereich nach einer Mittelung der Codebuchzentren über einige Zyklen T_S eindeutig den kritischen Parametern $(\sigma, \varepsilon)^{\text{krit}}$ zuzuordnen. Die Mittelung konnte problemlos durchgeführt werden, da das *Annealing* noch wesentlich langsamer war als es für die ε-Retardierungseffekte notwendig gewesen wäre (10^7 Datenpunkte im unteren Teil der Kurve, wobei σ/σ_S von 1.1 bis 0.2 linear variiert wurde, 10^8 Datenpunkte für den oberen Teil, mit $\sigma/\sigma_S = 0.6$ bis 0.2).

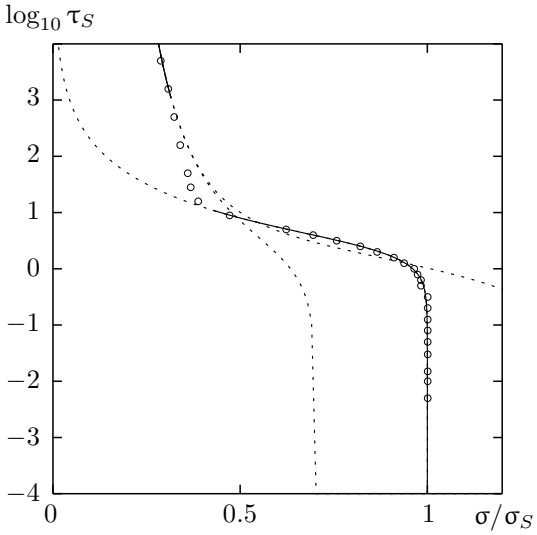


Abbildung 32: Vergleich der in Simulationen sichtbaren Phasenübergänge mit den analytischen Ergebnissen. Die Kurven aus den Abb. 30 und 31 (für $s = 2$ und 0.1) sind in dem Bereich guter Übereinstimmung schwarz eingezeichnet, dort wo die zugrundeliegenden Näherungen nicht gelten, gepunktet. Das Annealingverfahren in den Simulationen fand in allen Fällen sehr langsam statt, mit einer Gesamtlerndauer von 10^7 bzw. 10^8 Datenpunkten.

2.2.2 Korrektur durch Diskretisierungseffekte

Bei der analytischen Lösung der oben behandelten Fälle wurde davon ausgegangen, dass die Bedingung (2-22) gilt, und dass man die *univar*-Lernregel durch die Differentialgleichung (2-21) annähern darf. Wie bereits besprochen, ist das im Fall $\tau_S \ll 1$

immer möglich, während es in den anderen Fällen zu Diskretisierungseffekten kommen kann.

Bei großen $\tau_S \gg 1$ wird die *grobskalige* Definition (Typ II) des Phasenübergangs verwendet, die von dem Minimalabstand $s\tilde{\sigma}$ aus (2-29) abhängt (vgl. wieder Abb. 26c$_1$). Durch die Diskretisierung werden lediglich kleine Fluktuationen auf diesen Abstand übertragen. Dabei wird die Phasenübergangskurve jedoch nicht verändert, denn bei $\tau_S \gg 1$ wird der genaue Wert des Codebuchabstandes am Umkehrpunkt unwichtig, wie wir an Abb. 31 sehen konnten.

Es bleibt also nur der Bereich mittlerer Zeiten $\tau_S \approx 1$, wo die *infinitesimale* Definition (Typ I) des Phasenübergangs verwendet wird, in dem Diskretisierungseffekte eine Rolle spielen können. Dort kann es vorkommen, dass große Werte von ε auf kurze Lebensdauern T_S treffen (etwa $T_S = 5$ und $\varepsilon = 1/5$). In solchen Fällen ist die Annahme (2-22) nicht mehr gültig, und man muss Gleichung (2-27), welche die Abhängigkeit der kritischen Parameter voneinander beschreibt, durch ihre diskrete Version ersetzen,

$$\frac{\sigma_{\text{krit}}(T_S, \varepsilon)}{\sigma_S} = \left(1 \,+\, \frac{1-(1-\varepsilon/2)^{T_S}}{1+(1-\varepsilon/2)^{T_S}}\right) \sqrt{\frac{1}{T_S}\;\frac{1-(1-\varepsilon/2)^{2T_S}}{1-(1-\varepsilon/2)^2}} \quad . \tag{2-30}$$

Diese Gleichung wird in Appendix B.2 durch Übertragung der Überlegungen zu dem kontinuierlichen auf den diskreten Fall gefunden.

Abbildung 33 zeigt die Kurven kritischer Parameter für einige Kombinationen von ε und T_S. In (a) ist für drei verschiedene Werte von T_S der Lernparameter ε bis zu seinem

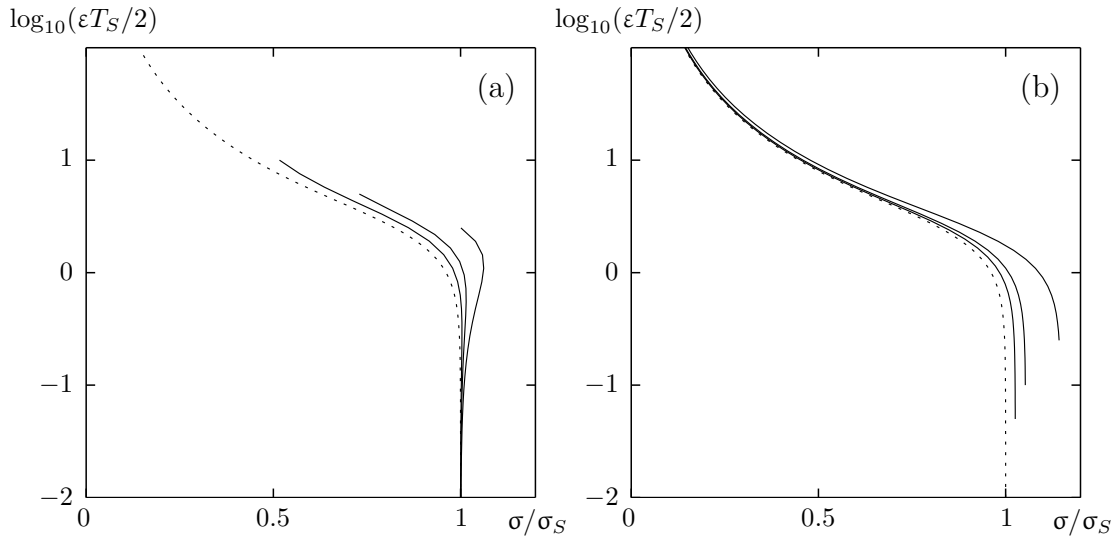


Abbildung 33: Die kritischen Parameterwerte nach Gleichung (2-30). In (a) für drei feste Lebendauern $T_S = 20$, 10 und 5 (von links nach rechts) bei kontinuierlicher ε-Variation, $\varepsilon \in (0, 1]$. Es handelt sich also um drei Systeme mit jeweils fester Lebensdauer, die von unterschiedlich beweglichen Lernern betrachtet werden. Je größer τ_S wird, umso beweglicher sind sie. In (b) sind drei verschieden flexible Lerner dargestellt, $\varepsilon = 0.1$, 0.2 und 0.5 (von links nach rechts), die sich in verschieden schnell schaltenden Umwelten befinden ($T_S \in [1, \infty)$, kontinuierlich variiert). Die gestrichelte Kurve ist diejenige aus Abb. 30.

Maximalwert $\varepsilon = 1$ dargestellt (obere Enden der Kuven). In (b) ist andersherum für verschiedene ε die Lebensdauer im System bis zu ihrem Minimalwert $T_S = 1$ variiert worden. Eine Phasenübergangskurve, die sich über den ganzen dargestellten Bereich $\log_{10}(\tau_S) \in (-2, 2)$ erstreckt, muss Eigenschaften beider Bilder in sich vereinen. Etwa bei $\tau_S \approx 1$ bekommt sie dann Ausbuchtungen zu großen σ/σ_S-Werten. Dieser Effekt muss als *Korrektur durch die diskrete Natur der Lernregel* in die Beschreibung des Phasenübergangs aufgenommen werden. In Abbildung 32 ist dieser Effekt nicht mehr sichtbar, weil dort ε mit 10^{-2} klein genug ist.

2.2.3 Berücksichtigung unterschiedlicher Gewichte der Datenquellen

Das bisher diskutierte Modell war auf einfache Weise symmetrisch, weil die beiden Punktquellen gleiches statistisches Gewicht und deshalb auch die gleiche Lebensdauer hatten. Die nächstkompliziertere Beschreibung des Phasenübergangs sollte aber verschiedene Zeitskalen berücksichtigen, denn erst dort kann untersucht werden, wie der Lerner auf die Kombination mehrerer Zeitskalen im System reagiert.

Das einfachste System mit mehreren Zeitskalen besteht wieder aus zwei Punktquellen, nur diesmal mit unterschiedlichen statistischen Gewichten P_A und P_B. Dies hat zur Folge, dass man bei deterministischem Schalten zwei verschiedene Lebensdauern bekommt, aber nur eine Periode. Es ist also nicht unmittelbar klar, ob man hier schon von zwei Zeitskalen sprechen kann. Als typische Zeitskala T_S wird die halbe Periode verwendet.

Ähnliche Annahmen und Rechenschritte wie oben bei mittleren Werten für τ_S und wie

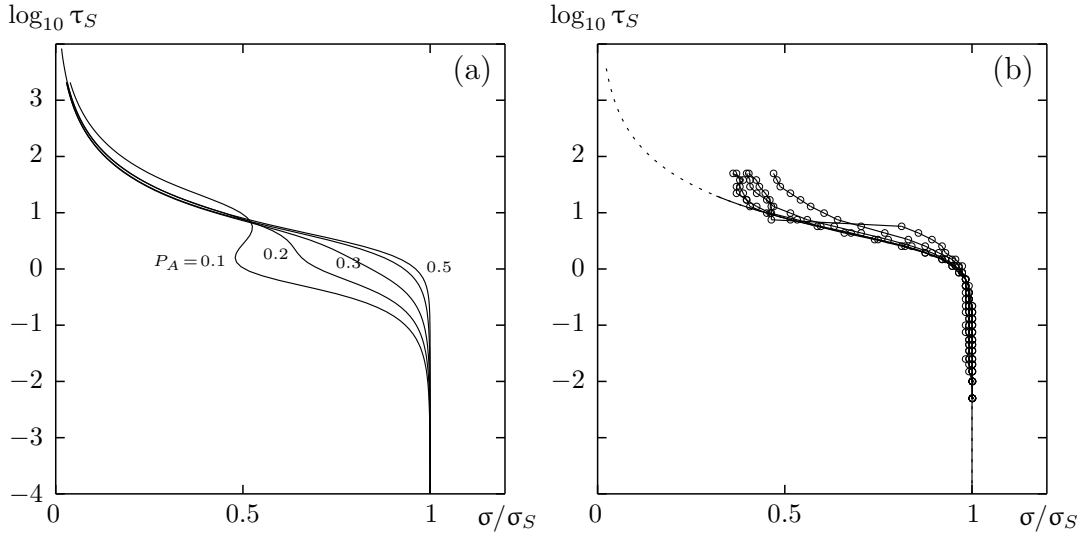


Abbildung 34: Kritische Werte (τ_S, σ) für verschiede Gewichte $P_A = 0.5, 0.4, 0.3, 0.2, 0.1$. In (a) aus analytischen Berechnungen, in (b) aus Simulationen der gleichen Systeme. Man sieht, dass hier die Simulationsergebnisse nicht mit den erwarteten theoretischen übereinstimmen. Verwendet wurde hier wieder $\varepsilon = 0.01$ und ein sehr kleines Rauschen $|R| = 5 \cdot 10^{-4}$ Dieses einfache Modell aus zwei Zuständen eignet sich demnach nicht, um die Effekte zu demonstrieren, die bei mehreren Zeitskalen auftreten.

in Appendix B.1 führen zu den kritischen Parameterkurven nach Typ II, die in Abbildung 34a dargestellt sind (die Berechnungen sind hier nicht noch einmal angeführt). Im Bereich $\tau_S \ll 1$ haben alle Kurven die gleiche Form wie die ursprüngliche Kurve ($P_A = P_B = 0.5$). Der Bereich der dynamischen Kopplung ($(\sigma/\sigma_S)_{\text{krit}} < 1$) dagegen rutscht für kleine P_A weiter zu kleinen τ_S, wird also bei stark unterschiedlichen Gewichten größer.

Der Vergleich mit Ergebnissen der Simulationen in Abb. 34 zeigt jedoch, dass die Unterschiede der Kurven in Abb. 34a nur bei der analytischen Behandlung auftreten, die Phasenübergänge der simulierten Systeme fallen in weiten Parameterbereichen zusammen. Es ist mir nicht gelungen, Simulationsparameter (insbesondere bei einem nicht zu kleinen Rauschen $\mathbf{R}$ in (1-41)) zu finden, die das Verhalten in 34a reproduzieren. Man kann demnach davon ausgehen, dass der Lerner im simulierten System die *gleiche effektive Zeitskala* sieht wie in einem System aus gleichen Gewichten.

2.2.4 Verhalten bei nichtdeterministischem Schalten

Bisher wurde in den analytischen Untersuchungen angenommen, dass beide Quellen genau T_S Datenpunkte hintereinander liefern, das Schaltverhalten also deterministisch war. Dass sich die Ergebnisse auch auf stochastisch schaltende Systeme übertragen lassen, soll nun an einem Beispiel vorgeführt werden. Dabei wird als Datenquelle ein Markovprozess angenommen, der zwischen beiden Punktquellen hin- und herschaltet. T_S ist wieder durch Gleichung (2-12) bestimmt.

Abbildung 35 vergleicht Simulationsergebnisse bei Markov-artigem Schalten mit denen bei deterministischen Schalten. Wie in Abb. 32 wurde sehr langsam abgekühlt und der erste Phasenübergang bestimmt.

Wie man sieht, wird der fast stufenförmige Verlauf der Phasenübergangskurve des deterministischen im Markov-Fall stark geglättet. Die Lebensdauern der Datenquellen gehorchen hier einer Poissonverteilung mit Mittelwert und Varianz T_S, es tritt also nicht immer die gleiche Lebensdauer auf. Daher wird das zeitliche Systemverhalten, das schließlich zum Phasenübergang führt, über viele momentane Lebensdauern gemittelt, was die Glättung qualitativ erklärt.

2.3 Diskussion: Gegenseitige Abhängigkeit von Zeit- und Raumskalen

Wie anhand des einfachen Beispiels oben gezeigt wurde, hängt die Phasenübergangskurve sowohl von σ/σ_S, als auch von τ_S ab. Es war jeweils nur eine einzige Raum- und Zeitskala gegeben, im allgemeinen Fall besitzt ein Datengenerator jedoch viele Raum- und Zeitskalen, die in die Phasenübergänge eines *univar*-Lernprozesses eingehen. Je nach verwendetem stochastischem Prozess sieht die Kurve dann anders aus (beim Markovprozess anders als beim determinsistischen Schalten, Abb. 35). Von den vielen

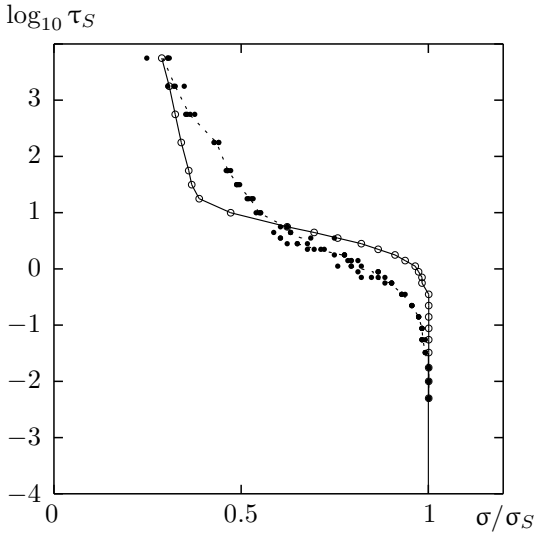


Abbildung 35: Vergleich der Phasenübergänge aus Simulationen von deterministisch nach T_S Schritten schaltenden Systemen (○) mit denjenigen von Markov-artig schaltenden Systemen mit mittlererer Lebensdauer T_S (•).

Raum- und Zeitskalen ist nicht klar, welche nun wie stark zum Aufbrechen des Codebuchs beitragen. Das *univar*-lernende MVNN mittelt schließlich sowohl in räumlicher wie auch in zeitlicher Hinsicht. Dadurch entsteht eine gegenseitige Abhängigkeit von kritischen Raum- und Zeitskalen, die i. a. sehr kompliziert ist. Dennoch können schon an den behandelten einfachen Beispielen einige Grundprinzipien festgestellt werden, die zu dieser gegenseitigen Abhängigkeit der Skalen führen. Die logische Reihenfolge der Skalen in einem *univar*-Training ist die folgende:

1. σ und ε werden vorgegeben und deterministisch abgekühlt. Mit σ hat man die Raumskala des Lerners fest vorgegeben, mit $T_L = M/\varepsilon$ jedoch eine zeitliche Größe, die ausschließlich vor dem ersten Aufbrechen als typische Lernzeitskala interpretiert werden darf.

2. Die räumlichen Skalen σ_S sind zwar alle im System fest vorgegeben, doch stellt man bei Daten mit echtem Rauschen schnell fest, dass eine Vielzahl räumlicher Größenordnungen vorkommt, von denen jedoch nicht alle im Sinne einer Clusterbildung *relevant* sind. Einem Lernalgorithmus, der diese Skalen detektieren soll, bleibt nun nichts anderes übrig, als bei der größten anzufangen und nacheinander alle Raumskalen durchzugehen. Im Fall von schaltenden Systemzuständen spielt aber – wie in den letzten Abschnitten ausführlich dargelegt – bei der Raumskalendetektion auch das Verhältnis von System- zu Lernzeitskala, d. h. $\tau_S^{\text{eff}} = T_S^{\text{eff}}/T_L^{\text{eff}}$ eine Rolle.

3. Die Lernzeitskala T_L^{eff} wird in nullter Näherung als $T_L = M/\varepsilon$ angegeben. Treten dagegen zeitliche oder räumliche Strukturen auf, dann findet man unterschiedliche Kompetition der einzelnen Neuronen und nichttriviale Gedächtniskerne (Abbildung 26). Allgemein ist also die Zeitskala T_L^{eff} des Lerners ein schwer zu definierender Begriff. Fest steht, dass jede sinnvolle Definition vom aktuellen Codebuch

und sowohl von den Raumskalen σ_S des beobachteten Systems, als auch von σ, der Raumskala des Lerners abhängen muss.

4. Es hängen also sowohl die relevanten Raumskalen von den Zeitskalen ab, als auch umgekehrt. Dennoch sind die Raumskalen in gewisser Weise grundlegender, denn sie können grundsätzlich auch im dynamisch entkoppelten Fall bestimmt werden. Im Gegensatz dazu gilt, dass in stochastischen Systemen niemals Zeitskalen ohne Raumskalen definiert werden können. Das ist leicht am Beispiel eines kontinuierlichen Markovprozesses einzusehen. Analog zu (2-12) ist die erwartete Lebensdauer einer Teilmenge Δ des Merkmalsraumes gerade

$$T_{S,\Delta} = \frac{1}{1 - P(\Delta|\Delta)}, \tag{2-31}$$

mit $P(\Delta|\Delta)$ der bedingten Wahrscheinlichkeit, dass das System von einem Zustand in Δ wieder in einen solchen Zustand springt. Man sieht, dass die Lebensdauer einer Menge (eines Clusters) Δ von seinem statistischen Gewicht, aber auch von seiner räumlichen Ausdehnung abhängt. Der größtmögliche „Cluster“, der aus dem gesamten Merkmalsraum besteht, hat eine unendliche Lebensdauer, denn $P(\mathcal{M}|\mathcal{M}) = 1$. Je weiter $\mathcal{M}$ in immer kleinere Partitionen unterteilt wird, umso kürzer werden die Lebensdauern, da $P(\Delta|\Delta) \to 0$ für $|\Delta| \to 0$,

$$T_\Delta \in [1, \infty). \tag{2-32}$$

Das MVNN führt auf dem Merkmalsraum in hierarchischer Weise genau so eine unscharfe Partitionierung durch und bekommt dementsprechend immer kürzere Lebensdauern zu sehen. Auf jeder Hierarchiestufe bildet sich eine effektive Systemzeitskala T_L^{eff} aus, die von der Clustereinteilung, also vom aktuellen Codebuch und von σ_S abhängt. Sie ist implizit also auch von σ und ε bzw. T_L^{eff} abhängig. Es ist jedoch nicht klar, wie diese effektiven Systemzeitskalen aus der Beobachtung des Lernprozesses und der auftretenden Phasenübergänge gemessen werden können.

In diesem Kapitel wurde der Ort des ersten Phasenübergangs im Parameterraum $(\sigma/\sigma_S, \tau_S)$ für einfach schaltende Datenquellen bestimmt (siehe Abb. 32). Da dort die Raumskala des beobachteten Systems durch einen dynamisch entkoppelten Phasenübergang ($\tau_S \to 0$ wegen $\varepsilon \to 0$) zugänglich ist, kann die Systemzeitskala T_S durch die Wiederholung des Phasenübergangs im dynamisch gekoppelten Bereich gemessen werden. Dazu ist nicht unbedingt notwendig, dass der Phasenübergang wieder von der entarteten Seite ausgehend stattfindet, es wäre genausogut möglich, andersherum zu verfahren.

Abbildung 36 zeigt das Vorgehen: Weg (1) dient der Bestimmung von σ_S im entkoppelten Fall. Da T_S nicht a-priori bekannt ist, müssen zusätzlich zur Vorgabe eines kleinen Lernparameters die Schwankungen des Codebuchs beobachtet werden, um sicherzustellen, dass es sich nicht im dynamisch gekoppelten Bereich befindet. Weg (2) dient einem Zeitskalen-*Scan*. Das Verhältnis von System- zu Lernzeitskala verändert sich so lange, bis ein Phasenübergang stattfindet. Da der Zusammenhang zwischen

den kritischen Parametern nach den Gleichungen (2-27) und (2-29) bekannt ist, kann aus den gefundenen kritischen Parametern $(\sigma/\sigma_S, \varepsilon)^{\text{krit}}$ sofort die Systemschaltzeit T_S bestimmt werden.

Dies liefert zwar ein operatives Verfahren zur Schätzung von Zeitskalen, jedoch ist Weg (1) wegen der starken Retardierung bei eventuell sehr großen T_S für on-line Zwecke nicht geeignet. Vielmehr wird hier ein Lerner benötigt, der zwar aufmerksam ist (ε genügend groß), aber dennoch nicht an jede Systemdynamik koppelt, der also noch zur Raumskalendetektion verwendet werden kann. Wir wenden uns nun dem Problem zu, wie bei einem aufmerksamen Lerner die dynamische Kopplung verhindert werden kann. Dazu verwenden wir die Idee aus Abb. 7, in der nicht alle präsentierten Datenpunkte gelernt werden, und beschäftigen uns zu ihrer Realisierung zunächst mit dem Verhalten der Meeresschnecke *Aplysia*.

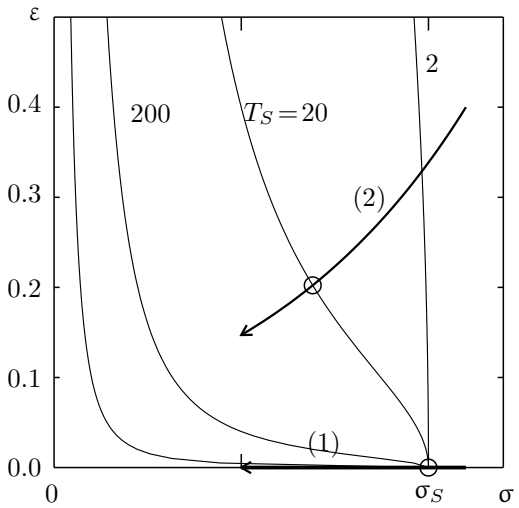


Abbildung 36: Zeitlicher Weg zweier *Annealing*-Verfahren im Raum der Lernparameter ε und σ (diesmal in linearer Darstellung). Eingezeichnet sind ebenfalls die kritischen Parameterkurven nach (2-27) für einige Schaltzeiten T_S. Der Weg (1) dient der Bestimmung von σ_S und muss dazu durch sehr kleines ε die dynamische Entkopplung von Lerner und System gewährleisten. Weg (2) muss bei so großen ε verlaufen, dass sein Phasenübergang (Kreis) im dynamisch gekoppelten Bereich stattfindet. Der Lerner verändert seine Parameter kontinuierlich, bis ein Phasenübergang die *Detektion einer Zeitskala im System* anzeigt (hier $T_S = 20$).

Kapitel 3

Neuronale Gewöhnung in *Aplysia californica*

An dieser Stelle möchte ich kurz ein einfaches Verhaltensmuster der Meeresschnecke *Aplysia californica* beschreiben, dessen Abstraktion zu einem neuen Lernverfahren für on-line Daten führen wird, wie im folgenden Kapitel 4 beschrieben. Analog zu dem Vorgehen in Abschnitt 1.2 wird hier eine physiologische Motivation für die im Algorithmus verwendeten Prinzipien angeführt. Die Darstellung ist hauptsächlich dem Aufsatz von Bailey & Kandel (1985) entnommen.

Abbildung 37: Meeresschnecke *Aplysia californica*, Photo von Caroline Schooly, http://siolibrary.ucsd.edu/slugsite/pacific/aplysia.htm

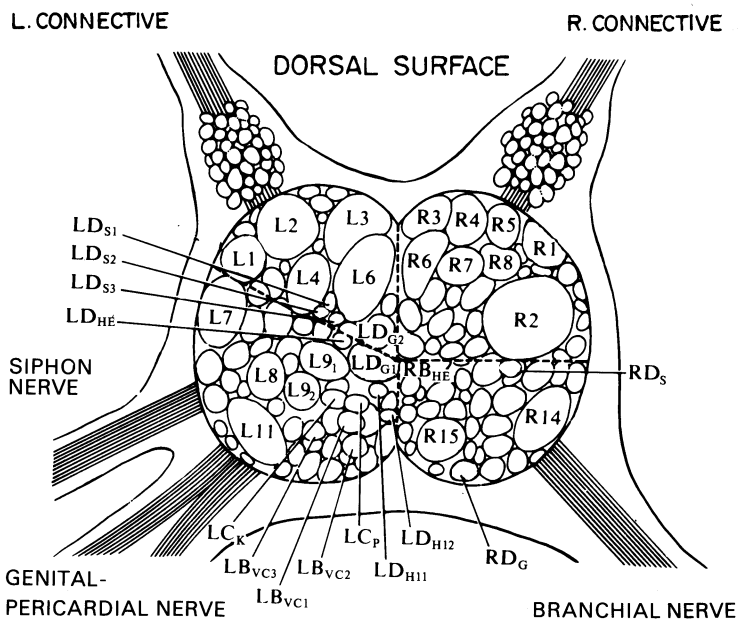


Abbildung 38: Plan der identifizierten Nervenzellen im Abdominalganglion der *Aplysia*. (Aus Bailey & Kandel, 1985.)

Die Meeresschnecke *Aplysia californica* (Abbildung 37) ist eines der Paradetiere der Erforschung biologischer neuronaler Netze. Sie besitzt etwa 20 000 Nervenzellen, aufgeteilt auf zehn größere Ganglien. Der Grund für die detaillierte Erforschung ihres Nervensystems ist die außergewöhliche Größe ihrer Nervenzellen. Manche haben einen Durchmesser von bis zu 1mm und gehören damit zu den größten tierischen Zellen überhaupt. Viele der Nervenzellen sind bei allen Individuen gleich und eignen sich deshalb hervorragend zum Studium ihrer Funktionsweise. Man bezeichnet sie mit R_2, L_7, etc. (s. Abbildung 38).

Wegen ihrer Größe lassen sich sowohl elektrophysiologische, biochemische als auch morphologische Studien an den Nervenzellen durchführen. Es wird dann möglich, die synaptische Konnektivität von Paaren identifizierter Zellen genau zu bestimmen, nämlich

(a) die Existenz einer funktionierenden Verbindung,

(b) ihr Vorzeichen (exzitatiorisch oder inhibitorisch) und

(c) ihre Stärke.

Dies ermöglicht, die neuronale Verschaltungsstruktur aufzufinden, die bestimmten Verhaltensmustern zugrundeliegt. Ein prominentes Beispiel dafür ist der *Kiemenrückzugsreflex*. Wie bei anderen Weichtieren ist *Aplysias* Kiemen in einer Hautöffnung verborgen und durch den *Mantle shelf* geschützt, der in einem fleischigen Rohr, dem *Siphon* endet. Wenn *Mantle shelf* oder *Siphon* gereizt werden, reagieren sie mit heftigem Zusammenziehen und verbergen sich und den Kiemen in der Öffnung. Wie Eric Kandel, der dafür mit dem Nobelpreis bedacht wurde, zeigen konnte, ist es eine typische Eigenschaft dieses defensiven Rückzugsreflexes, dass er durch *Erfahrung* modifiziert werden kann. Dabei treten sowohl Kurz- als auch Langzeitgedächtniseffekte auf.

Gewöhnung

Die einfachste Form der Reflexmodifikation ist die Gewöhnung an den Reiz. *Aplysia* reagiert bei der ersten Berührung des *Siphon* heftig. Bei minütlich wiederholter Berührung wird die Reaktion langsam schwächer, bis sie nach der zehnten praktisch verschwunden ist. *Aplysia* ignoriert anschließend alle weiteren Reize dieser Art. Die Gewöhnungsphase ist nach einigen Stunden wieder vorbei. Wird die Prozedur jedoch einige Male wiederholt, dann stellt sich der Reflex erst nach ein paar Wochen wieder ein.

Die Details der Verschaltung, die zur Gewöhnung führen, sind in Abbildung 39 dargestellt. 24 Sensorische Nervenzellen (SN) im Abdominalganglion aus Abb. 38 haben ihre rezeptiven Felder im *Siphon*. Sie geben die Signale sowohl an die sechs Motorneuronen (L_7 bis RD_G), als auch an die Interneuronen L_{16}, L_{22} und L_{23} weiter. Wenn ein Aktionspotential in den Endsynapsen der Sensorneuronen (Abbildung 40a) ankommt, dann wird dort eine gewisse Anzahl von Ca^{++}-Kanälen geöffnet, die Ca^{++}-Ionen fließen in die Synapse und erlauben dort den Transmittervesikeln, sich mit der Zellmembran zu verbinden und ihren Inhalt nach außen in den synaptischen Spalt auszuschütten. Bei wiederholter Reizung nimmt die Konzentration von offenen Ca^{++}-Kanälen in der Synapsenwand ab, und mit der Abnahme der Ca^{++}-Ionenkonzentration im synaptischen Endknöpfchen wird die Weiterleitung des Signals unterbunden (Abb. 40c). Das Gedächtnis hat hier also einen spezifischen Ort, nämlich die synaptischen Endknöpfchen von den Sensorzellen zu den Motorzellen.

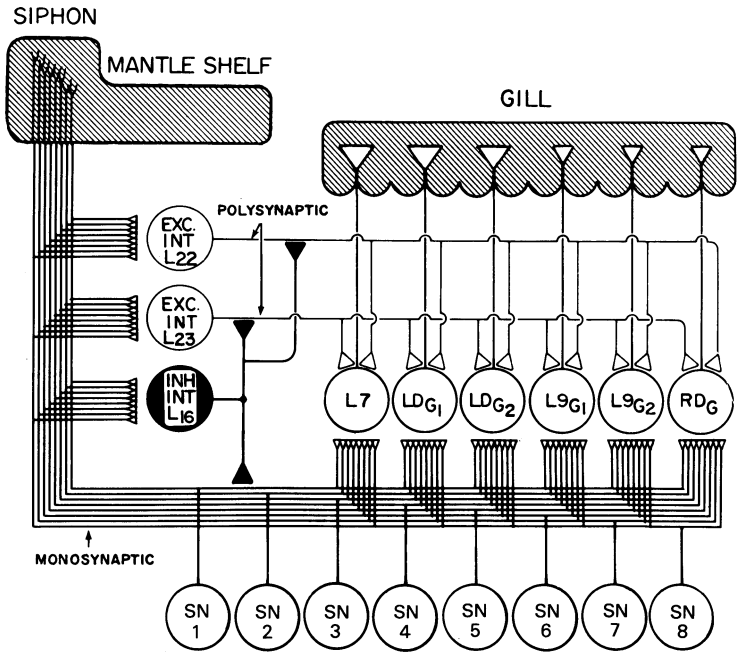


Abbildung 39: Verschaltung, die zur Unterdrückung des Kiemenreflexes führt. Der Reflexbogen reicht vom *Siphon* über die Sinneszellen (SN) zu den Motorneuronen (L_7 bis RD_G). Parallel steuern die Interneuron L_{16}, L_{22} und L_{23} die Übertragungseffizienz der synaptischen Verbindung. Ihre Synapsen solten eigentlich an den Synapsen der SN eingezeichnet sein, die an den Motorneuronen anliegen (Abb. 40). (Aus Bailey & Kandel, 1985.)

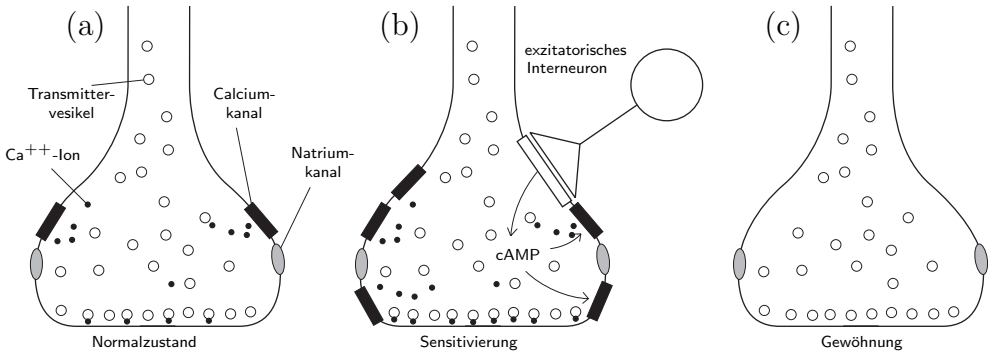


Abbildung 40: Molekulare Grundlage der Gewöhnung und Sensibilisierung. Die Sensitivierung entspricht einer erhöhten Ca^{++}-Konzentration, die durch molekulare Prozesse durch das Interneuron ausgelöst werden. Die Gewöhnung beruht dagegen auf Ca^{++}-Verarmung.

Sensitivierung

Der gegenteilige Effekt zur Gewöhnung ist ebenfalls beobachtbar. Durch einen gleichzeitigen Schlag auf den Kopf oder den Schwanz der *Aplysia* kann der Kiemenreflex auch in der Gewöhnungsphase wiederhergestellt werden. Bailey & Kandel (1985) schlagen dafür einen molekularen Mechanismus vor, der durch zyklisches Adenosinmonophosphat (cAMP) die Öffnung der Ca^{++}-Kanäle steuert. Abb. 40b zeigt die Verbindung eines steuernden Interneurons mit der Synapse eines Sensorneurons, das an einem Motorneuron anliegt (sog. Synapsen–Synapsen-Verbindung). Durch eine Kette von biochemischen Vorgängen wird die Ca^{++}-Konzentration im Endknöpfchen erhöht, was zu verstärkter Ausschüttung von Neurotransmittern führt. Die Interneuronen können also direkt Einfluss auf die Effektivität der Synapsen zu den Motorneuronen nehmen.

Abstraktionen

Selbstverständlich sind die Prozesse der Gewöhnung und Sensitivierung hier nur schematisch zusammengefasst. In Hinblick auf die Lernsituation eines *univar*-Netzwerkes lässt sich dieses Verhaltensmuster noch weiter abstrahieren.

Bei jedem Reiz, der von den Sensorzellen weitergeleitet wird, entscheidet das Interneuron aufgrund seiner zusätzlichen Information über die Umwelt (eventuelle Schläge auf den Kopf etc.), ob das Reizmuster am *Siphon* relevant ist oder nicht. Bei irrelevanten Reizen sorgt es dafür, dass die synaptischen Endknöpfchen langsam an Ca^{++} verarmen, während es bei einem relevanten Reiz die Ca^{++}-Konzentration schlagartig wiederherstellt. Übertragen auf das *univar*-Netzwerk ergeben sich zwei wichtige Änderungen am bisherigen Vorgehen, das auf der neuronalen Interpretation in Abschnitt 1.2 beruhte:

- Es werden nicht mehr einheitlich *normierte* Reize präsentiert, sondern unterschiedlich abgeschwächte. Dies hebt die Annahme (1-62) auf.
- Die aktuelle Normierung eines Reizes hängt nicht allein von seiner eigenen Beschaffenheit ab, sondern – wie bei *Aplysia* – auch vom momentanen Zustand des Netzes und von seiner Geschichte. Die Normierung geschieht *selbstreferentiell*.

Diese Vereinfachung kann sowohl auf die *Verarbeitung* von Reizmustern angewandt werden, bei der das Signal lediglich an Motorneuronen oder weitere Netzwerkschichten weitergegeben wird, als auch auf den gleichzeitig stattfindenden *Lernprozess*. Für den *univar*-Algorithmus, der gemäß der kompetitiven Hebb'schen Regel lernt, muss nun die entsprechende Regel für unnormierte Reizmuster gefunden werden. Dazu wird das unnormierte Reizmuster zum Zeitpunkt t in seine Norm $f(t)$ und das auf eins normierte Muster $\mathbf{h}(t)$ zerlegt. Der Gesamtreiz $f\mathbf{h}$ wird nun in Gleichung (1-67) eingesetzt, die a_r-Terme ändern sich wegen der kompetitiven Normierung der Aktivitäten in (1-63) nicht. Aus der kompetitiven Lernregel (1-70) für die Codebuchzentren $\mathbf{c}_r$ wird dann

$$\mathbf{c}_r(t) = \mathbf{c}_r(t-1) + \varepsilon\, f(t)\, a_r(t)\, \big(\mathbf{x}(t) - \mathbf{c}_r(t-1)\big), \tag{3-1}$$

wobei a_r und $\mathbf{x}$ dieselben Größen sind wie in der ursprünglichen Version mit normierten Reizen. Es ist also nur der Term $f(t)$ zur Lernrate hinzugefügt worden.

Es stellen sich zwei Fragen; erstens, wie das Gewicht $f(t)$ aus $\mathbf{x}(t)$ und $\boldsymbol{\theta}(t)$ bestimmt werden kann, und zweitens, welche Auswirkungen $f(t)$ auf das *univar*-Lernen hat. Hinsichtlich der ersten Frage kann aus *Aplysias* Gewöhnungsverhalten geschlossen werden, dass sie zwischen bekannten, sich ständig wiederholenden Reizen, und neuen, für sie interessanten unterscheidet. Übertragen auf den *univar*-Algorithmus wäre also zur Festlegung der *Relevanz* $f(t)$ des aktuellen Datenpunktes ein Maß für seine „Neuheit“ anzugeben.

Will man beispielsweise den Gewöhnungsprozess von *Aplysia* auf das *univar*-Lernen übertragen, so würde dieser durch stetig sinkende Werte von f ausgedrückt. Zur Sensibilisierung durch „neue“ Reize sollte f schlagartig wieder auf 1 gesetzt werden. Auf diese Weise kann f die Gedächtniskerne der Neuronen *modulieren* und diejenigen Daten schwächer gewichten, die für das MVNN – wie wiederholte Kiemenberührungen für die *Aplysia* – uninteressant sind. Diese Daten würden dann effektiv aus dem Strom der gelernten Daten ausgeschnitten. Im optimalen Fall sähe die Sequenz der gelernten Punkte wie in Abb. 7 aus. An Abb. 26c$_2$ wurde bereits festgestellt, dass Gedächtniskerne mit Lücken nach Gleichung (2-17) länger sind als solche ohne Lücken. Indem f als Datenselektor fungiert, verlangsamt es die effektive Lerndynamik so, dass sie von der Systemdynamik abgekoppelt werden kann.

Die zweite Frage zielt auf den Zweck der Einführung von $f(t)$ ab. In Kapitel 2 wurde der Zusammenhang zwischen $\varepsilon a_r(\mathbf{x}(t))$ und der effektiven Lerndynamik des MVNN diskutiert. Hier wird nun ein zusätzlicher Term eingefügt, der diese Dynamik modulieren kann. Durch kleineres f wird die Lerngeschwindigkeit vermindert, und es besteht die begründete Hoffnung, dass dadurch die Effekte der dynamischen Kopplung an den on-line Datenstrom abgeschwächt werden können. Wichtig ist dabei, dass f kein konstanter Faktor sein darf, sondern sowohl vom aktuellen Datenpunkt $\mathbf{x}(t)$ als auch von der Vorgeschichte des MVNN, also seinen Netzwerkparametern $\boldsymbol{\theta}$ abhängen muss. Dadurch wird die Lerndynamik nicht, wie im Übergang von Abb. 23a nach 23b, insgesamt verlangsamt, sondern nur an denjenigen Stellen im Datenstrom, an denen dies nötig ist, die also redundante Daten enthalten.

Wie eine solche selbstreferentielle Datenselektion im *univar*-Algorithmus implementiert werden kann, und welche Auswirkungen sie auf den Lernprozess hat, möchte ich im folgenden Kapitel beschreiben.

Kapitel 4

Neuigkeitsorientiertes Lernen

In Kapitel 2 wurde die Verschiebung des Phasenübergangs durch die dynamische Kopplung von Codebuch und System als das on-line Lernproblem des *univar*-Algorithmus entdeckt. Dort ergab sich als natürliches Maß für die effektive Beweglichkeit der Codebuchzentren die inversen Längen des Gedächtniskernes, also im wesentlichen der Lernparameter ε, moduliert durch die räumliche Aufteilung des Merkmalsraumes, die durch $a_r(\mathbf{x})$ vorgenommen wird. In Zusammenhang mit Abb. 26 konnte gezeigt werden, dass räumliche Kompetition die effektive Lerndynamik so stark verlangsamen kann, dass die Dynamikkopplung aufgehoben wird und ein Phasenübergang geschieht. Phasenübergang und Entkopplung bedingen einander dabei gegenseitig.

Im letzten Kapitel wurde daraufhin anhand des Gewöhnungsverhaltens von *Aplysia* eine weitere Möglichkeit zur Modulation der Lernrate entdeckt, und zwar der Faktor f in der Lernregel (3-1). Dieser ist nicht primär räumlicher, sondern auch zeitlicher Natur, da er von *Aplysia* als Filter für zeitlich redundante Daten verwendet wird. Die neurophysiologischen Untersuchungen an *Aplysia* wurden nicht so weit betrieben, dass sich bereits eine Regel erkennen ließe, wie f aus dem aktuellen Reiz $\mathbf{x}$ und dem Netzwerkzustand $\boldsymbol{\theta}$, der das effektive Gedächtnis an die vergangenen Reizmuster ist, zu berechnen sei. Da der *univar*-Algorithmus kein physiologienahes Modell ist, wie aus Abschnitt 1.2 ersichtlich wird, wäre dieser Ansatz hier auch nicht angemessen. Vielmehr soll nur das Prinzip der Datenfilterung von *Aplysia* übernommen werden. Dazu wurde bemerkt, dass f, welches ja die Netzwerkparameter $\boldsymbol{\theta}$ verändern soll, auf jeden Fall auch von $\boldsymbol{\theta}$ abhängen muss, was die Filterung von Datenpunkten *selbstreferentiell* werden lässt.

An dieser Stelle soll nun ein konkreter Algorithmus vorgeschlagen werden, der durch die neue Lernregel (3-1) und eine einfache selbstreferentielle Regel für $f(\mathbf{x}, \boldsymbol{\theta})$ in der Lage ist, die Lern- und die Systemdynamik so weit voneinander zu trennen, dass ihre Kopplung zwar nicht immer vollständig unterbunden, aber auf jeden Fall abgeschwächt wird. Wegen dieser Eigenschaft werde ich ihn ANTS (*Algorithm for Neural Timescale Separation*) nennen. Er ist eine Modifikation des *univar*-Algorithmus und als solcher auch eine Art Dichteschätzer. Welche Verteilungsdichte er approximiert, und welche Änderungen an seinem dynamischen Verhalten sich daraus ergeben, wird auf den folgenden Seiten dargestellt.

4.1 Entfernen von redundanten Daten

Der ANTS ist ein on-line lernender Algorithmus gemäß der Charakterisierung am Anfang von Kapitel 2. Es kann vorkommen, dass in dem präsentierten on-line Datenstrom zeitliche Korrelationen auf mehreren Zeitskalen vorkommen, ähnlich wie sie für den Markovprozess in (2-12) definiert sind. Die Forderung an einen on-line-fähigen Algorithmus ist, dass er keine a-priori Kenntnis dieser Skalen besitze, also insbesondere keine starren Regeln kennen kann, welche einen Datenstrom nach einer bestimmter Dauer als „redundant“ einstufen.

Vielmehr muss die Relevanz eines Datenpunktes $\mathbf{x}$ vom Lerner selbst festgestellt werden. Dafür stehen ihm ausschließlich der aktuelle Punkt $\mathbf{x}$ und seine Netzwerkparameter, also sein Modell für die Umgebung und ihre Vergangenheit, zur Verfügung.

4.1.1 Die Aufmerksamkeitsschwelle

Die Relevanz muss aus dem aktuellen Modell $A(\cdot\,;\boldsymbol{\theta})$ bestimmt werden. Dazu führt man üblicherweise eine *Aufmerksamkeitsschwelle* ein, mit der $A(\mathbf{x};\boldsymbol{\theta})$ verglichen wird (Wilden, 1998; Albrecht et al. 2000). Für diejenigen Datenpunkte $\mathbf{x}$, deren Aktivitäten $A(\mathbf{x};\boldsymbol{\theta})$ einen Schwellenwert $\epsilon_A > 0$ überschreiten, darf angenommen werden, dass dort bereits genügend Datenpunkte gesehen und gelernt wurden. Sie brauchen also nicht weiter beachtet zu werden, schließlich werden sie von hinreichend vielen Neuronen repräsentiert.

Das *Relevanzkriterium* ist hier also nichts anderes als ein *Neuigkeitskriterium*. Bereits bekannte Daten werden als irrelevant, unbekannte mit kleinem Aktivitätswert dagegen als relevant eingestuft.

Die einfachste Möglichkeit zur Berechnung von f aus dem Vergleich von $A(\mathbf{x};\boldsymbol{\theta})$ und ϵ_A ist diejenige, einen Datenpunkt entweder genau wie im *univar*-Algorithmus zu lernen, also mit $f=1$, oder ihn vollständig zu ignorieren ($f=0$). Formal wird dies durch die Heavyside-Funktion ausgedrückt,

$$f(\mathbf{x}) := \Theta\big(\epsilon_A - A(\mathbf{x};\boldsymbol{\theta})\big) = \begin{cases} 1 & \text{falls } A(\mathbf{x};\boldsymbol{\theta}) < \epsilon_A, \\ 0 & \text{sonst.} \end{cases} \tag{4-1}$$

Diese Regel ist die einfachste, die dem Prinzip der selbstreferentiellen Relevanzbestimmung genügt. Kompliziertere Funktionen als Θ ließen sich ohne weiteres verwenden, im folgenden soll jedoch von (4-1) ausgegangen werden. Es stellt sich nun die Frage, wie die Schwelle ϵ_A zu bestimmen ist. Sie soll keine a-priori-Kenntnis über die Daten enthalten, sich vielmehr nach den vorkommenden Werten von $A(\mathbf{x})$ richten.

Für jede feste Schwelle beobachtet man, dass ein gewisser Prozentsatz der Datenpunkte gelernt wird. Eine Schwelle $\epsilon_A > \max_{\mathbf{x}}\{A(\mathbf{x})\}$ lässt alle Punkte lernen, eine sehr kleine Schwelle kaum Punkte. Jedem Wert von ϵ_A lässt sich also der entsprechende Anteil α der gelernten Punkte am gesamten Datensatz zuordnen,

$$\mathcal{S}_A\colon\ [0,\infty) \to [0,1]\colon\ \epsilon_A \mapsto \alpha. \tag{4-2}$$

Um die Schwelle im Algorithmus bestimmen zu können, muss man diese Abbildung invertieren. Es sollte möglich sein, den Prozentsatz an Datenpunkten vorzugeben und die Schwelle daran anzupassen. Dies gelingt mit der *Update*-Regel

$$\epsilon_A(t+1) = \epsilon_A(t) + \begin{cases} \eta\,\alpha & \text{falls } f(\mathbf{x}_t) = 0, \\ \eta\,(\alpha - 1) & \text{sonst.} \end{cases} \tag{4-3}$$

Dabei ist $\alpha \in (0,1)$ der Anteil der gelernten Datenpunkte und $\eta \ll 1$ eine einfache Schrittweite. Immer dann, wenn ein Punkt eine zu kleine Aktivität hatte, wird die Schwelle verkleinert, und umgekehrt. Die Schwelle wird also bei jedem Datenpunkt verändert und konvergiert nicht, wenn η konstant ist. Dennoch liefert sie das gewünschte Ergebnis, denn der Erwartungswert von ϵ_A nimmt genau denjenigen Wert an, bei dem die Aktivität $A(\mathbf{x})$ mit einer Wahrscheinlichkeit von α darunter und mit der Wahrscheinlichkeit $(1-\alpha)$ darüber liegt. Dies ist leicht einzusehen, wenn die Regel (4-3) als Erwartungswert formuliert und ihr stationärer Zustand bestimmt wird,

$$\begin{aligned} & P(A(\mathbf{x}) > \epsilon_A) \cdot \eta\alpha + P(A(\mathbf{x}) < \epsilon_A) \cdot \eta(\alpha - 1) \\ & \quad = (1-\alpha)\eta\alpha + \eta\alpha(\alpha - 1) \\ & \quad = 0. \end{aligned} \tag{4-4}$$

Gleichung (4-3) scheint auf den ersten Blick wie alle anderen Lernregeln in dieser Arbeit eine gleitende Mittelwertsbildung nach der prototypischen Form (1-35) zu sein. Sie erlaubt aber nach genauerem Hinsehen keine solche Interpretation mehr. Die Größe, die „gemittelt“ würde, ist α bzw. $(\alpha-1)$, also ein *Prozentsatz* aus dem Intervall $[0,1]$. Die „gemittelte“ Größe ϵ_A ist dagegen eine *Aktivität* aus dem unbeschränkten Intervall $[0,\infty)$. Sie kann also kein gleitender Mittelwert der rechts stehenden Größe α resp. $(\alpha-1)$ sein.

Um die Interpretation als gleitenden Mittelwert zu ermöglichen, müsste demnach statt α der zugehörige Aktivitätswert ϵ_A der Schwelle sowie statt $(1-\alpha)$ die Differenz zwischen maximaler Aktivität und ϵ_A verwendet werden. Die Lernregel (4-3) als gleitender Mittelwert benötigte in diesem Fall die vollständige Invertierung der Abbildung $\mathcal{S}_A$ aus (4-2),

$$\epsilon_A(t+1) = \epsilon_A(t) + \begin{cases} \eta\,\mathcal{S}_A^{-1}(\alpha) & \text{falls } f(\mathbf{x}_t) = 0, \\ \eta\,\mathcal{S}_A^{-1}(\alpha - 1) & \text{sonst,} \end{cases} \tag{4-5}$$

oder zumindest das Maximum $\mathcal{S}_A^{-1}(1)$. Die Lernregel wurde aber genau zu dem Zweck eingeführt, $\mathcal{S}_A$ an der Stelle α zu invertieren. Die Regel (4-3) kann hier also nicht zu einer gleitenden Mittelwertsbildung umgeformt werden. Um dies zumindest näherungsweise trotzdem zu erreichen, muss man eine Approximation von $\mathcal{S}_A$ verwenden. Dazu könnten analytische Überlegungen angestellt werden, wie $\mathcal{S}_A$ durch Polynome zu approximieren sei (Wilden, 1998), hier soll jedoch ein ein algorithmisch einfacherer Weg eingeschlagen werden. Um bei den Zahlenwerten α und $(1-\alpha)$ keinen zu großen Fehler zu machen, muss berücksichtigt werden, dass die meisten Funktionen A bei sehr kleinen Werten sehr breit sind. Wenn man dort eine Änderung im Schwellenwert macht, ändert sich α nach (4-2) stärker als bei großen Werten von ϵ_A. Deshalb wird unten in der endgültigen Version des ANTS statt der Schwelle ihr Logarithmus angepasst. Dadurch wird für $\mathcal{S}_A$ bei sehr kleinen Schwellwerten ein exponentielles Verhalten angenommen.

4.1.2 Ein Algorithmus für die Entkopplung von Lern- und Systemdynamik

Der ANTS, als selbstreferentiell filternder *univar*-Algorithmus, lautet nun folgendermaßen:

(TS-1) Initialisiere die Netzwerkparameter wie in (MV-1)

(TS-2) Ziehe den *nächsten* Punkt $\mathbf{x}_t$ aus dem Datensatz $\mathcal{X}$. Berechne die Gesamtaktivität $A\big(\mathbf{x}_t, \boldsymbol{\theta}(t)\big)$ als Mischung von gleich gewichteten univariaten Normalverteilungen identischer Varianz σ, sowie die Zuständigkeiten $a_r\big(\mathbf{x}_t, \boldsymbol{\theta}(t)\big)$ durch globale Normierung nach Gleichung (1-10).

(TS-3) Bestimme die *Relevanz* des Punktes $\mathbf{x}$,

$$f\big(\mathbf{x}_t, \boldsymbol{\theta}(t)\big) = \Theta(\epsilon_A - A(\mathbf{x}_t)) \ \in \{0, 1\} \tag{4-6}$$

und passe die *Neuigkeitsschwelle* an,

$$\ln \epsilon_A(t+1) = \ln \epsilon_A(t) + \begin{cases} \eta\,\alpha & \text{falls } f(\mathbf{x}_t) = 0, \\ \eta\,(\alpha - 1) & \text{sonst.} \end{cases} \tag{4-7}$$

(TS-4) Berechne die neue Schätzung mit der Zentren-Lernregel

$$\mathbf{c}_r(t+1) = \mathbf{c}_r(t) + \varepsilon\, f\big(\mathbf{x}_t, \boldsymbol{\theta}(t)\big) a_r\big(\mathbf{x}_t, \boldsymbol{\theta}(t)\big)\,\big(\mathbf{x}_t - \mathbf{c}_r(t)\big), \tag{4-8}$$

(TS-5) Setze die Lernparameter ε und σ auf neue Werte, und zwar nach vorgegebenen *Annealing*-Schemata $\varepsilon(t)$ und $\sigma(t)$

(TS-6) Zähle die diskrete Zeit t weiter und gehe zu (TS-2).

Der Algorithmus ist so gemacht, dass die Lernparameter nur im Fall eines relevanten Datenpunktes abgekühlt werden. Implizit führt man also eine *subjektive Zeit* t_L des Lerners ein, die nur dann weitergezählt wird, wenn gelernt wird. In seiner eigenen Zeit verhält sich der Algorithmus also wie im *univar*-Modus. Wegen $f \in \{0, 1\}$ ist die subjektive Zeit hier

$$t_L(t) = \sum_{t'=0}^{t} f(t'). \tag{4-9}$$

In dieser Zeit erscheinen langlebige Datencluster nun von kürzerer Dauer, und man erhält, vom Standpunkt des Lerners aus gesehen, neue, verkürzte Systemzeitskalen. Diese Veränderung des zeitlichen Systemverhaltens ist Ziel und Grund für die Einführung des Faktors f in die Lernregel.

In der Formulierung des ANTS in (TS-1) bis (TS-6) wurde durchgängig die Notation der *Aktivitäten* A und a_r verwendet, und nicht, wie im *multivar*-Algorithmus, diejenigen der *geschätzten Wahrscheinlichkeiten* $\hat{p}$ und $\hat{P}(r|\cdot)$. Dies hat den Grund, dass die Gesamtaktivität A hier kein Modell für die Verteilungsdichte p mehr ist, sondern andere Approximationseigenschaften hat, wie im folgenden Abschnitt gezeigt wird.

4.2 Das Verhalten des ANTS

Die Einführung der modifizierten Lernregel (4-8) und der Aufmerksamkeitsschwelle in Schritt (TS-3) hat nicht nur zeitliche, sondern auch räumliche Auswirkungen. Zunächst sollen ihre Auswirkungen auf die stationären Eigenschaften des ANTS beschrieben werden, anschließend wird beschrieben, wie sie Lern- und Systemdynamik so entkoppelt, dass die Hierarchie in den Phasenübergängen erhalten bleibt.

4.2.1 Stationäre Approximationseigenschaften

In Abschnitt 1.1 wurde erklärt, warum der *univar*-Algorithmus eine Approximation der Verteilungsdichte p des präsentierten Datensatzes durchführt. Die Frage nach der Approximationseigenschaft des ANTS ist wesentlich schwieriger zu beantworten. Die Verteilungsdichte wird, da Datenpunkte nach der Regel (TS-3) ignoriert werden können, ständig selbstreferentiell modifiziert. Deswegen wird nicht eine einfache *Log-likelihood*-Funktion der ursprünglichen Form

$$\int_{\mathcal{M}} p(\mathbf{x}) \ \ln A(\mathbf{x}, \boldsymbol{\theta}) \, d\mathbf{x} \tag{4-10}$$

maximiert, sondern es wird p durch die Verteilungsdichte $\tilde{p}$ der *tatsächlich gelernten* Datenpunkte ersetzt. Diese hängt selbst von den Parametern $\boldsymbol{\theta}$ ab.

Abbildung 41 zeigt, dass die Codebuchzentren selbst im stationären Endzustand des Lerners ständig hin- und hergezogen werden. Sie weisen dadurch unterschiedliches Selektionsverhalten zu unterschiedlichen Zeitpunkten auf. Datenpunkte an der Stelle $\mathbf{x} \in \mathcal{M}$ können, je nach Zustand des Lerners, einmal gelernt und zu einem anderen Zeitpunkt ignoriert werden. Beim Versuch, diejenige Menge $\Omega \subset \mathcal{M}$ zu charakterisieren, in der Datenpunkte gelernt werden, stellt man fest, dass die kleinen Bewegungen im Codebuch diese ständig „verschmieren." Man bekommt also nicht eine scharfe Menge Ω, sondern für jeden Punkt $\mathbf{x} \in \mathcal{M}$ eine Wahrscheinlichkeit $\omega_A(\mathbf{x})$, dass dort gelernt wird. Diese Wahrscheinlichkeiten kann man auch als Zuordnungsfunktion ω_A der *fuz-*

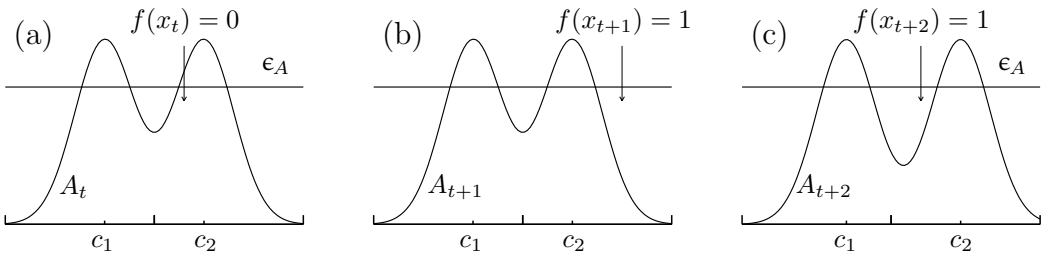


Abbildung 41: Skizze eines quasistationären Zustands im Lerner aus zwei Neuronen zu drei aufeinanderfolgenden Zeitpunkten. In (a) wird ein Datenpunkt x_t ignoriert, da die Aktivierung $A(x_t; \boldsymbol{\theta}(t))$ größer als die Schwelle ϵ_A ist. Der darauffolgende Punkt (b) wird gelernt, die Zentren rutschen leicht nach rechts. In (c) wird zufällig wieder der gleiche Punkt wie in (a) gezeigt, diesmal kann er jedoch gelernt werden.

zy-Menge Ω auffassen. Sie sind gleichzeitig die Ensemble-Mittelwerte $\langle f(\mathbf{x})\rangle_{\text{ens}}$ von $f(\mathbf{x})$, wobei das Ensemble aus allen Codebüchern des stationären Zustandes besteht.

Für einen Datensatz aus zwei verrauschten Zuständen, wie er im Laufe der Arbeit bereits des öfteren verwendet wurde, ist ω_A in Abbildung 42c dargestellt. Man sieht deutlich, dass es einen „äußeren" Bereich gibt, in dem immer $f=1$ gilt. Dort bleiben die Aktivitätswerte $A(\mathbf{x})$ immer unter ϵ_A. Dazwischen gibt es einen „inneren" Bereich, in dem die Datenselektion wirkt. Das stationäre Codebuch besteht hier aus zwölf Codebuchzentren. Die Funktion A, die in Abb. 42b eingezeichnet ist, ragt an acht Stellen leicht über die Schwelle ϵ_A hinaus (was in der Abbildung fast nicht zu sehen ist, da sie sich nur ganz leicht von dem Schwellwert unterscheidet). Dort werden verstärkt Datenpunkte ignoriert, was die Senken in ω_A verursacht.

Die Verteilungsdichte $\tilde{p}$ der tatsächlich gelernten Datenpunkte, die ebenfalls in Abb. 42b zu sehen ist, kann im quasi-stationären Zustand durch die Multiplikation von Zuordnungswahrscheinlichkeit ω_A und ursprünglicher Dichte p ausgedrückt werden. Ihre Normierung ist gerade der Anteil α der gelernten Daten, also

$$\tilde{p}(\mathbf{x},\boldsymbol{\theta}) = \frac{1}{\alpha}\, p(\mathbf{x})\; \omega_A(\mathbf{x},\boldsymbol{\theta}) \quad \forall \mathbf{x} \in \mathcal{M}. \tag{4-11}$$

In Abb. 42b sieht man bei $\tilde{p}$ scharfe Senken und Spitzen an den gleichen Stellen wie bei der Zuordnungsfunktion ω_A in 42c. Analog zu Gleichung (4-10), führt nun die Lernregel (4-8) eine *Maximum-likelihood*-Schätzung durch, nur diesmal mit der modifizierten Verteilungsdichte $\tilde{p}$, die selbst von A, also den Netzwerkparametern $\boldsymbol{\theta}$ abhängt. Insgesamt wird statt (4-10) die Funktion

$$\int_{\mathcal{M}} \tilde{p}(\mathbf{x},\boldsymbol{\theta})\; \ln A(\mathbf{x},\boldsymbol{\theta})\, d\mathbf{x} = \frac{1}{\alpha}\int_{\mathcal{M}} p(\mathbf{x})\; \omega_A(\mathbf{x},\boldsymbol{\theta})\; \ln A(\mathbf{x},\boldsymbol{\theta})\, d\mathbf{x} \tag{4-12}$$

unter Variation der Parameter $\mathbf{c}_r \in \theta_r$ maximiert.

Approximation des Stumpfes von p

Man sieht in Abb. 42b, dass die Funktion A ein sehr flaches Plateau bekommt, dessen Höhe etwa der Schwellenwert ϵ_A ist. Sie weicht leicht von ϵ_A ab und verursacht dadurch die scharfen Spitzen und Senken in $\tilde{p}$ und ω_A. Es kann also vermutet werden, dass die *ideale* Aktivierungsfunktion des stationären Codebuchs dort genau den Wert ϵ_A annehmen würde. Sie wäre in diesem Fall eine Approximation des α-Stumpfes von p. Dieser ist in Abb. 42a zu sehen, er wird als die renormierte Version des unteren Bereiches von p definiert, der das Integral α hat, oder formal

$$p^c := \begin{cases} \epsilon_A & \text{für } p(\mathbf{x}) > \alpha\epsilon_A, \\ p(\mathbf{x})/\alpha & \text{sonst.} \end{cases} \tag{4-13}$$

In Appendix C.2 wird gezeigt, dass dies tatsächlich die *ideale* Approximationseigenschaft des ANTS ist,

$$A = p^c\,. \tag{4-14}$$

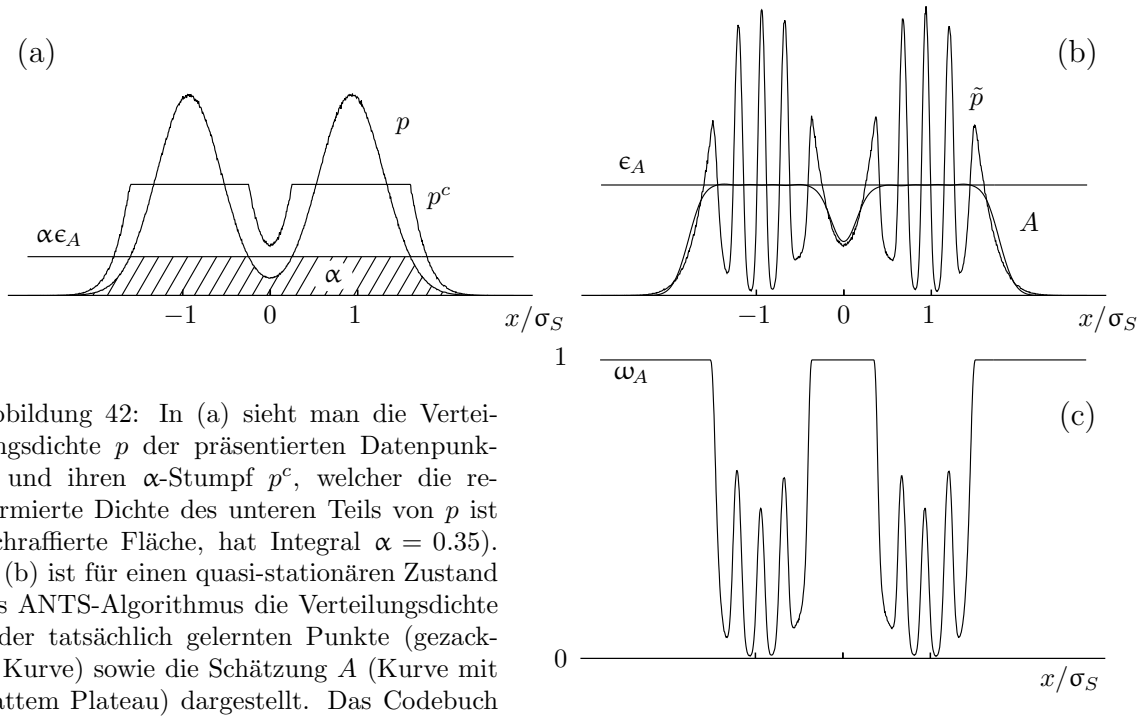


Abbildung 42: In (a) sieht man die Verteilungsdichte p der präsentierten Datenpunkte und ihren α-Stumpf p^c, welcher die renormierte Dichte des unteren Teils von p ist (schraffierte Fläche, hat Integral $\alpha = 0.35$). In (b) ist für einen quasi-stationären Zustand des ANTS-Algorithmus die Verteilungsdichte $\tilde{p}$ der tatsächlich gelernten Punkte (gezackte Kurve) sowie die Schätzung A (Kurve mit glattem Plateau) dargestellt. Das Codebuch besteht aus zwölf Neuronen ($M = 12$). Um eine gute Statistik für die Verteilungsdichten zu bekommen, wurde über lange Zeit mit konstanten Lernparametern trainiert ($\varepsilon = 0.01$, $\sigma = 0.16\sigma_S$). Die Zordnungsfunktion ω_A der Punkte von dem gezeigten zu dem gelernten Datensatz ist in (c) zu sehen. Man beachte, dass der Zusammenhang (4-11) hier schon visuell erkennbar ist.

Ideal ist sie deshalb, weil für diese Aussage vorausgesetzt werden muss, dass p^c genügend gut mit der endlichen Mischung A von Normalverteilungen approximiert werden kann, dass diese also aus beliebig vielen beliebig schmalen Normalverteilungen besteht. Gleichung (4-14) gilt demnach nur im Grenzübergang $M \to \infty$ und gleichzeitigem $\sigma \to 0$. Bemerkenswert ist dabei, dass die Verteilungsdichte $\tilde{p}$, je schmaler die Mischungskomponenten werden, umso schmalere und schärfere Spitzen bekommt. Das Plateau der Approximation A wird dagegen immer flacher und glatter. Dies ist in Ansätzen schon in Abb. 42b zu sehen, wird jedoch anhand von Abbildung 54 in Appendix C.2 noch deutlicher.

Wenn der Anteil α der gelernten Daten sehr klein gewählt wird, dann wird die Verteilungsdichte p sehr weit unten abgeschnitten. Effektiv wird dadurch der *Träger* von p geschätzt. Diese Eigenschaft des ANTS ließe sich in praktischen Anwendungen benutzen, in denen nicht die genaue Struktur des Datensatzes, sondern nur seine ungefähre Ausdehnung interessant ist.

Das zugehörige ML-Kriterium

Die Erkenntnis, dass durch den ANTS-Algorithmus die abgeschnittene Dichte p^c geschätzt wird, führt auf die Frage, bei welcher Breite σ_{opt} diese Schätzung optimal wird. Natürlich wäre sie durch einen *univar*-Lauf einfach zu bestimmen, wenn ein Datensatz

mit der Verteilungsdichte p^c zur Verfügung stünde. Da der Algorithmus jedoch keine Daten mit der Verteilungsdichte p^c sieht, sondern nur solche mit p oder $\tilde{p}$, muss man ein alternatives Kriterium für die optimale Breite σ_{opt} der Mischungskomponenten finden, welches dem ML-Kriterium mit einer zugrundegelegten Verteilungsdichte p^c äquivalent ist. Die Berechnungen und Abb. 54 am Ende von Appendix C.2 geben einen Hinweis darauf, dass im Idealfall – also wieder unter der Voraussetzung beliebig guter Approximationsfähigkeit durch die endliche Mischung A – kein Unterschied zwischen der Verteilungsdichte p^c und $\tilde{p}$ besteht, solange sie nicht als eigenständige „Funktionen" aufgefasst, sondern nur als Maß verwendet werden, bezüglich dessen in (4-12) integriert wird. Es liegt also nahe, zur Bestimmung von σ_{opt} das Funktional

$$\int_{\mathcal{M}} \tilde{p} \, \ln A \approx \langle f \, \ln A \rangle_{\mathcal{X}} \tag{4-15}$$

unter Variation von σ zu maximieren. Es ist das *Log-likelihood*-Maß für die Übereinstimmung von A und $\tilde{p}$; und da A immer glatter, je besser die Approximation wird, während $\tilde{p}$ immer schmalere Spitzen bekommt und der Verteilungsdichte p^c immer ähnlicher wird (Abb. 42 und 54), auch ein Maß für die Übereinstimmung von A und p^c. Von allen Größen, die dem Algorithmus zur Verfügung stehen, ist $\tilde{p}$ dem Stumpf p^c also am ähnlichsten.

Dies ist in numerischen Simulationen auf dem Datensatz aus Abb. 42 auch durch den Vergleich verschiedener anders lautender Funktionale deutlich geworden. Da ihre Ergebnisse wesentlich schlechter waren als diejenigen unter Verwendung von (4-15), werde ich sie hier nicht anführen. Ein Annealingprozess, bei dem das Funktional (4-15) beobachtet wurde, ist in Abbildung 43 zu sehen. Es wurde langsam abgekühlt und $\big(f(x) \ln A(x)\big)$ gleitend gemittelt. Der so bestimmte optimale Wert σ_{opt} liefert eine gute Approximation des α-Stumpfes von p.

Teilweise Rekonstruktion der Verteilungsdichte p

Dadurch, dass im ANTS nur der α-Stumpf approximiert wird, ist dennoch nicht alle Information über die ursprüngliche Verteilung verloren gegangen. Die mittlere Aktivierung eines Neurons, seine *Load* $\langle a_r \rangle$, wird im Algorithmus durch gleitende Mittelung berechnet. Wenn hierzu nun alle Datenpunkte, nicht nur diejenigen mit $f = 1$ verwendet werden, kann in einer zusätzlich mit der *Load* gewichteten Mischung aus Normalverteilungen,

$$A^l := \sum_r \frac{P_r \, \langle a_r \rangle}{(2\pi\sigma^2)^{d/2}} \, \exp\Big(-\frac{|\mathbf{x} - \mathbf{c}_r|^2}{2\sigma^2}\Big) \tag{4-16}$$

wieder die Struktur von p erkannt werden. Diese rekonstruierte Verteilungsdichte ist für das Beispiel oben in Abbildung 44 dargestellt. Natürlich erhält man bei dieser Methode fast nie eine so gute Approximation von p wie bei einem *univar*-Training, schließlich ist sie nicht das Optimierungsziel.

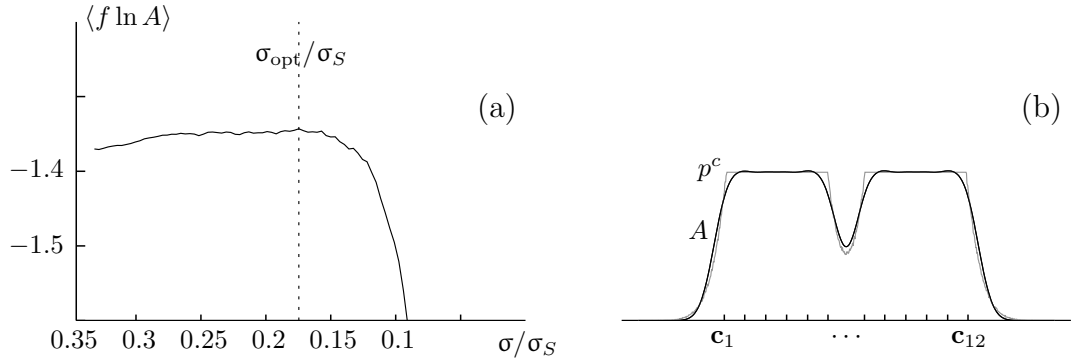


Abbildung 43: (a) Bestimmung des Maximums von Funktional (4-15) unter langsamem Annealing der Breite σ. Im Optimalfall wird die Verteilungsdichte p^c (graue Linie) relativ gut von der Mischung A univariater Normalverteilungen mit gleicher Varianz und gleichem statistischem Gewicht angenähert (b). Dieser Zustand des Codebuchs wurde auch in Abb. 42 verwendet.

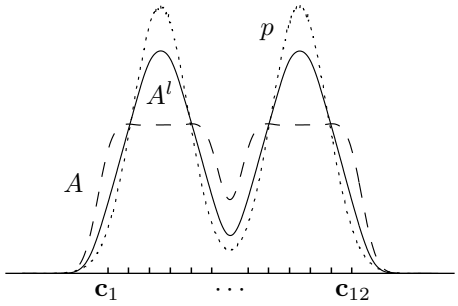


Abbildung 44: Die mit der *Load* modifizierte Schätzung A^l (kontinuierliche Linie) aus Formel (4-16) im Vergleich zu p (gepunktet). In einem einfachen Fall wie diesem lassen sich die Prototypen der Verteilung gut identifizieren. Die Aktivierungsfunktion A (gestrichelt) ist dieselbe wie in Abb. 43b.

4.2.2 Eigenschaften der Phasenübergänge

Die Phasenübergangskurve, wie sie in Kapitel 2 charakterisiert wurde, hängt immer auch vom gelernten Datensatz ab. Da dieser durch den ANTS im Vergleich zum ursprünglichen verändert wird, muss sich auch die kritische Parameterkurve verändern. Dabei lassen sich räumliche und zeitliche Aspekte zunächst noch klar trennen, sobald die Daten jedoch etwas kompliziertere Struktur aufweisen, vermischen sich Raum- und Zeitskalen wieder, wie die folgenden Beispiele zeigen werden.

Verschiebung in σ-Richtung

Im letzten Abschnitt wurde klar, dass der ANTS den Dichtestumpf p^c approximiert, der wesentlich breiter als die ursprüngliche Dichte ist. Seine Varianz ist also größer als die von p. Dadurch wird die Kurve der kritischen Parameter ebenfalls zu größeren σ-Werten geschoben, denn zumindest im Bereich $\tau_S \ll 1$ findet der Phasenübergang immer bei Gleichheit von σ^2 und der Varianz der zugrundegelegten Verteilung statt.

In Abbildung 45 ist die kritische Parameterkurve eines Beispiels gegeben, welches aus zwei verrauschten Systemzuständen besteht (wie in Abb. 22 und 42), das jedoch nicht Markov-artig schaltet, sondern deterministisch wie die Beispiele in Abb. 32. Deshalb findet man, wenn alle Datenpunkte gelernt werden ($\alpha = 1$), eine ähnliche Kurve kritischer Parameter wie in Abb. 32. Wieder stimmt sie hervorragend mit den theoretischen Kurven aus den Abbildungen 30 und 31 überein (hier ist $M = 20$). Wenn dagegen nur

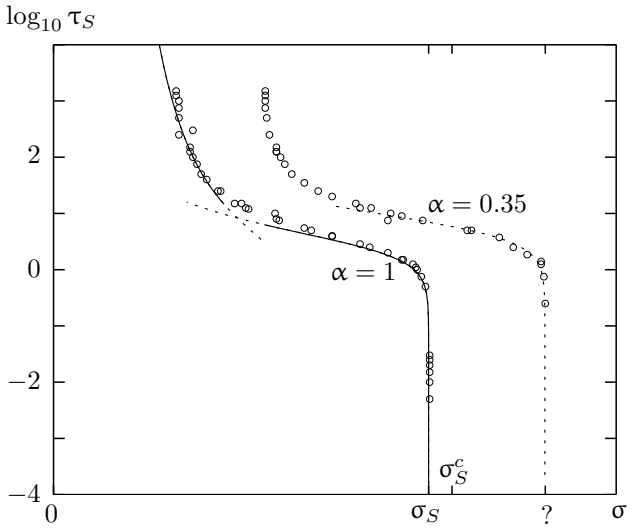


Abbildung 45: Phasendiagramm des ersten Aufbrechens in einem System, das zwischen zwei verrauschten Zuständen deterministisch schaltet. Der Lerner besteht aus $M = 20$ Neuronen und wurde nach dem ANTS sowohl mit allen ($\alpha = 1$), als auch mit 35 Prozent der Daten trainiert. Beim selektierenden Lernen ist die Kurve zu größeren σ verschoben. Sie ist noch weiter verschoben als es die Breite σ_S^c des Verteilungsstumpfes p^c erwarten ließe, die etwa auch die Breite des gelernten Datensatzes ist. Die gepunktete Linie nahe den ($\alpha = 0.35$)-Phasenübergängen ist aus der durchgezogenen Kurve links durch Streckung um den Faktor 1.31 in σ-Richtung und Verschieben um $\log_{10}(0.45)$ in τ_S-Richtung entstanden.

ein Datenanteil von $\alpha = 0.35$ gelernt wird, dann wird die Kurve zu größeren σ_{krit}-Werten verschoben, wie zu erwarten war.

Bemerkenswert und unerwartet an Abbildung 45 ist dagegen, dass die verschobene Kurve nicht bei $\sigma = \sigma_S^c$, der Standardabweichung des Stumpfes p^c abfällt, sondern bei noch größeren Werten σ_{krit}. Es führt also hier nicht nur die globale Varianz des insgesamt gelernten Datensatzes zum Phasenübergang.

Dies kann verstanden werden, wenn man sich die gelernten Punkte aus Sicht der einzelnen Normalkomponenten ansieht. Da die Spitzen der Mischung $A(\cdot\,;\boldsymbol{\theta})$ zur Datenselektion führen, also gerade dort weniger gelernt wird, wo sich auch die Zentren der Komponenten befinden, werden verstärkt Punkte an den Flanken der Normalverteilungen gelernt. Dies bedeutet, dass der Datensatz aus Sicht der einzelnen Komponenten eine noch größere *lokale* Varianz hat als es die globale vermuten ließe.

Verschiebung in τ_S-Richtung

Durch die Selektion der Datenpunkte findet die erwünschte Modulierung der Lernrate statt. Beim Weitertrainieren des Lerners im quasi-stationären Zustand bekommt man dadurch, dass bei jedem Punkt $\mathbf{x}$ die ursprüngliche Lernrate mit $f(\mathbf{x})$ multipliziert wird, effektiv einen um α skalierten Lernparameter. Denn dann ändern sich Codebuch und Aktivitätswerte a_r nur unwesentlich, und man kann den Mittelwert auseinanderziehen,

$$\varepsilon \langle f a_r \rangle_{\mathcal{X}} \approx \varepsilon \langle f \rangle_{\mathcal{X}} \langle a_r \rangle_{\mathcal{X}} = \varepsilon \alpha \langle a_r \rangle . \tag{4-17}$$

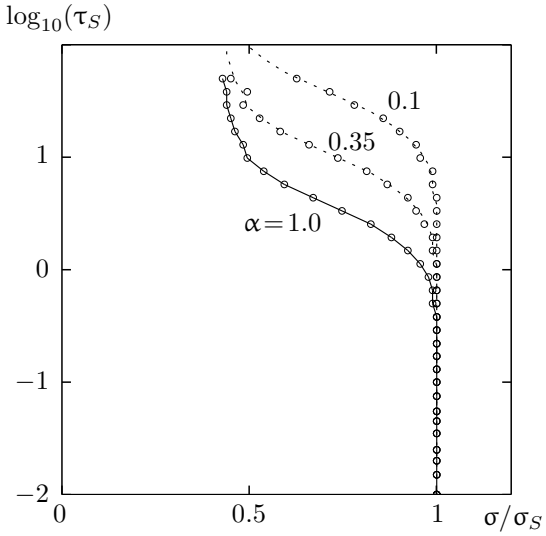


Abbildung 46: Kritische Parameterkurven für zwei punktförmige Datenquellen. Die Kurven des ANTS mit $\alpha=0.35$ und 0.1 sind jeweils um $|\log(\alpha)|$ verschoben. Wie genau diese Verschiebung ist, wird anhand der gepunkteten Kurven deutlich, die durch Verschiebung der durchgezogenen Kurve um exakt $|\log(\alpha)|$ entstanden sind.

Dies verschiebt die Kurve der kritischen Parameter in logarithmischer Darstellung um $|\log\alpha|$ nach oben. Besonders deutlich ist das an Datensätzen zu sehen, bei denen die Verschiebung der Kurve in σ-Richtung nicht stattfindet, beispielsweise an dem Datensatz aus punktförmigen Quellen, der bereits in Abb. 32 verwendet wurde. Das Ergebnis ist in Abbildung 46 für zwei verschiedene α-Werte gezeigt. Die dort abgebildeten Kurven haben genau den Abstand $|\log_{10}(\alpha)|$ von der ursprünglichen ($\alpha=1$)-Kurve.

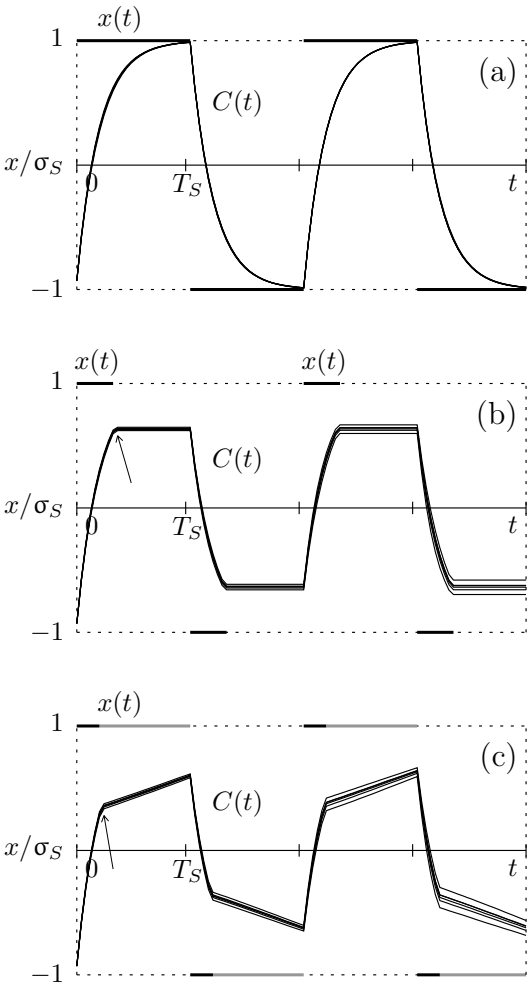
Auch in Abb. 45 findet diese Verschiebung in τ_S-Richtung statt. Dort ist sie wegen der starken σ-Verschiebung schwerer zu sehen, außerdem ist sie nicht so stark ausgeprägt wie sie nach Abb. 46 sein sollte. Der Abstand zwischen den Phasenübergangskurven mit $\alpha=1$ und $\alpha=0.35$ ist nur etwa $|\log_{10}(0.45)|$, und nicht der erwartete $|\log_{10}(0.35)|$. Hier ist die Näherung (4-17) nicht mehr gültig, denn da die Verteilungsdichte effektiv verändert wird, gibt es nichttriviale Korrelationen zwischen der räumlichen Zuständigkeit $a_r(x)$ und dem Datenselektor $f(x)$.

Anhand von Abb. 46 wird deutlich, weshalb der ANTS die Kopplung von Lern- und Systemdynamik aufheben oder zumindest abschwächen kann. Durch

$$\varepsilon \langle f a_r \rangle_{\mathcal{X}} < \varepsilon \langle a_r \rangle \tag{4-18}$$

wird die Kurve der kritischen Lernparameter für jedes $\alpha<1$ nach oben, zu größeren τ_S-Werten gestreckt. Der Parameterbereich von ε, in dem der Lernvorgang dynamisch *entkoppelt* stattfinden kann, wird auf diese Weise vergrößert. In Abb. 46 kann statt bis $\tau_S=1$ bei $\alpha=1$ nun bis $\tau_S=10$ bei $\alpha=0.1$ entkoppelt gelernt werden.

Durch den ANTS ist ist es nun möglich, mit wesentlich flexiblerem Codebuch in weiteren Bereichen der (zunächst ja noch unbekannten) Systemdynamik ungekoppelt zu lernen. Aufmerksame Lerner sind deshalb von Vorteil, weil sie genügend schnell mit einer Aufspaltung auf vorhandene Raumskalen reagieren, bei ihnen tritt die unerwünschte Retardierung durch zu kleines ε nicht auf.



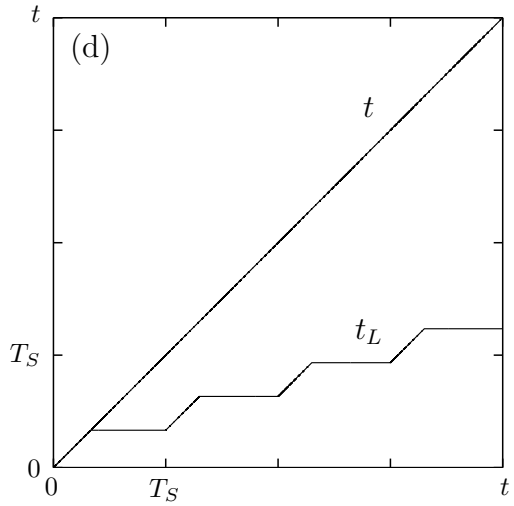


Abbildung 47: Kompression von Lebensdauern im System, sichtbar anhand dreier verschiedener Lernszenarien: in (a) ohne Selektion ($\alpha = 1$), in (b) mit Selektion ($\alpha = 0.3$), ohne die Schwelle zu verändern, und in (c) mit langsamer Anpassung von ϵ_A. Die gelernten Daten sind jeweils schwarz bei (± 1) eingezeichnet. In (d) ist die Entwicklung der subjektiven Zeit t_L aus (b) dargestellt. Man sieht deutlich die Zeitspannen, während derer sie „stehenbleibt," weil keine relevanten Daten mehr gesehen werden. Der Datensatz besteht aus zwei Punktquellen bei (± 1). Die Lernparameter sind mit $T_S/T_L = 5.0$, und $\sigma = 0.8\sigma_S$ in (a) in der entarteten, in (b) und (c) dagegen bereits in der aufgespalteten Phase (vgl. Abb. 46).

4.2.3 Dynamikentkopplung durch Zeitskalenkompression

Die Verteilungsdichte des in Abb. 45 gelernten Datensatzes ist diejenige, die bereits in Abb. 42a, b präsentiert wurde. Beide Datencluster werden von der Schwelle $\alpha\epsilon_A$ abgeschnitten, und da sie gleiche Form und gleiches statistisches Gewicht haben, werden sie dadurch auch *gleich stark komprimiert*. Dasselbe findet in Abb. 46 statt, dort werden zwei punktförmige Datenquellen mit gleichen statistischen Gewichten komprimiert.

Dass dieser Prozess nicht nur eine Kompression von statistischen Gewichten, sondern auch von *Zeitskalen* bedeutet, so wie sie in Abb. 7 als Ziel des Algorithmus vorgestellt wurde, kann am Beispiel aus Abbildung 47 demonstriert werden. Wieder besteht die Datenquelle aus zwei deterministisch abwechselnden Punktverteilungen. In Abb. 47a ist ein stark dynamisch gekoppelter Lernprozess, ähnlich demjenigen aus Abb. 26a$_1$ zu sehen. Es findet keine Aufspaltung statt, da sich der Lerner mit einem Zeitskalen-Verhältnis $\tau_S = 5.0$ bei dem gegebenen Verhältnis der Breiten $\sigma/\sigma_S = 0.8$ über der Phasenübergangskurve befindet (vgl. Abb. 46).

Zum Vergleich ist in Abb. 47b ein ANTS-Training mit fester Schwelle ϵ_A dargestellt,

die gerade so eingestellt ist, dass der Anteil $\alpha = 0.3$ gelernt wird. Man sieht hier, dass der Lernprozess zu denjenigen Zeitpunkten gestoppt wird, an denen die Aktivität $A(x) = A(\pm 1)$ gerade den Schwellwert ϵ_A erreicht (Pfeil). Dadurch ergibt sich eine ähnliche *Verkürzung der Lebensdauern* im System wie sie in Abb. 7 angedacht worden war. Die gelernten Datenpunkte sind schwarz bei (± 1) eingezeichnet.

In Teil (c) von Abb. 47 wird die Schwelle nach der *Update*-Regel (4-7) so variiert, dass sich insgesamt wieder ein Anteil $\alpha = 0.3$ einstellt. Während des schnellen Lernprozesses am Anfang jeder Lebensdauer sinkt die Schwelle ϵ_A deshalb ständig, bis der Lerner sie zum erstenmal überschreitet und „anstößt“ (Pfeil). Danach wird sie immer wieder leicht nach oben korrigiert, der Lerner kann demnach immer wieder ein paar Datenpunkte lernen, jedoch nicht alle. Die Schwelle wird hier abwechselnd unter- und überschritten (in Abb. 47c grau eingezeichnete Datenpunkte).

Sowohl in Abb. 47b als auch in 47c ist das Parameterpaar $(\sigma/\sigma_S, \tau_S) = (0.8, 5.0)$ auf der aufgespaltenen Seite des Phasenüberganges. Dies wird nicht nur aus Abb. 46 deutlich, sondern ist auch direkt an den Codebuchentwicklungen abzulesen. Die sechs Codebuchzentren entfernen sich in jeder Datenperiode weiter von einander, im Gegensatz zu Abb. 47a, wo dieselben Lernparameter (σ, ε) verwendet wurden, aber keine Aufspaltung stattfindet.

4.2.4 Dynamikentkopplung durch unterschiedliche Zeitskalenkompression

Mit den oben beschriebenen Situationen, in denen zwei Datencluster mit gleichen statistischen Gewichten verwendet wurden, konnten zwei Eigenschaften des ANTS verdeutlicht werden. In Abb. 46 ist die Entkopplung von Lern- und Systemdynamik zu sehen, in Abb. 47 die Tatsache, dass Phasenübergänge früher auftreten und deshalb wieder hierarchisch geordnet sein können.

Sie zeigen noch nicht deutlich genug die Fähigkeit des ANTS, auf unterschiedliche Zeit-

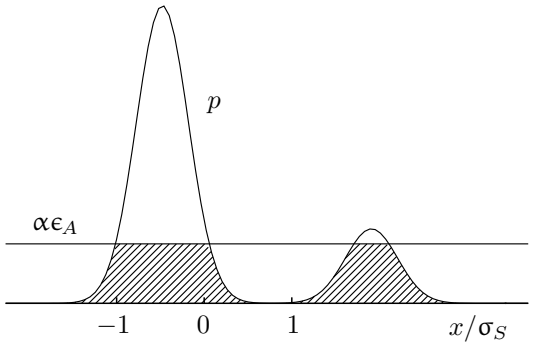


Abbildung 48: Unterschiedliche Zeitskalenkompression durch Abschneiden der Verteilungsdichte. Der Datengenerator schaltet zwischen zwei Zuständen hin- und her. Deswegen hat der Zustand mit größerem statistischen Gewicht auch längere Lebensdauern, und umgekehrt. Durch das Abschneiden der Verteilungsdichte werden die statistischen Gewichte und also auch die Lebensdauern einander angeglichen. Die längeren Dauern werden somit stärker verkürzt, was nach Abb. 34a zu schwächerer Dynamikkopplung führen sollte.

skalen im System auch mit *unterschiedlicher Kompression* zu reagieren. Dies wird anhand einer Verteilungsdichte mit stark unterschiedlichen Clustergewichten wie in Abb. 48 deutlich. Die präsentierte Verteilungsdichte dort weist einen großen und einen kleinen Cluster auf, die beide von der Schwelle ϵ_A abgeschnitten werden. Dementsprechend wird der große stärker komprimiert als der kleine.

Auf diese Weise wird das Problem *symmetrisiert*, und die Gewichte der tatsächlich gelernten Cluster werden einander angeglichen. Das zu lernende System aus zwei Zuständen mit unterschiedlichen Gewichten entpricht etwa demjenigen, deren theoretisch erwartete Phasenübergangskurven in Abb. 34a gezeigt sind. Dort sind Systeme mit ähnlichen Gewichten schwächer dynamisch gekoppelt als solche mit sehr unterschiedlichen Clustergewichten. Im Bereich nicht zu starker dynamischer Kopplung, also bei τ_S-Werten, die zwischen $\sigma/\sigma_S = 0.5$ und 1 eingezeichnet sind, bedeutet eine Angleichung der beiden statistischen Gewichte also eine *Entkopplung* der Dynamiken. Auch die beiden unterschiedlichen Lebensdauern aus Abb. 48 werden einander durch die im ANTS vorgenommene Datenselektion angeglichen und somit aus dem dynamisch gekoppelten Bereich entfernt.

Diese Fähigkeit, *unterschiedlich* auf unterschiedliche Lebensdauern im System zu reagieren, entsteht im ANTSdadurch, dass eine feste oder nur langsam veränderliche Schwelle verwendet wird. Die Aufmerksamkeitsschwelle unterscheidet nicht zwischen verschiedenen Datenclustern. Es zeigt sich hier, dass die *Zeitskala der Aufmerksamkeit* eine wichtige Rolle bei der Kompression unterschiedlicher Lebensdauern spielt. Ist ϵ_A zu beweglich, dann passt sie sich, wie im Beispiel aus Abb. 47c, während der Lebensdauer jedes Clusters an, und zwar so, dass sie leicht nach oben korrigiert wird, wodurch ständig weitere Daten gelernt werden. Das Bild in 48 wird durch diese Schwellenanpassung relativiert. Die effektiv wirksame Schwelle liegt bei dem großen Cluster mit den langen Lebensdauern höher als bei dem kleinen. Die unterschiedliche Zeitskalenkompression wird auf diese Weise etwas abgeschwächt.

Leider geht mit der Interpretation der *Update*-Regel (4-7) als gleitende Mittelwertsbildung auch die Bedeutung von $1/\eta$ als deren Zeitskala verloren. Es ist also sehr schwierig, die genaue Relaxationsgeschwindigkeit der Aufmerksamkeitsschwelle ϵ_A zu bestimmen oder einzustellen. Diese ist jedoch nur dann wichtig, wenn man an einer *maximalen* Kompression unterschiedlicher Lebensdauern interessiert ist. Für alle hier besprochenen Beispiele reicht die grobe Approximation durch $1/\eta$ aus.

Wie bereits in Abb. 34b deutlich wurde, eignen sich Systeme aus nur zwei Zuständen nur bedingt, um Effekte zu demonstrieren, die auf der Angleichung von statistischen Gewichten beruhen. Schon im einfachen Fall von $\alpha = 1$ können die theoretischen Ergebnisse nur sehr schwer in numerischen Simulationen reproduziert werden. Aus diesem Grund kann die unterschiedliche Zeitskalenkompression in Abb. 48 hier nicht an einem solchen System vorgeführt werden.

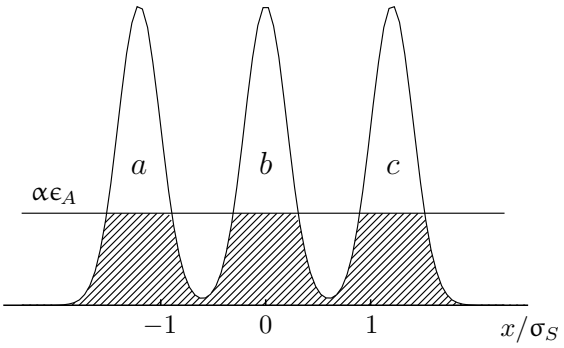


Abbildung 49: Die stationäre Verteilungsdichte eines Datensatzes aus drei Clustern. Dem zeitlichen Verhalten des Datensatzes liegt ein Markov-Prozess mit drei möglichen Zuständen $\gamma \in \{a, b, c\}$ zugrunde. Die Übergangswahrscheinlichkeiten $P(\gamma|\gamma)$ sind so eingestellt, dass a und b jeweils eine Lebensdauer von 10 haben, während diejenige von c mit $T_{S,c} = 1000$ wesentlich länger ist. Eingezeichnet ist auch die Schwelle $\alpha\epsilon_A$, mit $\alpha = 0.5$, die in Abb. 50 für die Datenselektion verwendet wird.

Ein System mit mehr als einer Zeitskala

Aus diesem Grund möchte ich einen indirekten Beweis der unterschiedlichen Zeitskalenkompression präsentieren. Dazu ist notwendig, den nächstkomplizierteren Datengenerator zu verwenden. Er besteht aus drei Datenclustern, die alle dieselbe Form und dasselbe statistische Gewicht haben. Seine stationäre Verteilungsdichte ist in Abb. 49 dargestellt, man sieht, dass sie bezüglich ihres Schwerpunktes 0 völlig symmetrisch ist. Der Datengenerator ist ein Markovprozess, der zwischen den drei Zuständen a, b und c schaltet. Die Lebensdauern der drei Zustände sind, obwohl sie gleiches statistisches Gewicht haben, sehr unterschiedlich. Die beiden Zustände links leben nur kurz, $T_{S,a} = T_{S,b} = 10$, während c sehr langlebig ist, $T_{S,c} = 1000$. Die Datenfolge springt also oft zwischen a und b hin- und her, wo sie jeweils nur kurz verharrt. Dieses Verhalten bleibt während erwarteten 2000 Zeitschritten gleich, danach schaltet es in den Zustand c, wo es erwartete 1000 Zeitschritte lang bleibt. Insgesamt kommt also jeder der Cluster, wie auch in Abb. 49 deutlich, gleich oft vor.

Abbildung 50 vergleicht die verschiedenen bisher besprochenen Lernverfahren auf dieser Datenfolge. In Abb. 50a wurde die optimale Codebuchentwicklung aus einem dynamisch vollständig entkoppelten Training bestimmt. Auffällig ist hier, dass das Codebuch zunächst in zwei Stücke mit jeweils drei Neuronen zerbricht, und erst bei etwa $\sigma/\sigma_S \approx 0.58$ die eigentlich dreiteilige Struktur der Datenverteilung entdeckt. Trotz allem bleibt die Aufspaltung vollkommen symmetrisch, sieht man von kleinen Artefakten bei den nachgeordneten Phasenübergängen ab.

In (b) wurde der Datensatz mit beweglicherem Codebuch gelernt ($\varepsilon = 0.02$), weshalb der Phasenübergang aus dem dynamisch gekoppelten Bereich heraus stattfindet. Da in den Daten mindestens zwei sehr verschiedene Zeitskalen enthalten sind, kommt es zu unterschiedlich starker Kopplung an die zugehörige Systemdynamik. Dadurch bildet der Lerner zunächst ein lokales Modell für den kurzlebigen Datencluster a (2 Neuronen) und ein weiteres für eine Kombination aus b und c (4 Neuronen). Aufgrund ihrer zeitlichen Struktur können diese beiden, die sehr unterschiedliche Lebensdauern haben, bei $\sigma/\sigma_S \approx 0.65$ noch nicht unterschieden werden. Die Lerndynamik bleibt, da c eine sehr lange Lebensdauer hat, an das Schaltverhalten des Systems gekoppelt. Erst bei kleineren Varianzen, etwa bei $\sigma/\sigma_S \approx 0.5$ findet auch hier der Phasenübergang statt. Da alle räumliche Information in den Daten vollständig symmetrisch ist, muss die Unsymme-

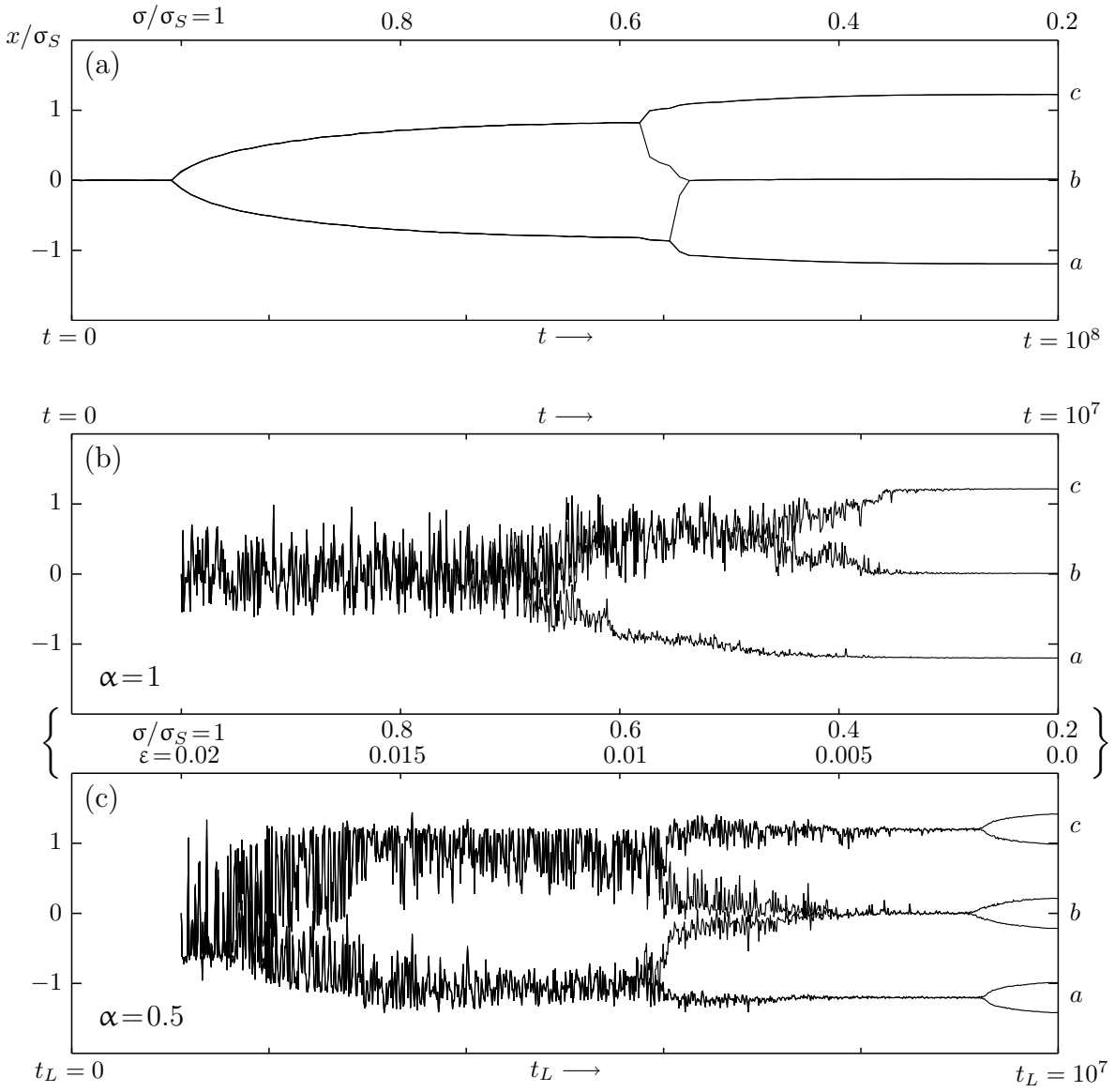


Abbildung 50: Die Codebuchentwicklungen ($M=6$) während drei verschiedener Lernprozesse des Systems aus Abb. 49. In (a) wurde eine randomisierte Datenfolge nach der *univar*-Methode gelernt. Die Phasenübergänge sind hier das Ergebnis der *Raumskalendetektion*, wobei nicht sofort die „eigentliche" dreiteilige Struktur des Datensatzes entdeckt, sondern zunächst ein Zustand mit zwei Clusterzentren angenommen wurde. In (b) wurde bei dem gleichen σ-Annealing auch ε linear abgekühlt (die σ/σ_S- und ε-Skalen gelten für (b) und für (c) gleichermaßen). Dadurch kommt es sukzessive zu Phasenübergängen, die aus dem dynamisch gekoppelten Bereich heraus geschehen. Das Aufbrechen des Codebuch findet aufgrund der unterschiedlich starken Dynamikkopplung an die unterschiedlichen Lebensdauern $T_{S,\gamma}$ bei unterschiedlichen ε-Werten statt, was die Codebuchentwicklung *unsymmetrisch* werden lässt, obwohl die zugrundeliegende statische Verteilungsdichte aus Abb. 49 symmetrisch ist. Bei dem ersten Phasenübergang zerbricht das Codebuch in zwei Teile mit 2 und 4 Neuronen, wie es die *Load-balance* verlangt. Der obere Ast bildet ein lokales Modell für die Zustände b und c, also einen schnellen und einen langlebigen ($T_{S,b}=10$, $T_{S,c}=1000$). In (c) ist ein mit $\alpha=0.5$ gelerntes Ergebnis zu sehen. Hier wird die Unsymmetrie wieder aufgehoben, wie in (a) finden hintereinander mehrere symmetrische Phasenübergänge statt. Da hier die abgeschnittene Verteilungsdichte gelernt wird, entdeckt der Lerner bei keinen σ/σ_S weitere Strukturen in den einzelnen Stümpfen (Abb. 49). In (b) war die Unsymmetrie im Codebuch auf die stark unterschiedlichen Lebensdauern im System zurückzuführen. Hier wurde diese Unsymmetrie vom ANTS behoben.

trie im Aufbrechverhalten auf die zeitliche Struktur des Datensatzes zurückzuführen sein. Insbesondere die dynamische Kopplung, die im kombinierten Modell für b und c stattfindet, zeigt, dass die stark unterschiedlichen Lebensdauern diese Unsymmetrie verursachen.

In (c) wurde dasselbe Annealing-Schema für ein ANTS-Training mit 50 Prozent aller Daten wiederholt (vgl. den α-Stumpf in Abb. 49). Bereits diese relativ schwache Datenselektion reicht aus, um die beiden sehr unterschiedlichen Lebensdauern der Daten anzugleichen. Man beobachtet qualitativ wieder das gleiche symmetrische Aufspaltungsmuster wie in (a).

Zwar ist, wie am Ende von Kapitel 2 angesprochen, das in Abb. 50b beobachtete Aufbrechverhalten hier nicht vollständig quantifizierbar, doch kann die Aufhebung der unsymmetrischen Aufspaltung – die allein auf die Existenz von zwei sehr unterschiedlichen Zeitskalen im System zurückzuführen ist – als Beweis für die unterschiedliche, an die Daten angepasste Lebensdauerkompression gewertet werden.

4.2.5 Zusammenfassung und Diskussion

Mit Einführung der selbstreferentiellen Datenselektion durch $f(\mathbf{x}_t; \boldsymbol{\theta}(t))$ kann, wie das letzte Beispiel gezeigt hat, selbst ein Datenstrom, in dem sehr unterschiedliche Systemzeitskalen enthalten sind, von einem aufmerksamen MVNN gelernt werden, ohne dass die hierarchischen Aspekte der Dichteschätzung vollständig verlorengehen. Der ANTS passt dabei die statistischen Gewichte im System durch den Mechanismus seiner Aufmerksamkeitsschwelle so an, dass sehr häufig vorkommende Cluster stärker abgeschwächt werden als seltene. Dadurch erscheinen ihm auch ihre Lebensdauern verkürzt, und der on-line Datenstrom wird *quasi randomisiert*.

Diese Quasi-Randomisierung führt dazu, dass in einem wesentlich größeren Parameterbereich *dynamisch entkoppelt* gelernt werden kann. Mit dem ANTS ist also ein Algorithmus gegeben, der eine wesentliche Fähigkeit zum on-line Lernen besitzt, nämlich selbst auf die Zeitskalen im System zu reagieren.

Versucht man, wie in Abb. 36 den Parameterweg des ANTS darzustellen, bekommt man effektiv mehrere Wege. Da die Näherung (4-17) für verschiedene Cluster verschieden gültig ist, bekommen die Neuronen, die für verschiedene Cluster zuständig sind, unterschiedliche *effektive Lernraten* $\varepsilon \langle f\, a_r \rangle_{\mathcal{X}}$. Diese Unterschiede in den Lernraten sind der unterschiedlich starken Gewichtskompression äquivalent. Es sind nun Phasenübergänge möglich, die auf sehr unterschiedliche Zeitskalen im System zurückzuführen sind.

Der ANTS benötigt dazu nur eine einzige a-priori Annahme über das zeitliche Systemverhalten, nämlich den Anteil α der zu lernenden Daten. Ein großes α schwächt die Möglichkeiten des ANTS zur unterschiedlichen Zeitskalenkompression ab, während bei zu kleinem α eventuell die relevante Struktur in der Datenverteilung nicht mehr sichtbar wird.

Bei allen Untersuchungen zum räumlichen und zeitlichen Verhalten des ANTS wurde festgestellt, dass das Ziel, die Modifikation des *zeitlichen* Systemverhaltens, nicht erreicht werden konnte, ohne auch die *räumlichen* Eigenschaften zu verändern. Statt der Verteilungsdichte p wird hier ihr Stumpf p^c gelernt. Hiermit ist erneut ein Hinweis darauf gegeben, dass jedes Verfahren zur Verarbeitung von on-line Datenströmen notwendigerweise eine *kombinierte* räumliche und zeitliche Beschreibung fordert. Einen ersten Hinweis hatten wir bereits in Kapitel 2 gesehen, wo sich der Zusammenhang zwischen Raum- und Zeitskalen in Form der Phasenübergangskurve zeigte.

Kapitel 5

Zusammenfassung und Ergebnisse der Arbeit

Das Ziel dieser Arbeit war, ausgehend von den bereits bestehenden Methoden zur Bildung effektiver Modelle von randomisierten Datensätzen, ein Verfahren zu finden, das auch die Verarbeitung von on-line Datenströmen ermöglicht.

In der *Einleitung* wurde zunächst kurz beschrieben, dass viele Lernmodelle der Neuroinformatik Mittelwertsbildungen durchführen. Daraufhin konnte bereits anhand des einfachsten Beispiels vorgeführt werden, dass solche Mittelwertsbildungen immer Probleme mit der Verarbeitung von on-line Daten haben müssen. Die Schwierigkeit des on-line Lernens äußerte sich in diesem Zusammenhang als das ständige Nachfolgen des gleitenden Mittelwertes hinter den Zuständen des Datengenerators. Diese Eigenschaft des Lernverfahrens wurde als die *Kopplung der Lerndynamik an die Systemdynamik* bezeichnet.

In *Kapitel 1* wurde das *multivar*-Netzwerk mit den zugehörigen Lernregeln eingeführt, die den Ausgangspunkt der nachfolgenden Untersuchungen darstellen. Im ersten Abschnitt des Kapitels wurde deutlich, dass es sich dabei um einen Algorithmus zur Approximation der Verteilungsdichte des Datensatzes handelt. Bei Vorgabe der räumlichen Skala σ, auf der diese Schätzung geschieht, bekommt er gleichzeitig die Eigenschaft eines Clusteringalgorithmus. Diese ist von besonderem Interesse für den Rest der Arbeit, da sie besonders anfällig gegen die Effekte der dynamischen Kopplung ist. Die Eigenschaften des *univar*-Algorithmus, der dieses Clustering durchführt, sind wegen der verwendeten Funktionenklassen, den Normalverteilungen, in einer wie in sehr vielen Dimensionen völlig gleich. Es reichte daher in den folgenden Kapiteln aus, seine Eigenschaften an ein- oder zweidimensionalen Datensätzen zu demonstrieren.

Im zweiten Abschnitt von Kapitel 1 wurde versucht, die Lernregel des *univar* aus biologischen Tatsachen zu motivieren. Dabei wurde klar, dass es sich hierbei um ein sehr stark abstrahiertes Modell für die Reizverarbeitung in Gehirnen biologischer Lebewesen handelt. Dieser Abschnitt diente als Anknüpfungspunkt für die später anhand der *Aplysia* in Kapitel 3 vorgenommenen Modifikationen am *univar*-Algorithmus, die seine on-line Lerneigenschaften möglich machen sollten.

In *Kapitel 2* wurde das dynamische Verhalten des *univar*-Algorithmus untersucht, der

zwei relevante Parameter, σ und ε besitzt. In der *ursprünglichen* Formulierung bei randomisiertem Lernen steuern die beiden Parameter *getrennt* seine räumlichen und seine zeitlichen Eigenschaften. Die räumlichen Eigenschaften äußern sich in Aufspaltungsphänomenen, den Phasenübergängen, die zeitlichen bestehen in gleitender Mittelung über den Datenstrom, was bei kleinen Werten von ε starke Retardierungen hervorrufen kann.

In der *on-line* Formulierung des *univar* konnte ich zunächst einen *dynamischen Phasenübergang* finden, der aus dem Bereich der dynamischen Kopplung heraus geschieht. Die Kopplung wird bei dem Auftreten der Phasentrennung aufgehoben. Dieser Phasenübergang kann sowohl durch *Annealing* von σ als auch von ε hervorgerufen werden. Daraus ergab sich die Phasenübergangskurve als gegenseitige Abhängigkeit der kritischen Parameter. Im Laufe von Kapitel 2 konnte ich die genaue Position dieser Kurve im Parameterraum auch quantitativ durch mehrere analytische Näherungen angeben. Das Ergebnis dieser Überlegungen war die Darstellung des on-line lernenden *univar* als einen Algorithmus, der durch die bewusste Wahl seines zeitlichen Weges im Parameterraum Zeitskalen im System zu bestimmen vermag.

In *Kapitel 3* wurde zunächst das Gewöhnungs- und Sensibilisierungsverhalten der Meeresschnecke *Aplysia* skizziert. Dieses wurde – in ähnlich abstrakter Weise wie bei der neuronalen Motivation des *multivar* – als ein Mittel zur selbstorganisierten Datenfilterung erkannt.

In *Kapitel 4* wurde dieses Prinzip der selbstreferentiellen Datenfilterung durch eine modifizierte *univar*-Lernregel verwirklicht. Der daraus resultierende Algorithmus, ANTS, bestimmt die Relevanz eines Datenpunktes an seinem eigenen effektiven Modell der bislang gelernten Datenpunkte, indem er mittels einer Aufmerksamkeitsschwelle seinen Neuigkeitsgehalt bestimmt. Dadurch ist er in der Lage, verschiedene Lebensdauern von Zuständen im on-line Datenstrom auch unterschiedlich zu behandeln. So vermeidet er sowohl Retardierungseffekte durch zu kleines ε, als auch die Kopplung der Lern- an die Systemdynamik, und kann in vielen Fällen das Lernziel, die Bildung eines effektiven Modells der präsentierten Daten, erreichen.

Der ANTS ist vermutlich der erste on-line Algorithmus, der selbstorganisiert auf verschiedene Zeitskalen reagiert. Er darf deshalb als ein erster Schritt in Richtung on-line Datenverarbeitung angesehen werden.

Appendix A

Gedächtniskerne

Hier soll aus der Differenzengleichung (2-2) bzw. der Differentialgleichung 2-4 die Form der Gedächtniskerne $g_r(t,t')$ hergeleitet und anhand einiger Spezialfälle veranschaulicht werden. Dazu wird die Lernregel (2-2) zunächst in etwas anderer Form geschrieben,

$$\mathbf{c}_r(t+1) = \big(1-\varepsilon(t)a_r(\mathbf{x}_t)\big)\,\mathbf{c}_r(t) + \varepsilon(t)a_r(\mathbf{x}_t)\,\mathbf{x}_t. \tag{A-1}$$

Hier sieht man deutlicher, dass der aktuelle Punkt $\mathbf{x}_t$ mit der Gewichtung $\varepsilon a_r(\mathbf{x}_t)$ in die Neuberechnung von $\mathbf{c}_r$ eingeht. Der Gedächtniskern soll nun diese Gewichtungen für alle vergangenen Punkte $\mathbf{x}_{t'}$ mit $t'<t$ angeben.

Dazu wird die Lernregel (A-1) rekursiv in sich selbst eingesetzt. Man bekommt, wenn man abkürzend

$$\kappa_t := \varepsilon(t)\,a_r(\mathbf{x}_t) \tag{A-2}$$

schreibt und den Index r weglässt,

$$\begin{aligned}
\mathbf{c}(t) &= \mathbf{c}(t-1)\,(1-\kappa_{t-1}) + \mathbf{x}(t-1)\,\kappa_{t-1} && \text{(A-3)}\\
&= \mathbf{c}(t-2)\,(1-\kappa_{t-1})(1-\kappa_{t-2}) \\
&\quad + \mathbf{x}(t-1)\,\kappa_{t-1} \\
&\quad + \mathbf{x}(t-2)\,\kappa_{t-2}(1-\kappa_{t-1}) && \text{(A-4)}\\
&= \mathbf{c}(t-3)\,(1-\kappa_{t-1})(1-\kappa_{t-2})(1-\kappa_{t-3}) \\
&\quad + \mathbf{x}(t-1)\,\kappa_{t-1} \\
&\quad + \mathbf{x}(t-2)\,\kappa_{t-2}(1-\kappa_{t-1}) \\
&\quad + \mathbf{x}(t-3)\,\kappa_{t-3}(1-\kappa_{t-2})(1-\kappa_{t-1}) && \text{(A-5)}\\
&\;\;\vdots \\
&= \mathbf{c}(0)\prod_{t'=0}^{t-1}(1-\kappa_{t'}) \quad + \quad \sum_{t'=0}^{t-1}\mathbf{x}(t')\,\kappa_{t'}\prod_{t''=t'+1}^{t-1}(1-\kappa_{t''}) && \text{(A-6)}\\
&=: \mathbf{c}(0)k(t) \quad + \quad \sum_{t'=0}^{t-1}\mathbf{x}(t')\,g(t,t'). && \text{(A-7)}
\end{aligned}$$

Im letzten Schritt wurden die beiden Terme k und g definiert, die bereits in Gleichung (2-9) verwendet wurden.

Man sieht, dass das linke Produkt in (A-6), also $k(t)$, implizit von allen bisherigen κ-Termen, explizit aber nur noch von dem Zeitparameter t abhängt. Es zerfällt wegen $(1-\kappa_t)<1$ mit fortschreitendem t immer weiter (exponentiell für konstantes κ).

Der Faltungskern $g(t,\cdot)$ in der rechten Summe ist dagegen der Gedächtniskern. Er gibt, wie man sofort sieht, die Gewichtung jedes vergangenen Punktes $\mathbf{x}_{t'}$ in der Summe zur Berechnung von $\mathbf{c}(t)$ an. Schon an (A-1) sieht man sofort, dass der letzte Gewichtungsfaktor

$$g(t+1,t) = \varepsilon a_r(\mathbf{x}_t) \tag{A-8}$$

ist. Die um eins verschobenen Argumente von g rühren daher, dass $\mathbf{c}$ und $\mathbf{x}$ formal einen um eins verschobenen zeitlichen Index tragen.

An (A-1) sieht man unmittelbar, wie ein Gedächtniskern algorithmisch bestimmt werden kann:

(G-1) Initialisiere einen Gedächtniskern $g(0,\cdot)=0$, setze $t=1$

(G-2) Bestimme κ_t zum aktuellen Zeitpunkt t

(G-3) Multipliziere den gesamten Kern mit $(1-\kappa_t)$ und setze $g(t,t-1)=\kappa_{t-1}$

(G-4) Zähle t weiter und gehe zu (G-2).

Auf diese Weise wurden alle in dieser Arbeit dargestellten Gedächtniskerne ermittelt.

Typische *Annealing*-Schemata

Für ein paar Spezialfälle ist es sinnvoll, den Effekt zu demonstrieren, den die Veränderung von $\varepsilon(t)$ auf die Form der Gedächtniskerne hat. Um analytische Berechnungen anstellen zu können, muss dazu das lerndende MVNN auf ein Neuron reduziert werden ($a=1$). Die Ergebnisse dieser Berechnungen sind in Abb. 20 zu sehen.

- Eine konstante Lernrate ε führt wegen Gleichung (19) zu

$$\mathbf{c}_r(t) = (1-\varepsilon)^t\,\mathbf{c}(0) + \varepsilon\sum_{t'=1}^{t}(1-\varepsilon)^{t-t'}\,\mathbf{x}(t'), \tag{A-9}$$

  also einer Repräsentation mit exponentiell zerfallendem Gedächtniskern.

- Eine Lernrate von $\varepsilon(t)=1/t$ führt zu einer Repräsentation, die alle gesehenen Datenpunkte gleich gewichtet,

$$\mathbf{c}_r(t) = \frac{1}{t}\sum_{t'=1}^{t}\mathbf{x}(t'), \tag{A-10}$$

  wie man durch Induktion leicht nachrechnen kann. Dies ist die sequentielle Formulierung einer arithmetischen Mittelwertsberechnung.

– Eine exponentiell abnehmende Lernrate $\varepsilon(t) = \varepsilon(0)\exp(-t/\tau)$, wie sie momentan im *multivar*-Algorithmus implementiert ist, führt bei $\tau \gg t$ zu einem exponentiell zerfallenden Gedächtniskern ähnlich demjenigen in Gleichung (A-9). Denn in diesem Fall kann von

$$\varepsilon(t) = \varepsilon(0)\, e^{-t/\tau} \approx \varepsilon(0) \tag{A-11}$$

ausgegangen werden. Er wird mit steigendem t langsam länger.

– Wenn dagegen schnell exponentiell abgekühlt wird, werden die Lernraten nach $t \gg \tau$ Iterationsschritten so klein, dass praktisch nichts mehr hinzugelernt wird. Man bekommt dann eine zerfallenden Gewichtung, also einen Gedächtniskern, der dem vorigen gerade entgegengesetzt ist. Ausgehend von einem Lernparameter

$$\varepsilon(t) = (1-e^{-1/\tau})e^{-t/\tau} = (1-q)q^t, \tag{A-12}$$

wobei $q := e^{-1/\tau} < 1$ gelten soll, bekommt man unter der Annahme $t \gg \tau$ eine sehr kleine Lernrate, also verändern sich die Codebuchzentren kaum noch. Dann ist $\mathbf{c}(t+1) \approx \mathbf{c}(t)$ und die Lernregel für die Zentren kann wiefolgt genähert werden,

$$\begin{aligned} \mathbf{c}(t) &= \mathbf{c}(t-1) + \varepsilon(t)(\mathbf{x}(t) - \mathbf{c}(t-1)) \\ \mathbf{c}(t) &\approx (1-q^t)\mathbf{c}(t-1) + q^{t+1}\mathbf{c}(t) + q^t(1-q)\mathbf{x}(t) && \text{(A-13)} \\ &= \sum_{t'=1}^{t} \mathbf{x}(t') e^{-\frac{t'}{\tau}} \Big/ \sum_{t'=1}^{t} e^{-\frac{t'}{\tau}} \quad \text{für } t \gg \tau. && \text{(A-14)} \end{aligned}$$

Es gehen also fast ausschließlich die ersten Datenpunkte in die Repräsentation ein, da die Schrittweiten später zu klein werden.

Der generische Fall im *univar*-Algorithmus ist derjenige, in dem ε langsam verkleinert wird. Dabei bekommt man Gedächtniskerne, die etwa denen mit konstantem ε entsprechen. Wegen

$$(1-\varepsilon)^{t-t'} = \exp\big((t-t')\ln(1-\varepsilon)\big) = \exp\Big(-\frac{t-t'}{\tau}\Big) \quad \text{und} \tag{A-15}$$

$$1/\tau = -\ln(1-\varepsilon) \approx \varepsilon \tag{A-16}$$

ist die Länge dieses Gedächtniskernes etwa $1/\varepsilon$. Er „gleitet“ bei der Mittelung über den Datenstrom, wobei er sich selbst auch leicht verändert und länger wird, weil er parametrisch von t abhängt. Auf diese Weise bekommt man das erwähnte Mittelungsfenster als den Träger des Gedächtniskerns, in den oben besprochenen Beispielen immer das Intervall $[0, t-1]$. In diesem Fenster wird mit $g_r(t,\cdot)$ als Gewichtung der Mittelwert aller Datenpunkte berechnet.

Appendix B

Einige einfache Modelle

Gesucht ist der Zusammenhang der kritischen Werte $\tau_S^{\text{krit}} := \varepsilon^{\text{krit}} T_S/M$ und $\sigma_{\text{krit}}/\sigma_S$, die den Phasenübergang charakterisieren. Während einer Zeitspanne T_S kommen alle Daten von einem einzigen Punkt (hier von $+1$), anschließend wieder T_S lang nur von Punkt (-1). Diese Datenverteilung wird von einer Mischung aus zwei Normalverteilungen gelernt, die beide die Breite σ und die Lernrate ε haben. Der Vorgang wird so oft wiederholt, bis der Aufspaltungsprozess (durch beliebig langsame Parametervariation bewirkt) stattfindet.

B.1 Eine kontinuierliche Näherung für mittlere τ_S

Während eines Zeitabschnitts, bei dem nur die Quelle bei $(+1)$ feuert, lauten die Differenzengleichungen von *univar*

$$\begin{aligned} \Delta c_1(t) &= \varepsilon a_1(t)\,(1 - c_1(t)) \\ \Delta c_2(t) &= \varepsilon a_2(t)\,(1 - c_2(t)). \end{aligned} \tag{B-1}$$

Wenn man stattdessen den Mittelwert $\mu := (c_1 + c_2)/2$ und den halben Abstand $\xi := (c_1 - c_2)/2$ der Zentren betrachtet, erhält man eine wesentlich symmetrischere Form,

$$\begin{aligned} \frac{d}{d\tau}(1-\mu) &= \Big(-(1-\mu) + \xi \tanh\big(\tfrac{(1-\mu)\xi}{\sigma^2}\big)\Big)\,, \\ \frac{d}{d\tau}\xi &= \Big(-\xi + (1-\mu)\tanh\big(\tfrac{(1-\mu)\xi}{\sigma^2}\big)\Big)\,. \end{aligned} \tag{B-2}$$

Dabei wurde die differentielle Näherung in (2-21) verwendet. Wenn man die entsprechende Differentialgleichung für den Ausdruck $\big(\xi^2 - (1-\mu)^2\big)$ aufschreibt, findet man die Gleichung eines einfachen exponentiellen Zerfalls, deren Lösung auf die im folgenden nützliche Gleichung

$$\xi^2(\tau) - \big(1 - \mu(\tau)\big)^2 = e^{-2\tau}\big(\xi(0)^2 - (1-\mu(0))^2\big) \tag{B-3}$$

führt.

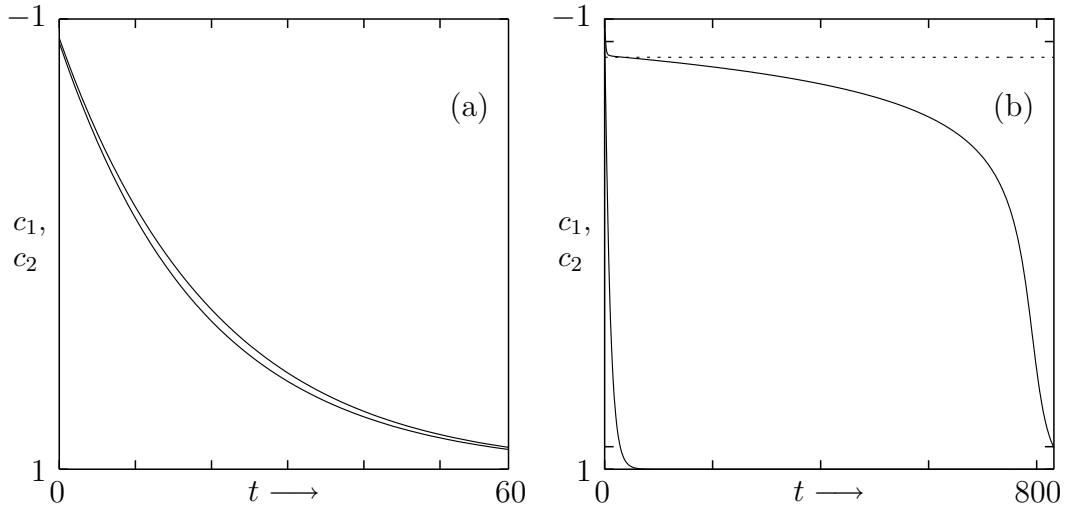


Abbildung 51: Numerische Integration der Gleichung (B-1) mit $\varepsilon = 0.1$ und $\sigma = 0.9$ (a). Hier ist die Approximation (B-9) für mittleres τ_S gültig, der Abstand der beiden Linien wird während T_S kaum größer als an den Endpunkten, während sie in (b) mit $\sigma = 0.5$ den klassischen Anwendungsfall der Approximation für sehr große τ_S darstellt. Sie wird auf die Abb. 52 führen, in der das bewegliche Neuron seinen Anfangspunkt auf der gestrichelten Linie bei $t \approx 50$ hat.

In der entarteten Phase gilt immer $\xi(0) = \xi(\tau_S) = 0$. In der anderen Phase gibt es dagegen von Null verschiedene Abstände, die während einer Periode *stabil* bleiben,

$$\xi(0) = \xi(\tau_S) > 0. \tag{B-4}$$

Jeder der dieser Abstände hat ein nach Gleichung (B-3) zugehöriges $\mu(0)$, welches die Bewegung periodisch und symmetrisch macht ($\mu(\tau_S) = -\mu(0)$). Nahe an der Phasenübergangskurve unterscheidet sich diese Bewegung nur minimal vom Grenzzyklus der entarteten symmetrischen Codebuchbewegung.

Der Umkehrpunkt dieser entarteten Bewegung ist schnell aus (B-3) mit $\xi = 0$ bestimmt. Mit $\mu(\tau_S) = -\mu(0)$ bekommt man

$$\mu(0) = -\tanh\Big(\frac{\tau_S}{2}\Big). \tag{B-5}$$

Um zu untersuchen, ob sich $\mu(0)$ bei leichter Aufspaltung ändert, löst man (B-3) nach $\mu_0 := \mu(0)$ auf und entwickelt diesen Anfangswert, als Funktion von $\xi_0 := \xi(0)$ aufgefasst, nach Taylor. Unter der Annahme $\xi(\tau) \approx \xi_0$, liefert diese keinen konstanten und keinen linearen Term. Deshalb wird in diesem Fall die folgende *Definition des infinitesimalen Phasenübergangs* (Typ I) verwendet:

$$\begin{aligned} &\xi(0) = \xi(\tau_S) \quad \text{mit} \quad \xi(0) \to 0 \quad \text{und} \\ &\mu(0) \text{ ist periodischer Anfangswert für } \xi = 0. \end{aligned} \tag{B-6}$$

Dabei sei o. E. $\xi > 0$, $\mu(0) < 0$ und $x = 1$ während $\tau \in (0, \tau_S)$.

Nun wird ξ nach (B-2) iteriert, wobei die Differentialgleichung für ξ zunächst in Fixpunktdarstellung aufgeschrieben und anschließend (B-3) eingesetzt wird,

$$\xi(\tau_S) = \xi_0 + \sum_{\tau=0}^{\tau_S} \xi(\tau+\varepsilon/M) - \xi(\tau) \approx \xi_0 + \int_0^{\tau_S} d\tau \frac{d\xi}{d\tau} \tag{B-7}$$

$$= \xi_0 + \int_0^{\tau_S} d\tau \Bigg[-\xi(\tau) + \sqrt{\xi(\tau)^2 + e^{-2\tau}((1-\mu_0)^2 - \xi_0^2)} \tanh\left(\frac{\xi(\tau)}{\sigma^2}\sqrt{\xi(\tau)^2 + e^{-2\tau}((1-\mu_0)^2 - \xi_0^2)}\right)\Bigg]. \tag{B-8}$$

Um dieses Integral näherungsweise berechnen zu können, soll nun eine Taylorentwicklung von $\xi(\tau;\xi_0)$ in seinem Anfangswert ξ_0 durchgeführt werden. Sie weist keinen konstanten Term auf, ferner wird angenommen, dass sie linear mit der Steigung 1 beginnt. Es wird hier also angenommen, dass der Abstand $\xi(\tau)$ während der gesamten Periode $\tau \in (0, \tau_S)$ klein und $\xi(\tau) \approx \xi_0$ bleibt. Diese Näherung ist in Abbildung 51a sicher erfüllt, in 51b dagegen nicht. Da die hier verwendete Phasenübergangsdefinition vom infinitesimalen Typ ist, handelt es sich um keine starke Einschränkung. Unter Vernachlässigung von Termen $O(\xi^2)$ bekommt man

$$\xi(\tau_S) \approx \xi_0 \left[1 + \int_0^{\tau_S} d\tau \left(-1 + \frac{(1-\mu_0)^2}{\sigma^2} e^{-2\tau}\right)\right] \tag{B-9}$$

$$= \xi_0 \left[1 - \tau_S + \frac{(1-\mu_0)^2}{2\sigma^2}(1 - e^{-2\tau_S})\right]. \tag{B-10}$$

Nach σ aufgelöst wird diese Gleichung zur Phasenübergangsbedingung (2-27).

B.2 Die diskrete Formulierung der Näherung für mittlere τ_S

Hier soll nun die diskrete Version von Gleichung (2-27) gefunden werden. Dazu werden die einzelnen Schritte der Berechnung oben statt mit Integralen mit Summen durchführt. Wieder ist eine exakte Lösung in der Näherung $\xi(\tau) \approx \xi(0)$ möglich.

Das direkt aus (B-1) durch Transformation auf den Mittelwert $\mu := (c_1 + c_2)/2$ und den halben Abstand $\xi := (c_1 - c_2)/2$ zu bestimmende Pendant zu Differentialgleichung (B-2) lautet

$$\begin{aligned} \Delta(1-\mu) &= \frac{\varepsilon}{2}\left(-(1-\mu) + \xi \tanh\left(\tfrac{(1-\mu)\xi}{\sigma^2}\right)\right), \\ \Delta\xi &= \frac{\varepsilon}{2}\left(-\xi + (1-\mu)\tanh\left(\tfrac{(1-\mu)\xi}{\sigma^2}\right)\right). \end{aligned} \tag{B-11}$$

Durch Iteration der ersten Gleichung bei $\xi = 0$ erhält man den symmetrischen Anfangswert $\mu(0)$,

$$1 + \mu_0 \stackrel{!}{=} 1 - \mu(T_S) = (1 - \varepsilon/2)^{T_S}(1 - \mu_0), \quad \text{woraus folgt,} \tag{B-12}$$

$$\mu_0 = -\frac{1 - (1-\varepsilon/2)^{T_S}}{1 + (1-\varepsilon/2)^{T_S}}. \tag{B-13}$$

Dies ist die diskrete Entsprechung von (B-5). Danach wird wieder die erste Grenze des Phasenübergangs (Instabilität der Entartung) mit dem linearen Ansatz in (B-9) bestimmt. Hier muss nun summiert werden, nicht mehr integriert, was mit Hilfe der geometrischen Summenformel das Ergebnis

$$\sigma_{\text{krit}}(T_S, \varepsilon) = \left(1 + \frac{1 - (1-\varepsilon/2)^{T_S}}{1 + (1-\varepsilon/2)^{T_S}}\right) \sqrt{\frac{1}{T_S} \, \frac{1 - (1-\varepsilon/2)^{2T_S}}{1 - (1-\varepsilon/2)^2}} \tag{B-14}$$

liefert. Dies ist die diskrete Entsprechung von (2-27).

B.3 Eine Näherung für sehr große τ_S

Zur Zeit $t = 0$ sei die Situation aus Abb. 28c gegeben,

$$c_1(0) = c_0, \quad \text{und} \quad c_2(0) = 1.$$

Es bleibt dann $c_2(t) = 1$ unverändert, während c_1 nach der Lernregel

$$\frac{\Delta c_1(t)}{\Delta t} \approx \frac{dc_1(t)}{dt} = \frac{\varepsilon(1 - c_1(t))}{1 + \exp\left\{\frac{(1-c_1(t))^2}{2\sigma^2}\right\}} \tag{B-15}$$

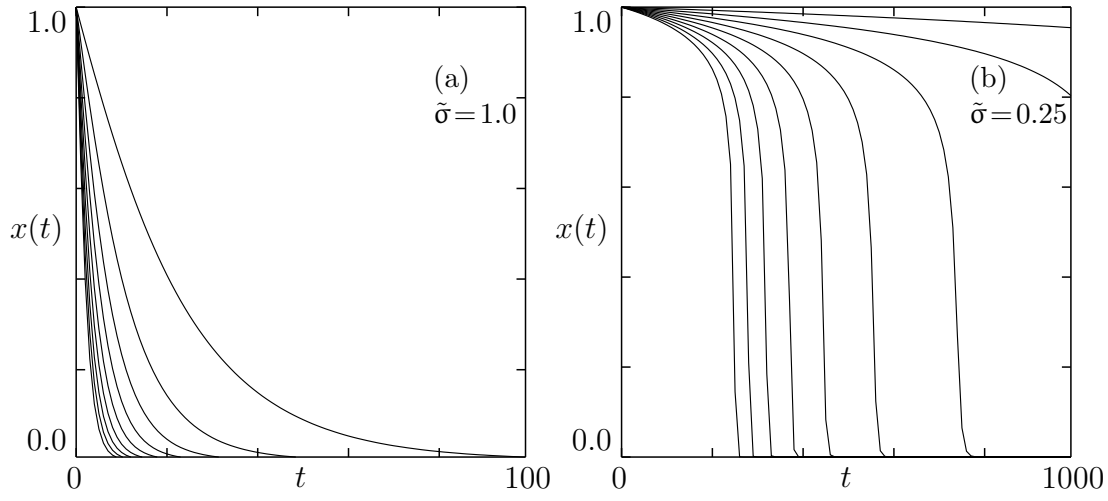


Abbildung 52: Numerische Lösung des Anfangswertproblems (B-16) mit (a) $\tilde{\sigma} = 1.0$ (b) $\tilde{\sigma} = 0.25$ und jeweils $\varepsilon = 0.1$ (oberste) bis 0.9 (unterste Linie)

berechnet wird. Dann löst $x(t) := (1 - c(t))/(1 - c_0)$ das Anfangswertproblem

$$\begin{aligned} x(0) &= 1 \\ \frac{d}{dt}x(t) &= -\frac{\varepsilon x(t)}{1 + \exp\{\frac{x(t)^2}{2\tilde{\sigma}^2}\}} \end{aligned} \tag{B-16}$$

mit $\tilde{\sigma} := \sigma/(1 - c_0)$. Abbildung 52 zeigt numerisch bestimmte Lösungen dieser Differentialgleichung für verschiedene Werte von $\tilde{\sigma}$ und ε. Der Ansatz $x(t) = \exp(z(t))$ führt auf die Differentialgleichung für z

$$\frac{d}{dt}z(t) = -\frac{\varepsilon}{1 + \exp\{\frac{x(t)^2}{2\tilde{\sigma}^2}\}}\ , \tag{B-17}$$

die sich durch Variablentrennung und Integration formal lösen lässt,

$$dt = -\frac{1}{\varepsilon}(1 + e^{\frac{x^2}{2\tilde{\sigma}^2}})\, dz, \tag{B-18}$$

$$-\varepsilon t = -\varepsilon \int_0^t dt' = z(t) - z(0) + \int_{z(0)}^{z(t)} e^{\frac{x^2}{2\tilde{\sigma}^2}}\, dz$$

$$-\varepsilon t = z(t) + \frac{1}{2}\,\mathrm{Ei}\Big(\frac{e^{2z(t)}}{2\tilde{\sigma}^2}\Big) - \frac{1}{2}\,\mathrm{Ei}\Big(\frac{1}{2\tilde{\sigma}^2}\Big). \tag{B-19}$$

Ei ist das uneigentliche Exponentialintegral (vgl. Formeln 5.1.2 und 5.1.11 in Abramowitz & Stegun, 1972)

$$\mathrm{Ei}(x) := \fint_{-\infty}^{x} \frac{e^{\xi}}{\xi} d\xi = \gamma + \ln x + \sum_{k=1}^{\infty} \frac{x^k}{k \cdot k!}, \quad x > 0, \tag{B-20}$$

womit man folgenden Zusammenhang zwischen verstrichener Zeit t und Ort z bekommt,

$$\varepsilon t = -2z(t) + \frac{1}{2}\sum_{k=1}^{\infty} \frac{(2\tilde{\sigma}^2)^{-k}}{k \cdot k!}(1 - e^{2kz(t)}). \tag{B-21}$$

Bis hier ist der Ausdruck vollkommen exakt. Alle Approximationen können von dieser Gleichung ausgehen.

- Wenn man nach der mittleren Anpassungszeit τ_L an die Daten fragt, dann ist $z(\tau_L) = -1$, und

$$2\tau_L = \varepsilon t(z = -1) = 2 + \frac{1}{2}\sum_{k=1}^{\infty} \frac{(2\tilde{\sigma}^2)^{-k}}{k \cdot k!}(1 - e^{-2k}). \tag{B-22}$$

  Wie man in Abbildung 52 sieht, findet die Anpassung bei großem $\tilde{\sigma}$ annähernd exponentiell statt, während bei kleinem $\tilde{\sigma}$ fast der gesamte Weg praktisch zu einem festen Zeitpunkt zurückgelegt wird. Im ersten Fall ist τ_L die exponentielle Relaxationszeit, im zweiten Fall diejenige Zeit, während der fast der gesamte Übergang vom Anfangswert zur Datenquelle stattfindet.

– Wenn man eine Stabilitätsbedingung für die Aufspaltung im Codebuch sucht, wird die Lebensdauer T_S der Quelle (± 1) wichtig. Während dieser Zeitspanne darf der Abstand der Codebuchzentren nicht zu klein werden. Er muss, verglichen mit $\tilde{\sigma}$, *relativ groß* bleiben. Der Phasenübergang vom Typ II wird so definiert, dass der minimale Abstand der Zentren während einer Periode, also an den Umkehrpunkten, immer gleich bleibt, und zwar $s\tilde{\sigma}$, wobei s ein noch unbestimmter Parameter ist. Man bekommt als Kriterium

$$x(T_S) \stackrel{!}{=} s\tilde{\sigma} \quad \text{oder} \quad z(T_S) \stackrel{!}{=} \ln(s\tilde{\sigma}), \quad \text{und nach Gl. (B-19)} \tag{B-23}$$

$$\begin{aligned} 2\tau_S = \varepsilon T_S \stackrel{!}{=} \varepsilon t(z{=}\ln s\tilde{\sigma}) &= -\ln(s\tilde{\sigma}) - \frac{1}{2}\,\mathrm{Ei}\Big(\frac{s^2}{2}\Big) + \frac{1}{2}\,\mathrm{Ei}\Big(\frac{1}{2\tilde{\sigma}^2}\Big) \\ &= -2\ln(s\tilde{\sigma}) + \frac{1}{2}\sum_{k=1}^{\infty}\frac{1}{kk!}\Big((2\tilde{\sigma}^2)^{-k} - (s^2/2)^k\Big)\,. \end{aligned} \tag{B-24}$$

Dies ist das oben verwendete Ergebnis (2-29).

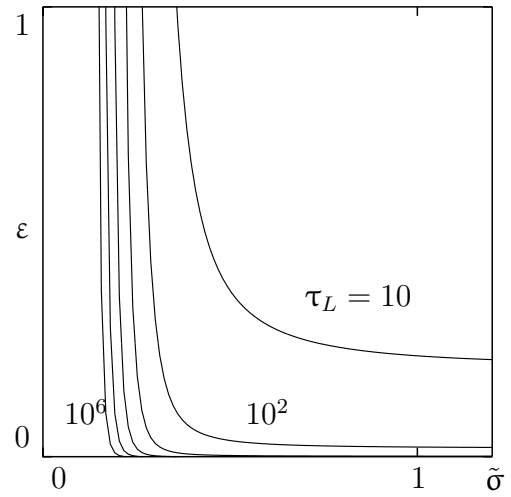


Abbildung 53: Konturenplot der Zeiten τ_L, die das bewegliche Neuron nach Gleichung (B-22) benötigt, um einen Anteil von $(1/e)$ der Strecke zur Datenquelle, wo auch das andere Neuron sitzt, zurückzulegen.

Appendix C

Ergebnisse der Variationsrechnung

In diesem Anhang wird eine der zentralen Eigenschaften des ANTS-Algorithmus begründet, nämlich dass er den unteren Stumpf einer Verteilungsdichte p approximiert. Die dafür verwendete Methode ist die Variationsrechnung. Da sich dieser Kalkül aber nicht auf beliebige Funktionen p anwenden lässt, gelten die im folgenden gemachten Aussagen nicht in aller mathematischer Strenge, sondern sind als Heuristiken zu verstehen, die in vielen Fällen unter Verwendung von ausreichend gutartigen Funktionenklassen richtig sind.

Zunächst möchte ich an einem einfachen Beispiel demonstrieren, wie die Verwendung des Variationskalküls viele ansonsten schwer zu beweisende Aussagen vereinfacht. Anschließend soll das stationäre Verhalten des ANTS-Algorithmus untersucht und die Behauptung (4-14) gezeigt werden.

C.1 Warum eine ML-Schätzung die Verteilungsdichte approximiert

In Abschnitt 1.1 wurde klar, dass der stationäre Zustand des *multivar*-Algorithmus, der durch die Gleichungen (1-13) bis (1-15) charakterisiert ist, eine *Maximum-likelihood*-Dichteschätzung darstellt. Anhand der Erklärung des ML-Prinzips auf den Seiten 15f wurde klar, dass es sich bei $\hat{p}$ tatsächlich um ein Modell der Verteilungsdichte p handelt, die dem Datensatz zugrundeliegt. Diese Frage, ob p und $\hat{p}$ im Idealfall eines vollständigen Datensatzes und einer beliebig guten Approximationsfähigkeit durch $\hat{p}$ gleich werden können, wurde von Dersch (1995) beantwortet, der die mögliche Gleichheit von p und $\hat{p}$ feststellte. Als Bedingung nannte er, dass die Breiten der verwendeten Normalverteilungen von der gleichen Größenordnung sein sollen wie ihre Abstände. Mit dem ML-Prinzip ist diese Bedingung noch genauer gefasst worden.

In diesem Abschnitt möchte ich nun mit einer formalen Methode das Ergebnis $\hat{p} = p$ für beliebig genaue Approximierbarkeit durch $\hat{p}$ (d. h. beliebig viele Neuronen) erneut ableiten. Dazu betrachtet man die *Log-Likelihood* aus (1-7) und erweitert sie zu dem

Funktional

$$E\,[\hat{p}] := \int_{\mathcal{M}} p\,\ln\hat{p}. \tag{C-1}$$

Dieses Log-likelihood-Funktional wird unter der Nebenbedingung maximiert, dass $\hat{p}$ immer auf eins normiert bleibt,

$$\int_{\mathcal{M}} \hat{p} = 1. \tag{C-2}$$

Da diese Nebenbedingung ebenfalls eine Integralbedingung ist, handelt es sich um ein *isoperimetrisches Variationsproblem*. Man kann die Maximierung von E, eingeschränkt auf die durch (C-2) definierte Mannigfaltigkeit, durch Maximierung des Funktionals L mit einem reellen Lagrange-Parameter λ,

$$L\,[\hat{p}] = \int p\ln\hat{p} \;+\; \lambda\left(\int \hat{p} - 1\right), \tag{C-3}$$

ersetzen (siehe Tapia & Thompson, 1978, Appendix I oder Klingbeil, 1988). Wenn man nun die Definition der Variationsableitung $\delta L/\delta\hat{p}$ verwendet (Bishop, 1995, Appendix D),

$$\delta L = L\,[\hat{p}+\delta\hat{p}] - L\,[\hat{p}] =: \int_{\mathcal{M}} \frac{\delta L}{\delta\hat{p}}\,\delta\hat{p}\;d\mathbf{x} + \mathcal{O}(\delta\hat{p}^2), \tag{C-4}$$

dann bekommt man die notwendige Bedingung für die Maximierung von L,

$$\frac{\delta L}{\delta\hat{p}} = \frac{p}{\hat{p}} + \lambda \stackrel{!}{=} 0. \tag{C-5}$$

Aus $p=-\lambda\,\hat{p}$ und der Normierung von p folgt sofort der Wert des Lagrange-Parameters $\lambda=-1$ und die Behauptung der Dichteschätzung,

$$\hat{p} = p. \tag{C-6}$$

Dies gilt natürlich nur, wenn man $\hat{p}$ tatsächlich beliebig „nahe“ an p bringen kann, also wenn p ebenfalls eine endliche Mischung von Normalverteilungen ist. Ansonsten müsste die Variation auf die Menge der Mischungen von Normalverteilungen eingeschränkt werden, was schwieriger aufzuschreiben ist. Man bekommt dann M Lagrange-Parameter (einen für jedes Codebuchzentrum), was letztendlich nur wieder den stationären Zustand (1-13) der Lernregeln wiederholt. Wenn man die zweite Ableitung des Funktionals bestimmt, so bekommt man

$$\frac{\delta^2 L}{\delta\hat{p}^2} = -\frac{p}{\hat{p}^2} < 0, \tag{C-7}$$

was als Funktion kleiner als Null ist, denn es ist auch

$$\begin{aligned} &\hat{p}(\mathbf{x}) > 0 \quad \text{für alle } \mathbf{x}\in\mathcal{M}, \text{ und} \\ &p(\mathbf{x}) > 0 \quad \text{für mindestens ein } \mathbf{x}\in\mathcal{M}. \end{aligned}$$

Mit Gleichung (C-7) ist also die asymptotische Stabilität des stationären Zustandes $p=\hat{p}$ von *multivar* bewiesen.

C.2 Warum selbstreferentielles Lernen die Datenverteilungsdichte abschneidet

Behauptung

Der ANTS, bestehend aus den Schritten (TS-1) bis (TS-6) auf Seite 80, oder kurz,

$$\mathbf{c}_r(t+1) = \mathbf{c}_r(t) + \varepsilon\, f(\mathbf{x}_t)\, a_r(\mathbf{x}_t)\, \big(\mathbf{x}_t - \mathbf{c}_r(t)\big) \quad \text{mit} \tag{C-8}$$

$$f(\mathbf{x}_t) := \Theta(\epsilon_A - A(\mathbf{x}_t)) = \begin{cases} 1 & \text{für } A(\mathbf{x}_t) < \epsilon_A \\ 0 & \text{sonst,} \end{cases} \tag{C-9}$$

angewandt auf eine stationäre Quelle von Datenpunkten mit Verteilungsdichte p führt im Grenzübergang beliebig guter Approximation, also $M \to \infty$, zu einer Schätzung des α-Stumpfes p^c,

$$A \to p^c := \begin{cases} \epsilon_A & \text{für } p(\mathbf{x}) > \alpha\epsilon_A, \\ p(\mathbf{x})/\alpha & \text{sonst.} \end{cases} \tag{C-10}$$

Dabei soll o. E. angenommen werden, dass p glatt ist, also $\hat{p}$ bis auf den Rand $\{p(\mathbf{x}) = \alpha\epsilon_A\}$ des Plateaus ebenfalls. Dann gilt der Grenzübergang in (C-10) sogar punktweise.

Beweisidee

Die Beweisidee soll für den stationären Fall formuliert werden. Dann ist die logische Reihenfolge der im folgenden verwendeten Größen:

1. p und ϵ_A seien vorgegeben; in jedem Zeitschritt wird ein $\mathbf{x}$ gezogen, es werden $A(\mathbf{x})$ und $f(\mathbf{x})$ berechnet.

2. Im stochastisch stationären Endzustand bewegt sich das Codebuch immer ein wenig, und man kann für $\mathbf{x}$ die Wahrscheinlichkeit berechnen, gelernt zu werden. Sie ist der (Ensemble-)Erwartungswert von $f(\mathbf{x})$,

   $$\omega_A(\mathbf{x}) := \langle f(\mathbf{x}) \rangle_{\text{ens}}\,. \tag{C-11}$$

   Da f über die Heavyside-Funktion Θ definiert ist, und da der Ensemblemittelwert bei kleinem ε lediglich eine kleine Verschmierung bedeutet, kann man ω_A als durch ε „aufgeweichte“ Heavyside-Funktion Θ_ε schreiben,

   $$\begin{aligned} \omega_A(\mathbf{x}) &= \langle \Theta(\epsilon_A - A(\mathbf{x})) \rangle_{\text{ens}} \\ &\approx \Theta_\varepsilon(\epsilon_A - A(\mathbf{x})). \end{aligned} \tag{C-12}$$

   Die Funktion ω_A ist in Abb. 42c dargestellt. Sie teilt den Merkmalsraum in zwei Mengen ein, die als „inneres“ und „äußeres“ zu verstehen sind. In Γ_o wird jeder Punkt gelernt, in Γ_i wird selektiert,

   $$\begin{aligned} \Gamma_o &\quad \text{Menge aller } \mathbf{x} \in \mathcal{M} \text{ mit } \Theta(\epsilon_A - A(\mathbf{x}) = 1) \\ \Gamma_i &\quad \text{Menge aller } \mathbf{x} \in \mathcal{M} \text{ mit } \Theta(\epsilon_A - A(\mathbf{x}) < 1). \end{aligned}$$

Da die Funktion Θ_ε sowohl in Abhängigkeit von $\mathbf{x}$ als auch von A aufgefasst werden kann, benutze ich im folgenden unterschiedliche Notationen für diese beiden Bedeutungen,

$$\begin{array}{rll} \mathbf{x} & \mapsto \omega_A(\mathbf{x}) & \text{als Funktion von } \mathbf{x}, \text{ und} \\ A(\mathbf{x}) & \mapsto \Theta(\epsilon_A - A(\mathbf{x})) & \text{als „Funktion“ von } A. \end{array} \tag{C-13}$$

3. Der Anteil α der gelernten Datenpunkte ist

$$\alpha := \int p\,\omega_A. \tag{C-14}$$

4. Daraus ergibt sich wegen der Definition (C-11) die Verteilungsdichte $\tilde{p}$ aller gelernten Punkte, Gleichung (4-11),

$$\tilde{p} = \frac{p\,\omega_A}{\int p\,\omega_A} = \frac{1}{\alpha}\,p\,\omega_A. \tag{C-15}$$

Der Algorithmus (TS-1) bis (TS-6) extremiert das Funktional

$$E[A] = \int\limits_{\mathcal{M}} \tilde{p}[A]\,\ln A = \frac{1}{\alpha}\int\limits_{\mathcal{M}} p\,\Theta_\varepsilon(\epsilon_A - A)\,\ln A \tag{C-16}$$

unter den Nebenbedingungen

$$\int\limits_{\mathcal{M}} A = 1 \tag{C-17}$$

$$\int\limits_{\mathcal{M}} \tilde{p} = \frac{1}{\alpha}\int\limits_{\mathcal{M}} p\,\Theta_\varepsilon(\epsilon_A - A) = 1. \tag{C-18}$$

Wie in (C-3) kann dieses mit Lagrange-Parametern geschrieben werden,

$$L[A] = E[A] \;+\; \lambda_1 \int A \;+\; \lambda_2 \int p\,\Theta_\varepsilon(\epsilon_A - A). \tag{C-19}$$

Variation dieses Funktionals und Nullsetzen liefert die Bedingung

$$0 \stackrel{!}{=} \frac{\delta L}{\delta A} = \frac{p}{\alpha}\frac{\Theta_\varepsilon(\epsilon_A - A)}{A} - \frac{p}{\alpha}\Theta'_\varepsilon(\epsilon_A - A)(\ln A + \lambda_2) + \lambda_1, \tag{C-20}$$

die eine Konsistenzbedingung für den stationären Zustand von A ist. Da Θ_ε im wesentlichen eine Heavyside-Funktion ist, kann man ohne die genaue Form von Θ_ε zu kennen – allein an der Differentialgleichung (C-20) – die gesuchten Approximationseigenschaften des stationären A finden.

- In Γ_o, wo $\Theta_\varepsilon(\epsilon_A - A) = 1$ und $\Theta'_\varepsilon(\epsilon_A - A) = 0$ gilt, dort ist auch

$$\frac{p(\mathbf{x})}{\alpha} = -\lambda_1 A(\mathbf{x}) \quad \text{für alle} \quad \mathbf{x} \in \Gamma_o. \tag{C-21}$$

- Die andere Möglichkeit, die Ableitung von Θ_ε verschwinden zu lassen, resultiert in $\Theta_\varepsilon(\epsilon_A - A) = 0$, woraus wiederum $A \gg \epsilon_A$ folgt. Dies widerspricht nicht nur direkt dem Mechanismus des ANTS-Algorithmus, der ja genau diejenigen Punkte $\mathbf{x}$ ignoriert, die zu $A(\mathbf{x}) \gg \epsilon_A$ führen könnten, sondern würde in Gleichung (C-20) auf

$$\lambda_1 = 0 \tag{C-22}$$

führen, was (C-21) widerspricht.

- Γ_i kann demnach nur aus Bereichen bestehen, wo $A(\mathbf{x})$ knapp über oder unter ϵ_A liegt. Wenn man dabei annimmt, dass (wegen des kleinen ε) alle Terme in Gleichung (C-20) klein gegen die Ableitung $\Theta'_\varepsilon(\epsilon_A - A)$ sind, dann folgt

$$A(\mathbf{x}) = e^{-\lambda_2} \quad \text{für alle} \quad \mathbf{x} \in \Gamma_i, \tag{C-23}$$

womit begründet ist, dass A in diesem Bereich eine Gleichverteilung approximiert. Da hier aber gleichzeitig $A \approx \epsilon_A$ gelten muss, weil oben der Fall $A \gg \epsilon_A$ auf Widersprüche führte, muss auch $e^{-\lambda_2} \approx \epsilon_A$ sein. Da mit Verkleinerung von ε die Funktion Θ_ε beliebig scharfkantig gemacht werden kann, kommt nichts anderes als

$$e^{-\lambda_2} = \epsilon_A \quad \text{bzw.} \quad \lambda_2 = -\ln \epsilon_A \tag{C-24}$$

in Frage.

Man hat nun die Höhe des Plateaus als ϵ_A bestimmt. Außerdem ist A in Γ_i proportional zu p. Wenn man nun voraussetzt, dass A am Übergang zwischen Γ_i und Γ_o stetig sei, und außerdem bemerkt, dass die einzige auf Eins normierte Funktion, die ebenfalls eine Plateauhöhe von ϵ_A hat und ansonsten proportional zu p ist, gerade der α-Stumpf p^c ist, dann folgt daraus die Behauptung

$$A = p^c. \tag{C-25}$$

Der erste Lagrange-Parameter ist dann gerade $\lambda_1 = -1$.

Weitere Überlegungen

Im letzten Abschnitt wurden nur qualitative Aspekte der Funktion Θ_ε aus Gleichung (C-20) benötigt, während sie nun genauer spezifiert werden soll. Die Aussagen und Rechenschritte oben können vermutlich alle durch Verwendung genügend glatter Funktionen p in einem mathematisch völlig exakten Sinne formuliert werden. Die folgenden Bemerkungen würden, um sie genau zu fassen, einen nicht zu rechtfertigenden Aufwand bedeuten. Alles was im Rest dieses Appendix gesagt wird, sollte nicht unter dem Aspekt der mathematischen Wahrheit gelesen werden, sondern als Versuch, die bisher verwendeten Größen $\tilde{p}$, A und ω_A genauer zu beschreiben.

Im strengen Sinne ist (C-20) keine gewöhnliche Differentialgleichung, sondern eine Differentialgleichung von nichtlinearen Operatoren in einem Funktionenraum; $\Theta_\varepsilon(\epsilon_A - A)$

wird als Funktion von A aufgefasst, nicht als Funktion von $\mathbf{x}$. Ohne auf die mathematischen Details einzugehen, liefert symbolische Integration über A die Lösung

$$\Theta_\varepsilon(\epsilon_A - A) = \frac{\alpha\lambda_1\,(\epsilon_A - A)}{p\,(\ln(A) + \lambda_2)} + \frac{B}{\ln(A) + \lambda_2}, \tag{C-26}$$

mit einer Integrationskonstanten B (ebenfalls eine Funktion von $\mathbf{x}$). Wenn man $\lambda_1 = -1$ und $\lambda_2 = -\ln\epsilon_A$ einsetzt, und außerdem als Randbedingung annimmt, dass

- als Gleichheit von Funktionen aus $A = p/\alpha$ (nur möglich für $\alpha = 1$) auch $\omega_A = 1$ folgt, dann bekommt man daraus die Funktion B als

$$B = \ln\frac{p}{\epsilon_A} + \frac{\epsilon_A}{p} - 1. \tag{C-27}$$

- Um α wieder in diese Formel zu schreiben, betrachtet man einzelne Punkte, $A(\mathbf{x}) = p(\mathbf{x})/\alpha$, was den endgültigen Ausdruck für B liefert,

$$B = \ln\frac{p}{\alpha\epsilon_A} + \frac{\alpha\epsilon_A}{p} - 1. \tag{C-28}$$

Somit ist die Lösung der Gleichung (C-20)

$$\omega_A(\mathbf{x}) = \Theta_\varepsilon(\epsilon_A - A(\mathbf{x})) = \frac{\alpha A(\mathbf{x}) - p(\mathbf{x})}{p(\mathbf{x})\ln\big(A(\mathbf{x})/\epsilon_A\big)} + \frac{\ln\big(p(\mathbf{x})/(\alpha\epsilon_A)\big)}{\ln\big(A(\mathbf{x})/\epsilon_A\big)}. \tag{C-29}$$

Diese Funktion hat folgendes Verhalten:

- für $p(\mathbf{x}) \approx \alpha A(\mathbf{x})$, also in Γ_o, wird die Randbedingung $\omega_A(\mathbf{x}) = 1$ reproduziert;
- für $p(\mathbf{x}) \approx \alpha\epsilon_A$ ist $B(\mathbf{x}) \approx 0$ (Übergangsbereich zwischen Γ_o und Γ_i), und

$$\omega_A(\mathbf{x}) = -\frac{\alpha\,(\epsilon_A - A)}{p\ln(A/\epsilon_A)} \tag{C-30}$$

  Dieser Ausdruck ist 1 für $A(\mathbf{x}) = \epsilon_A$, ist aber an dieser Stelle nicht nach Taylor entwickelbar, was eine Diskussion benachbarter Punkte schwierig macht.

- Für den Bereich $A(\mathbf{x}) \approx \epsilon_A$ wird es noch schwieriger. Hier divergiert ω_A wegen des logarithmischen Terms an jedem Punkt. Die „Funktion“ ω_A zeigt sich als Distribution. Sie hat dennoch das richtige Verhalten, denn das Integral über jede beliebige Menge $\Omega \subset \Gamma_i$ liefert

$$\int_\Omega \omega_A(\mathbf{x})d\mathbf{x} = \int_\Omega \frac{\alpha\epsilon_A}{p}, \quad \text{oder äquivalent} \tag{C-31}$$

$$\int_\Omega \tilde{p} = \int_\Omega \epsilon_A = \epsilon_A|\Omega| \quad \text{für alle } \Omega \subset \Gamma_i, \tag{C-32}$$

  wodurch sich wieder $A = p^c$ im Idealfall beliebig guter Approximation durch Gauß-funktionen ergibt. Die Begründung folgt unten.

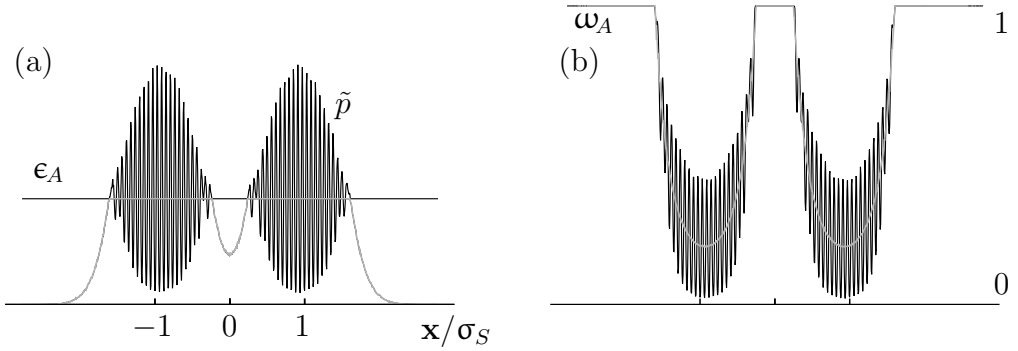


Abbildung 54: (a) Die tatsächlich gelernte Dichte $\tilde{p}$ und der α-Stumpf (grau übermalt), sowie (b) die Zuordnung ω_A der Punkte zum gelernten Datensatz eines ANTS-Trainings mit $M = 60$, $\varepsilon = 0.01$, $\sigma = 0.037\sigma_S$. Die Kurven sind im Vergleich zu Abbildung 42 wesentlich stärker gezackt, zeigen aber im Mittel das Verhalten der abgeschnittenen Dichte p^c (grau übermalt), wie auch in den Gleichungen (C-31) und (C-32) gefordert.

Die Abbildungen 42 und 54 liefern einen Eindruck, wie diese Approximation aussieht. In beiden Beispielen wurde der gleiche Datensatz gelernt, einmal von 12 und einmal von 60 Neuronen. Dementsprechend stark sind die gelernten Dichten und die Zuordnungsfunktionen ω_A gezackt.

Um die beiden Integrale (C-31) und (C-32) zu berechnen, muss man sich daran erinnern, dass A auf der Menge Γ_i eine Approximation von ϵ_A versucht. Da aber eine konstante Funktion nur dann durch Normalverteilungen beliebig gut approximiert werden kann, wenn diese auf einem regulären Gitter angeordnet sind, muss A etwa eine Jacobi'sche Thetafunktion sein (Adolphs, 1992). Für sehr kleine Gitterabstände l bleiben von einer Fourierentwicklung der Thetafunktion nur die ersten beiden Terme übrig, weshalb man für jede Dimension schreiben kann

$$A(x) \approx \epsilon_A \left(1 + 2 \lim_{l \to 0} e^{-2\pi^2 (\sigma/l)^2} \cos(2\pi x/l)\right) =: \epsilon_A + h(x), \quad \mathbf{x} \in \Gamma_i \tag{C-33}$$

Um nun das linke Integral in (C-32) auszurechnen, integriert man über $\Omega \subset \Gamma_i$,

$$\int_\Omega \tilde{p} = \int_\Omega \frac{p}{\alpha} \omega_A = \int_\Omega \frac{1}{\ln A/\epsilon_A} \left(A - \frac{p}{\alpha} + \frac{p}{\alpha} \ln \frac{p}{\alpha \epsilon_A}\right). \tag{C-34}$$

Wegen $A(x) \to \epsilon_A$ reicht die lineare Entwicklung des Logarithmus aus, und da man das Integrationsvolumen in kleine Teilvolumina Ω_i zerlegen kann, reicht auch eine lineare Entwicklung der (glatten) p-Terme. Ein solches Integral sieht dann folgendermaßen

aus:

$$h(x) := A(x) - \epsilon_A, \qquad \frac{1}{\ln A/\epsilon_A} \approx \frac{\epsilon_A}{h}, \tag{C-35}$$

$$\begin{aligned}\int_{\Omega_i} \tilde{p} &= \int_{\Omega_i} \frac{\epsilon_A}{h}\left(\epsilon_A + h + \frac{p}{\alpha} + \frac{p}{\alpha}\ln\frac{p}{\alpha\epsilon_A}\right) \\ &\approx \int_{\Omega_i} \epsilon_A + \gamma_1 \int_{\Omega_i} \frac{1}{h(x)}dx + \gamma_2 \int_{\Omega_i} \frac{x}{h(x)}dx \\ &= \int_{\Omega_i} \epsilon_A,\end{aligned} \tag{C-36}$$

mit nicht näher spezifizierten reellen Zahlen γ_1 und γ_2. Der letzte Schritt folgt aus der Symmetrie des Cosinus in (C-33) bezüglich der x-Achse, woraus $\int 1/\cos = 0$ folgt, und daraus, dass er, weil er in Ω_i beliebig oft oszilliert, o. E. auch gerade ist, $\int x/\cos(x)dx = 0$.

Demnach ist $\tilde{p}$ zwar eine Distribution und nimmt als solche an keiner Stelle den Wert ϵ_A an, doch hat sie, als Verteilungsdichte betrachtet – die ja eigentlich nur unter einem Integral verwendet wird – dieselben Eigenschaften wie wie p^c.

Literatur

Abramowitz, M. & Stegun, I. (Hrsg.) (1972). *Handbook of Mathematical Functions*. New York: Dover Publications.

Adolphs, U. (1992). Selbstorganisierende Perzeptronstrukturen mit radialen Basisfunktionen. Diplomarbeit, Fakultät für Physik der Ludwig-Maximilians-Universität, München.

Albrecht, S. (1999). *Multivariate GRBF-Netzwerke und Systeme lokaler Experten: Neue Konzepte und ihre Anwendung bei komplexen Regelungsaufgaben*. Dissertation, Fakultät für Physik der Ludwig-Maximilians-Universität, München.

Albrecht, S., Busch, J., Kloppenburg, M., Metze, F., & Tavan, P. (2000). Generalized Radial Basis Function Networks for Classification and Novelty Detection Self-Organization of Optimal Bayesian Decision. *Neural Networks 13*, 29–47.

Anderson, J. A. & Rosenfeld, E. (Hrsg.) (1988). *Neurocomputing: Foundations of Research*. Cambridge: MIT Press.

Bailey, C. H. & Kandel, E. R. (1985). Molecular Approaches to the Study of Short-Term and Long-Term Memory. In C. W. Coen (Hrsg.), *Functions of the Brain*, Kapitel 5, S. 98–129. Oxford: Clarendon Press.

Bandelow, C. (1989). *Einführung in die Wahrscheinlichkeitstheorie*. Mannheim: BI Wissenschaftsverlag.

Benveniste, A., Métivier, M., & Priouret, P. (1990). *Adaptive Algorithms and Stochastic Approximations*. Berlin: Springer-Verlag.

Bishop, C. M. (1995). *Neural Networks for Pattern Recogniton*. New York: Oxford University Press.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society 39*, 1–38.

Dersch, D. R. (1995). *Eigenschaften neuronaler Vektorquantisierer und ihre Anwendung in der Sprachverarbeitung*. Dissertation, Fakultät für Physik der Ludwig-Maximilians-Universität, München.

Duda, R. O. & Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. New York: John Wiley.

Fischer, H. & Kaul, H. (1990). *Mathematik für Physiker 1*. Stuttgart: Teubner.

Hebb, D. O. (1949). *The Organization of Behaviour*. New York: Wiley. Siebte Druckauflage, 1964.

Heskes, T. & Wiegerinck, W. (1998). On-line Learning with Time-Correlated Examples. In D. Saad (Hrsg.), *On-line Learning in Neural Networks*, Kapitel 12, S. 251–278. Cambridge, UK: Cambridge University Press.

Heskes, T. M. & Kappen, B. (1993). On-Line Learning Processes in Artificial Neural Networks. In J. G. Taylor (Hrsg.), *Mathematical Approaches to Neural Networks*, S. 199–233. Amsterdam: Elsevier Science Publishers B.V., North-Holland.

Heskes, T. M., Slijpen, E. T., & Kappen, B. (1993). Cooling Schedules for Learning in Neural Networks. *Physical Review E 47*(6), 4457–4464.

Hopkins, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academic Society USA 79*, 2554–2558. Abgedruckt in Anderson & Rosenfeld (1988).

Hubel, D. H. & Wiesel, T. N. (1974). Sequence Regularity and Geometry of Orientation Columns in the Monkey Striate Cortex. *Journal of Comparative Neurology 158*(3), 267–294.

Jaynes, E. T. (1957). Information Theory and Statistical Mechanics. *Physical Review 106*(4), 620–630.

Jaynes, E. T. (1963). Information Theory and Statistical Mechanics. In *Statistical Physics 3*, New York, Amsterdam. Brandeis Summer Institute 1962: Benjamin Inc.

Klingbeil, E. (1988). *Variationsrechnung*. Mannheim: B.I. Wiss.-Verlag.

Kloppenburg, M. (1996). Lokale Hauptkomponentenanalyse in hochdimensionalen Merkmalsräumen. Diplomarbeit, Institut für Medizinische Optik, Ludwig-Maximilians-Universität, München.

Kloppenburg, M. & Tavan, P. (1997). Deterministic Annealing for Density Estimation by Multivariate Normal Mixtures. *Physical Review E 55*(3), 2089–2092.

Kohonen, T. (1982). Self-organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics 43*, 59–69. Abgedruckt in Anderson & Rosenfeld (1988).

Kullback, S. & Leibler, R. A. (1951). On Information and Sufficiency. *Annals of Mathematical Statistics 22*, 79–86.

McCulloch, W. S. & Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bull Math Biophys 5*, 115–133. Abgedruckt in Anderson & Rosenfeld (1988).

Oja, E. (1982). A Simplified Neuron Model as a Principal Component Analyzer. *Journal of Mathematical Biology 15*, 267–273.

Rieke, F., Warland, D., de Ruyter van Steveninck, R., & Bialek, W. (1999). *Spikes: Exploring the Neural Code*. Cambridge, MA: MIT Press.

Ritter, H., Martinetz, T., & Schulten, K. (1992). *Neuronale Netze – Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke*. Bonn: Addison-Wesley.

Rose, K., Gurewitz, E., & Fox, G. C. (1990). Statistical Mechanics and Phase Transitions in Clustering. *Physical Review Letters 65*(8), 945–948.

Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review 65*, 386–408. Abgedruckt in Anderson & Rosenfeld (1988).

Rubner, J. & Tavan, P. (1989). A Self-Organizing Network for Principal Component Analysis. *Europhysics Letters 10*(7), 693–698.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature 323*, 533–536. Abgedruckt in Anderson & Rosenfeld (1988).

Schmidt, R. F. & Schaible, H.-G. (Hrsg.) (2000). *Neuro- und Sinnesphysiologie* (4. Aufl.). Berlin: Springer-Verlag.

Schmidt, R. F., Thews, G., & Lang, F. (2000). *Physiologie des Menschen* (28. Aufl.). Berlin: Springer-Verlag.

Sejnowski, T. J. & Tesauro, G. (1989). The Hebb Rule for Synaptic Plasticity: Algorithms and Implementations. In J. H. Byrne & W. O. Berry (Hrsg.), *Neural Models of Plasticity*, Kapitel 6, S. 94–103. San Diego: Academic Press.

Sonner, M. (1997). Selbstorganisation paralleler Experten zur Identifikation und Repräsentation wechselnder Systemzustände. Diplomarbeit, Institut für Medizinische Optik, Ludwig-Maximilians-Universität, München.

Tapia, R. A. & Thompson, J. R. (1978). *Nonparametric Probability Density Estimation*. Baltimore: John Hopkins University Press.

Wilden, A. (1998). Mustererkennung, Restmengenklassifikation und deren Anwendung in der Sprachverarbeitung. Diplomarbeit, Fakultät für Physik der Ludwig-Maximilians-Universität, München.

Willshaw, D. J. & von der Malsburg, C. (1976). How Patterned Neural Connections Can Be Set up by Self-organization. *Proceedings of the Royal Society, London, B 194*, 431–445.

Notation

Abkürzungen:

NN	Neuronales Netz
ANN	Adaptives oder lernendes Neuronales Netz
MVNN	*multivar*-Netzwerk
PCA	Hauptkomponentenanalyse oder *principal component analysis*
GF	Gaußfunktion oder Glockenkurve in beliebig vielen Dimensionen
RBF	Netzwerk, das aus radialen Basisfunktionen aufgebaut ist, hier sind das meistens Gaußfunktionen
V1	Primärer visueller Kortex
ML	*Maximum-likelihood*
EM	*Expectation Maximization*

Einführung und mathematische Abstraktionen:

a_r	Aktivität der r-ten Zelle
$\mathbf{a}$	Zustand des NN, Vektor aller Aktivitäten
$\boldsymbol{\theta}$	Netzwerkparameter
$\mathcal{A}$	Abbildung von altem auf neuen Netzwerkzustand
$\mathcal{P}$	Lernregel für die Netzwerkparameter
ε	Lernparameter von $\mathcal{P}$
q	Index der Nervenzellen auf der Eingabeschicht
h_q	Aktivität der q-ten Zelle auf der Eingabeschicht
$\mathbf{h}$	Ein Reiz von außen, Vektor der h_q
$\mathbf{z}_q$	Ort der q-ten Zelle z. B. auf der Retina
r	Index der Nervenzellen auf der unteren Schicht
$\mathbf{x}$	Ort eines Bestreizes $\mathbf{h}$ im Merkmalsraum $\mathcal{M}$
S_{rq}	Gewicht der synaptischen Verbindung von q nach r

Zum Dichteschätzer:

p	Verteilungsdichte der Daten
$\mathbf{x}$	ein zu lernender Datenpunkt
$\mathcal{X}$	der ganze Datensatz $\mathcal{X} = (\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(T))$
T	Anzahl aller gezeigter Datenpunkte
$\mathcal{M}$	Merkmalsraum der Daten
$\hat{p} \equiv A$	Schätzung von p durch den Lerner
P_r	a-priori Wahrscheinlichkeit, dass das $r$te Neuron angesprochen wird
M	Anzahl der Gaußfunktionen in der Schätzung $\hat{p}$

$\mathbf{c}_r$ Zentren der Gaußfunktionen bei einer Schätzung. Alle M Zentren zusammen bilden das *Codebuch*.
σ_{ri} Gaußbreite des $r$ten Neurons in Richtung der $i$ten Hauptachse
σ Breite der Gaußfunktionen im univariaten Fall, sonst Wert, an den die σ_{ri} gebunden werden.
$\mathbf{w}_{ri}$ Die $i$te Hauptachse des $r$ten Neurons
Σ_r Die Kovarianzmatrix des $r$ten Neurons, $\Sigma_r = \sum_{i=1}^{d} \sigma_{ri}^2 \mathbf{w}_{ri} \mathbf{w}_{ri}^T$
θ Zusammenfassung aller $\mathbf{c}_r$, $\mathbf{w}_{ri}$ und σ_{ri}
ε Lernrate in der verwendeten kompetitiven Hebb'schen Lernregel
$\mathbf{R}$ Größe des Rauschterms in der Lernregel für die Zentren, um Entartungen aufzuheben
E Einheitsmatrix

Zum Abschneide-Algorithmus:
α Anteil der gelernten Daten an den Gesamtdaten
ϵ_A Schwelle für $A(\mathbf{x})$, die zur Entscheidung dient, ob $\mathbf{x}$ gelernt wird oder nicht
$\tilde{p}$ Verteilungsdichte der tatsächlich gelernten Datenpunkte
$\omega_A(\mathbf{x})$ Erwartungswert für $f(\mathbf{x})$ für gegebenes $\mathbf{x}$
$\Theta_\varepsilon(\epsilon_A - \hat{p})$ Der gleiche Erwartungswert, jedoch als Funktion von $\hat{p}$ aufgefasst und mit Betonung der Eigenschaft von $\omega_A(\mathbf{x})$ als etwas verwaschene Heavyside-Funktion
Γ_o Der „äußere“ Bereich von $\mathcal{M}$, wo alle Punkte gelernt werden
Γ_i Der „innere“ Bereich von $\mathcal{M}$, wo Punkte selektiert werden

Ein Dankeschön

Ich möchte an dieser Stelle die Gelegenheit nutzen, allen Mitarbeitern des Lehrstuhls für BioMolekulare Optik meinen Dank auszusprechen für die Unterstützung, die sie mir im Laufe meiner Arbeit haben zuteil werden lassen.

Herrn Prof. Dr. Paul Tavan möchte ich für die Möglichkeit danken, im letzen Jahr an einem spannenden und anregenden Thema zu arbeiten. Für die geleistete Betreuung und die viele Zeit, die er meiner Arbeit während der schriftlichen Ausarbeitungsphase widmete, möchte ich mich erkenntlich zeigen.

Für die äußerst angenehme Arbeitsatmosphäre sei in erster Linie meinem Zimmerkollegen Andi Weiß gedankt, der sich allem Anschein nach in jeder noch so entlegenen Ecke literarischen Schaffens bestens auskennt. Ihm und Paul Strodel sowie den weiteren Teilnehmern des regelmäßig stattfindenden Medienkompetenzseminars sei für ihre leider vergeblichen Versuche gedankt, mich in die wirklich wichtigen Dinge des täglichen Geschehens einzuweihen. Bei allen anderen Mitgliedern der Arbeitsgruppe *Theoretische Biophysik*, Tassilo Christ, Bernhard Egwolf, Gerald Mathias, Heiko Carstens, Martina Stork, Thomas Hirschberger, Niels Lange, Mattias Schmitz und Marco Nonella, möchte ich mich für das freundschaftliche Arbeitsklima und die große Hilfsbereitschaft bedanken.

Mein besonderer Dank für die wöchentliche Abreibung im Cosimabad gilt Erik Baigar, der mir auch in allen anderen Fragen – besonders beim Thema *Schokolade* – immer hilfreich zur Seite stand.

Erklärung

Hiermit erkläre ich, Michael Schindler, geboren am 13. 10. 1976 in Basel, in der Schweiz, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Michael Schindler